The Kelkar Education Trust's
V G Vaze College of Arts, Science and Commerce
(Autonomous)

Live Cricket Scoring

A Project Report
Submitted in partial fulfilment of the Requirements for the award of the Degree of

BACHELAOR OF SCIENCE
(INFORMATION TECHNOLOGY)

By

Pushkar Prasad Sane
3945A048

Under The Esteemed Guidance of
Mrs. Rakhee Rane
Assistant Professor

DEPARTMENT OF INFORMATION TECHNOLOGY
V G VAZE COLLEGE OF ARTS, SCIENCE AND COMMERCE
(AUTONOMOUS)
(Affiliated to University of Mumbai)
Mulund, 400080 MAHARASHTRA
2022-2023

THE KELKAR EDUCATION TRUST'S VINAYAK GANESH VAZE COLLEGE OF ARTS, SCIENCE AND COMMERCE

(Autonomous)

MULUND, MAHARASHTRA, 400081

DEPARTMETN OF INFORMATION TECHNOLOGY

CERTIFICATE

This is to certify that the project entitled," LIVE CRICKET SCORING ", is bonafied work of MR. PUSHKAR PRASAD SANE bearing Roll No: 3945A048 submitted in partial fulfillment of the requirements for the award of degree of BACHELOR OF SCIENCE in INFORMATIONTECHNOLOGY.

04/03/2023

Internal Guide

Head Of Department

8/3/23.

External Guide

8 MAR 2023

K.E.T.'S VAZE COLLEGE 1984

Date:

College Seal

THE APPROVAL PROJECT PROPOSAL

(Note:All entries of the proforma of approval should be filled up with appropriate and complete information. Incomplete proforma of approval in any respect will be summarily rejected.)

PNR No :

1. Name of the Student	
PUSHKAR PRASAD SANE	
2. Title of the Project	
LIVE CRICKET SCORING	
3. Name of the Guide	
MRS. RAKHEE RANE	
4. Is this your first submission?	
YES	
Student Name :	PUSHKAR PRASAD SANE
Student Control ID :	2021070007
Student Roll No. :	3945A048

Rakhee

ACKNOWLEDGEMENT

I am glad to say that, I have satisfactorily reached my intentions to make this documentation. However, it would not have been possible without the kind support and help of many individuals. I would like to extend my sincere thanks to all of them.

I am highly indebted of my guide, **Mrs. Rakhee Rane** for her guidance and constant supervision as well as for providing necessary information regarding the project.

I would also like to extend my gratitude towards **Principal (Prof) Dr. Preeta Nilesh** and the Head of the Department, **Mrs. Pournima Bhangale** for providing me with all the facilities that was required.

Then I would like to thank my parents who have helped me with their valuable suggestions and guidance which has been very helpful.

Last but not the least I would like to thank my classmates who have helped me a lot. Directly or indirectly their contribution was indispensable, and will always be remembered.

This opportunity has given me a valuable experience about software development for which I shall be thankful for the years to come.

~Pushkar Prasad Sane

DECLARATION

I here by declare that the project entitled, "LIVE CRICKET SCORING

" done at place where the project is done, has not been in any case duplicated to submit to any other university for the award of any degree. To the best of my knowledge other than me, none has submitted to any other university. The project is done in partial fulfillment of the requirements for the award of degree of BACHELOR OF SCIENCE (INFORMATIONTECHNOLOGY) to be submitted as final semester project as part of our curriculum.

Name : PUSHKAR PRASAD JANE

Signature :

Live Cricket Scoring (Scorify)

Table of Contents:

List of Figures:

List of Tables:

Synopsis

Title:

Live Cricket Scoring Codename: Scorify

Problem Statement:

To create a live cricket scoring app. Here, various types of functions are provided such as ball-by-ball scoring, professional scorecard etc. It will provide ball-by-ball coverage of all domestic and other tournaments that are conducted.

Why this Topic?

Cricket scoring is one of the most complex of all games scoring. It has so many permutations and combinations that it becomes tedious task for the scorer to do it on a paper.

Objective and Scope.

- To reduce multiple manual calculations during the match such as net run rate, batsman's analysis, bowler's analysis.
- To provide ball-by-ball update of the match.
- To provide detailed scorecard.

Methodology for developing project.

In this system, I am going to use Extreme Programming for developing an appropriate system as a solution for rapidly changing requirements

Advantages: Communication, Simple, Easy, Agile.

Proposed Architecture

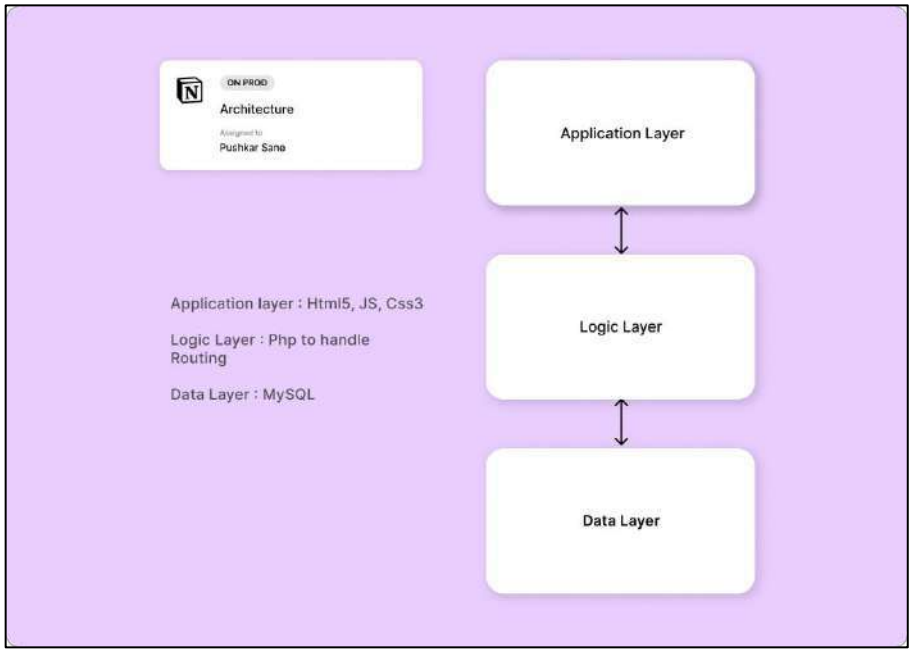


Figure 1.1 Architecture

Requirements

Software Requirements

- Front-end: HTML5, CSS, JS, Bootstrap
- Back-end: Php, MySQL.
- Operating System: Windows 7.0 +

Hardware Requirements

- Processor: Intel Core Duo 2.0 GHz or more.
- RAM: 2 GB or more.
- Monitor: 17 CRT or LCD.
- Hard disk: 500 GB or more.
- Keyboard: Normal or multimedia.

Platform

Visual Studio Code

Contribution

As cricket scoring is one the most complex of all games scoring, many permutations and combinations are to be considered and multiple manual calculations to calculate, this app will make it simple and easy to do scoring with a click of button.

Conclusion

This system will help to reduce paperwork and will make work of scorers easy.

Chapter 1: Introduction

1.1 Background

In cricket, a scorer is someone appointed to record all the events taking place before, during and after the match. It is possible to record this using a pen and plain paper. Scorers often use printed score books. Sometimes the scorers also produce their own scoring sheets to suit their techniques and some use coloured pens to highlight events such as wickets, extras, etc.

1.2 Objective

On the day of the match there are multiple manual calculations to calculate such as batting analysis for each batsman, bowling analysis for each bowler, etc. Hence, this project will help the scorers to make their task easy with a click of a button.

1.3 Purpose

The purpose behind making this project is to make task of scorers easy. As cricket scoring is one of the most complex of all games scoring. It has many permutations and combinations that is becomes a tedious task for scorers to do it on paper or scorebook.

1.4 Application

The idea can be fundamentally used in any management score of matches in cricket games.

1.5 Scope

Creating a platform for all domestic producers to sell their vaccines to local consumers, NGOs, etc, and to simplify the delivery process, regulations and management.

1.6 Achievements

It will be applicable for clubs matches, practice games for various formats like T20, One day and multi-day games.

Chapter 2: Survey of Technologies

The number of Technologies available for the implementation is listed below:

1. **Front-end Languages:**
    a) HTML 5
    b) CSS
    c) JavaScript
    d) ASP.Net
    e) Bootstrap
    f) Python

a) **HTML 5**: HTML5 is a markup language used for structuring and presenting content on the World Wide Web. HTML5 includes detailed processing models to encourage more interoperable implementations; it extends, improves, and rationalizes the markup available for documents and introduces markup and application programming interfaces (APIs) for complex web applications.

b) **CSS**: Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language such as HTML. CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript. CSS is designed to enable the separation of the presentation and content, including layout, colours, and fonts. This separation can improve content accessibility, provide more flexibility and control in the specification of presentation characteristics, enable multiple web pages to share formatting by specifying the relevant CSS in a separate .css file which reduces complexity and repetition in the structural content as well as enabling the .css file to be cached to improve the page load speed between the pages that share the file and its formatting.

c) **JavaScript**: JavaScript is the Programming Language for the Web. JavaScript can update and change both HTML and CSS. JavaScript can calculate, manipulate and validate data. It is a dynamic computer programming language. It is lightweight and most commonly used as a part of web pages, whose implementations allow client-side script to interact with the user and make dynamic pages. It is an interpreted programming language with object-oriented capabilities.

d) **ASP.Net**: ASP.NET is a web development platform, which provides a programming model, a comprehensive software infrastructure and various services required to build robust web applications for PC, as well as mobile devices. ASP.NET works on top of the HTTP protocol, and uses the HTTP commands and policies to set a browser-to-server bilateral communication and cooperation. ASP.NET is a part of Microsoft .Net platform. ASP.NET applications are compiled codes, written using the extensible and reusable components or objects present in .Net framework. These codes can use the entire hierarchy of classes in .Net framework.

e) **Bootstrap**: Bootstrap is a giant collection of handy, reusable bits of code written in HTML, CSS and JavaScript. It's also a front-end development framework that enables developers and designers to quickly build fully responsive website. Bootstrap includes user interface components, layouts and JS tools along with the framework for implementation.

f) **Python**: Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built-in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse.

2. **Back-end Languages:**
    - PHP
    - MySQL
    - MongoDB

a) **Php**: PHP (recursive acronym for PHP: Hypertext Pre-processor) is a widely-used open-source general-purpose scripting language that is especially suited for web development and can be embedded into HTML. PHP is a server-side scripting language that is used to develop Static websites or Dynamic websites or Web applications. PHP scripts can only be interpreted on a server that has PHP installed.

b) **MySQL**: MySQL is a relational database management system (RDBMS) developed by Oracle that is based on structured query language (SQL). MySQL is a fast, easy-to-use RDBMS being used for many small and big businesses. MySQL provides an implementation of a SQL database very well suited for small to medium web pages. A database is just a structured collection of data that is organized for easy use and retrieval. Common applications for MySQL include php and java-based web applications that require a DB storage backend.

c) **MongoDB**: MongoDB is an open-source document-oriented database that is designed to store a large scale of data and also allows it to work with that data very efficiently. It is categorized under the NoSQL (Not only SQL) database because the storage and retrieval of data in the MongoDB are not in the form of tables. The data model that MongoDB follows is a highly elastic one that lets users combine and store data of multivariate types without having to compromise on the powerful indexing options, data access, and validation rules.

For my current project, I am going to use PHP as a development platform for easy implementation of the requirements proposed.

Why PHP?

One of the major benefits of PHP is that it is platform independent. It can be used on Mac, Windows, Linux and supports most web browsers. It also supports major web servers, making it easy to deploy on different systems and platforms. It is a server-side scripting language embedded in HTML in its simplest form. PHP allows web developers to create dynamic content

Chapter 3: Requirement and Analysis

3.1 Problem Definition

What to expect about the system?

The system is for users who do scoring in cricket matches. It will be useful for scorers to maintain score as well as other records like bowling analysis, batting analysis and other records. On the day of match the scores will have to login, create a match then add players to respective teams and can quick off scoring the match as it goes on.

3.1.1 Sub-Systems / Sub-Problems

- Login / Registration:
  - User will be able to create a new account i.e., register themselves
  - Registered users can login directly into the app i.e., they will be granted access after verifying their credentials.
- Dashboard:
  - Dashboard will be used for creating a new match, view previous matches.
- Match Creation:
  - In here, the scorer will be able to create a new match, add teams and player in the teams.
- Live Scoring:
  - During the match, the umpire will give signal to the scorer and the scorer will make a note of it as the match progresses.
- Scorecard:
  - User will get scorecard for the match.
  - The user will be able to view the scorecard.
  - It will have details like batting analysis, bowing analysis, extras etc.

3.1.2 Problem Description.

This system is for digital scoring of a cricket match. It will make it simple for scorers to handle and manage the mathematical operations easily. This system covers important issues of the scorers having problems in managing the scoresheets, summary sheets and other papers.

3.2 Requirements Specification

3.2.1 Requirement Gathering

Various requirements gathering technologies include

- Brainstorming – To get as many ideas from group of people Generally used to identify possible solutions to problems & clarify details of opportunities.

- Interview – Interview of users are critical to create a great software without understanding the goals & expectations of the users, we are unlikely to satisfy them Listening is a skill that helps a great analyst to get more value from an interview than an average analyst.

- Observations – By observing users, an analyst can identify a process flow, pain points & opportunities for improvement. Observer can be passive or active. Passive observations are a better for getting feedback and a prototype whereas active observations are more effective at gathering and understanding an existing business process.

- Survey / Questionnaire – The survey can force users to select from choices, rate something or have open ended questions allowing free form responses.

I prepared a questionnaire using Google forms and look feedback from students about my project. The questions and responses were as follows.

3.2.2 Requirement analysis

Identify stakeholders i.e., in case the people who are going to use this site.

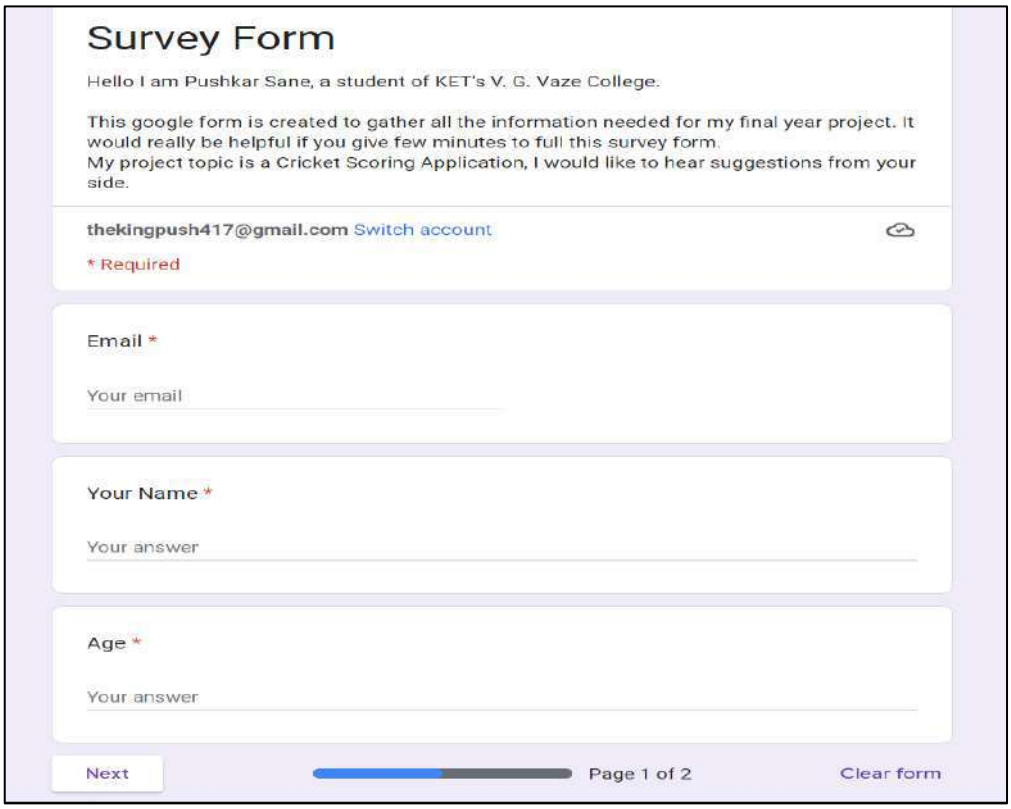
- Capture requirements
- Holding One-One interviews
- Conduct Group Workshops
- Get Feedbacks
- Build Small Prototypes

For the current situation, I used the Feedback method to identify the requirements for the project using Google Forms as a means to collect the data.

The link for spreadsheet of responses I got is below:

https://docs.google.com/spreadsheets/d/1TcOC6hnWNdawqq9xfcP0FMfusH_fRYPg2GTdW_PTRbY/edit?usp=sharing\

The below figures are the collected data that was generated.



Survey Form

Hello I am Pushkar Sane, a student of KET's V. G. Vaze College.

This google form is created to gather all the information needed for my final year project. It would really be helpful if you give few minutes to full this survey form.
My project topic is a Cricket Scoring Application, I would like to hear suggestions from your side.

thekingpush417@gmail.com Switch account

* Required

Email *

Your email

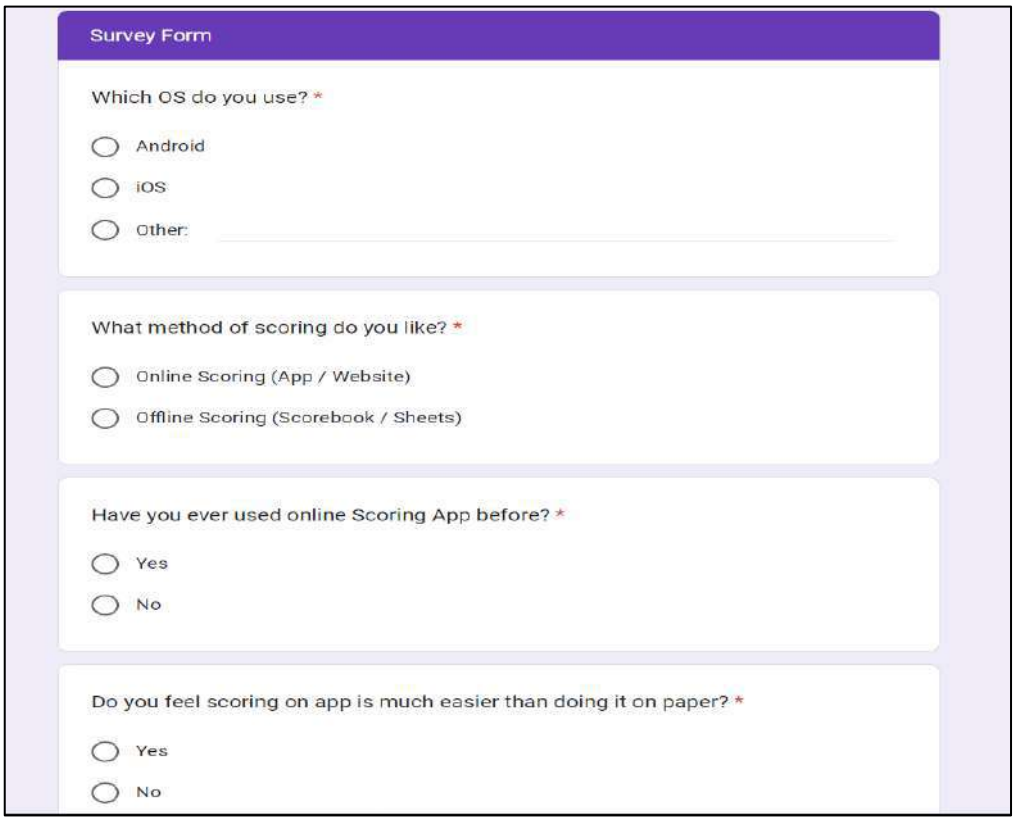
Your Name *

Your answer

Age *

Your answer

Next | Page 1 of 2 | Clear form



Survey Form

Which OS do you use? *

- Android
- iOS
- Other:

What method of scoring do you like? *

- Online Scoring (App / Website)
- Offline Scoring (Scorebook / Sheets)

Have you ever used online Scoring App before? *

- Yes
- No

Do you feel scoring on app is much easier than doing it on paper? *

- Yes
- No

What do you think about scoring application that will help you solve all problems? *

- It will be helpful
- Not sure
- Will be of no use

Do you have any other suggestions related to my app?

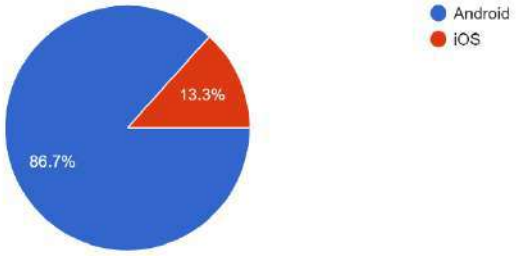
Your answer

Back | Submit | Page 2 of 2 | Clear form

Never submit passwords through Google Forms.

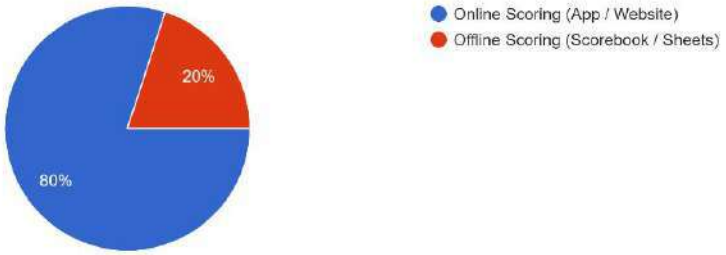
Which OS do you use?

60 responses



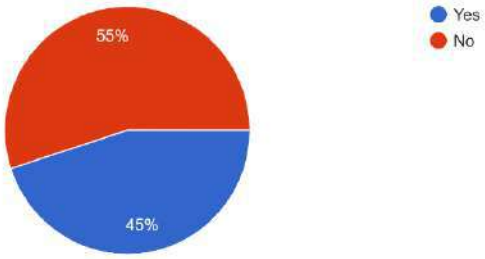
What method of scoring do you like?

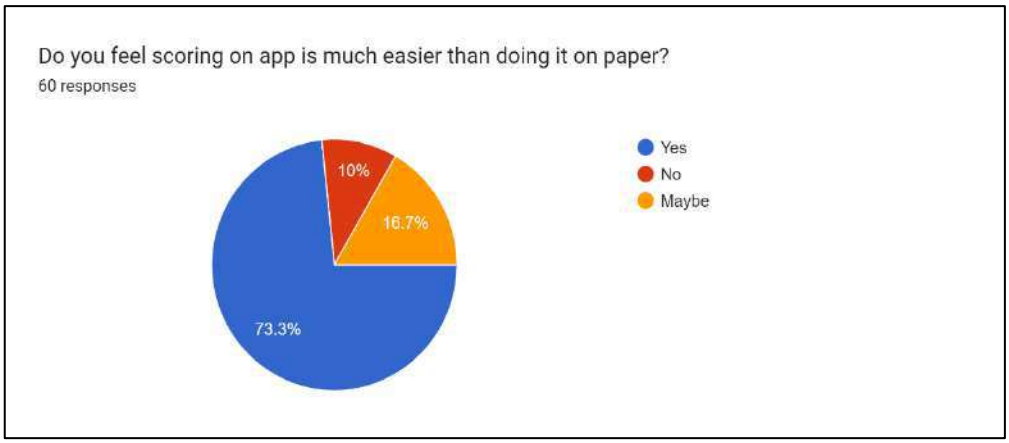
60 responses



Have you ever used online Scoring App before?

60 responses





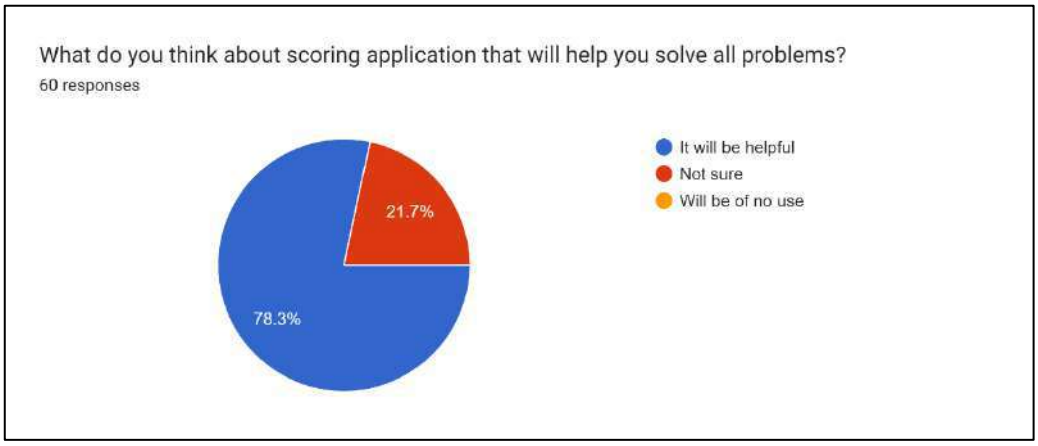


Fig. 3.1 Requirement Gathering

3.2.3 Functional Requirements

a. Login/ Register: The user should sign up i.e., create an account. After creation of account, they'll have to enter valid username and password to login and proceed further.

b. Ball-to-ball Scoring: The scorer should be able to maintain the ball-to-ball score of the ongoing match.

c. Professional Scorecard: After the match ends, the users should be able to view scorecard which includes batting analysis, bowling analysis, etc.

d. Responsive Design: User have different devices as a result the size of display varies from user to user. Hence, the UI of screen should be flexible that can adjust to different screen sizes.

3.2.4 Non-Functional Requirements

a. Usability: Should be user-friendly and only required detail should be shown in a minimal way.

b. Reliability: The system should be user friendly to use.

c. Flexibility: can run on any Platform.

3.2.5 System Requirements

1. Login:
    a) Description: The user will be able to login to their respective accounts.
    b) Input: Username, password
    c) Source: User
    d) Output: Gets logged in to the system.
    e) Destination: -
    f) Action: After entering username and password, the user will get redirected to the dashboard,
    g) Pre-condition: The user must have an account
    h) Post-condition: -
2. Register:
    a) Description: The user will be able to create a new account.
    b) Input: Name, Email, Username, Password.
    c) Source: User
    d) Output: Account gets created.
    e) Destination: Entered data will get stored in database.
    f) Action: After registering, the account of user gets created.
    g) Pre-condition: User must provide the required details.
    h) Post-condition: User can login to their account with registered username and password.
3. Create a match:
    a) Description: User will be able to create a new match.
    b) Input: Details of match, teams and players.
    c) Source: User.
    d) Output: New match is created.
    e) Destination: Data will be displayed and added on scorecard.
    f) Action: Match gets created as per the given details.
    g) Pre-condition: User must be logged in to their account:
    h) Post-condition: User will be able to add team and players.

4. Live Scoring:
    a) Description: User will be able to maintain score of ongoing matches.
    b) Input: Event happening on each ball.
    c) Source: User
    d) Output: Updates the score.
    e) Destination: User interface.
    f) Action: Score gets updated after each ball.
    g) Pre-condition: Match should be created/started.
    h) Post-condition: -
5. Scorecard (View):
    a) Description: User will be able to view the scorecard of completed match.
    b) Input: Select the match.
    c) Source: Database.
    d) Output: Scorecard is displayed.
    e) Destination: -
    f) Action: User gets to view the scorecard.
    g) Pre-condition: Match should be finished.
    h) Post-condition: -

3.3 Planning and Scheduling

3.3.1 Activity Table

Chapter Name	Start Date	End Date
• Project Synopsis	27-04-2022	16-06-2022
• Introduction	20-06-2022	25-06-2022
• Survey of Technologies	20-06-2022	25-06-2022
Requirement and Analysis • Problem Definition	27-07-2022	02-07-2022
Requirement Specification • Requirement Gathering	04-07-2022	09-07-2022
Requirement Analysis • Functional Requirements • Non-Functional Requirements • System Requirements	04-07-2022 04-07-2022 11-07-2022	09-07-2022 09-07-2022 16-07-2022
• Planning and Scheduling	18-07-2022	23-07-2022
• Hardware and Software Requirements	18-07-2022	23-07-2022
Conceptual Models • Entity-Relationship Diagram • Schema Diagram • Data Flow Diagram • Use Case Diagram	18-07-2022 25-07-2022 01-08-2022 29-08-2022	23-07-2022 30-07-2022 13-08-2022 10-09-2022

• Sequence Diagram • Activity Diagram • State Diagram	12-09-2022 12-09-2022 12-09-2022	17-09-2022 17-09-2022 17-09-2022
System Models • User Interface Design • Test Cases	 19-09-2022 19-09-2022	 24-09-2022 19-09-2022

Table 3.1 Activity Table

3.3.2 Gantt Chart

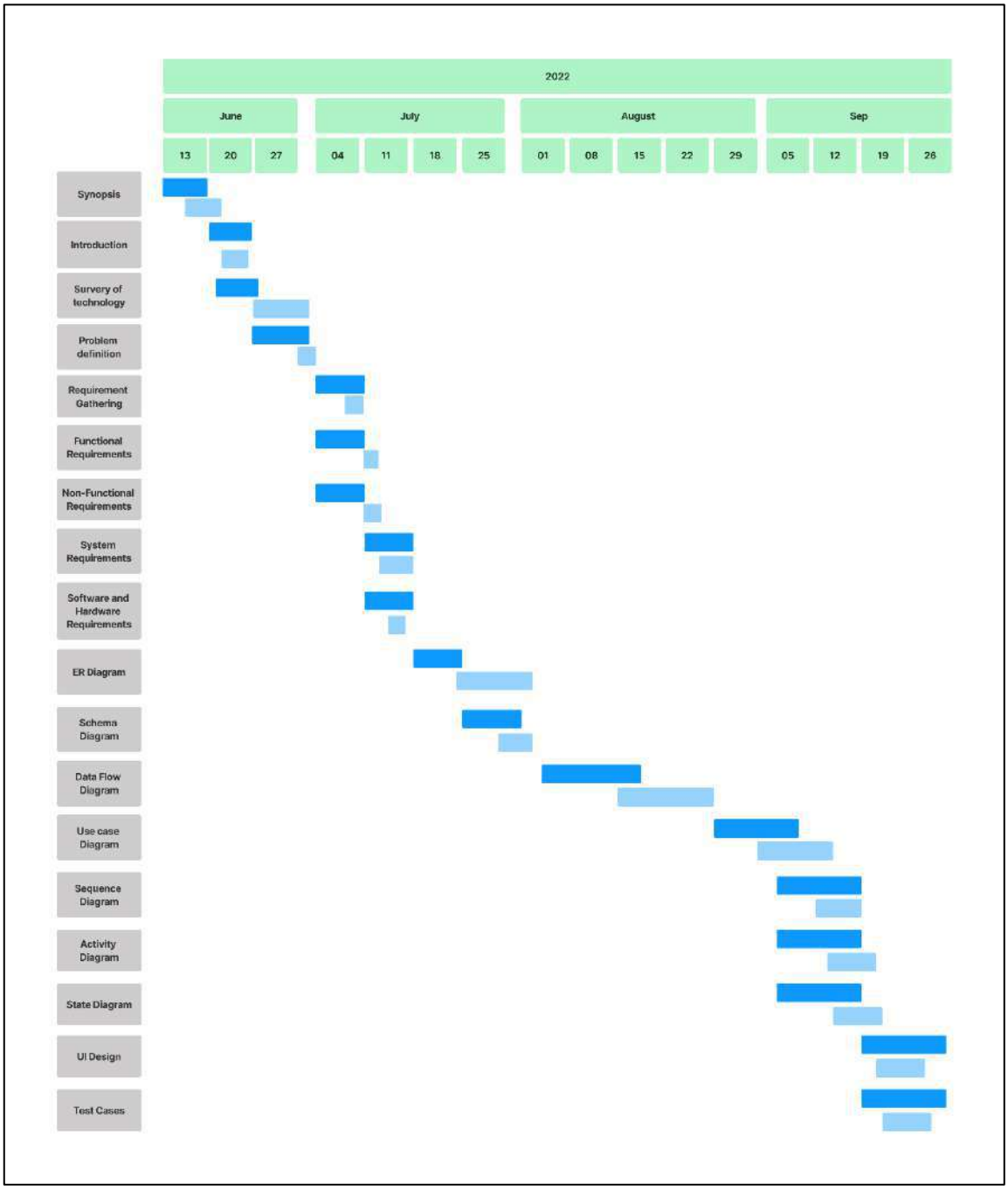


Fig. 3.2 Gantt Chart

Re-Engineering Table

Activity	Start Date	End Date
1) Re-engineering	01-10-2022	05-12-2022
2) Home Page	14/12/2022	17/12/2022
3) Register & Login	19/12/2022	22/12/2022
4) Dashboard	26/12/2022	28/12/2022
5) Match Creation	29/12/2022	06/01/2023
6) Live Scoring	09/01/2022	16/02/2022
7) Scorecard	18/02/2023	26/02/2023

Table 3.2 Re-engineering Table

Gantt Chart

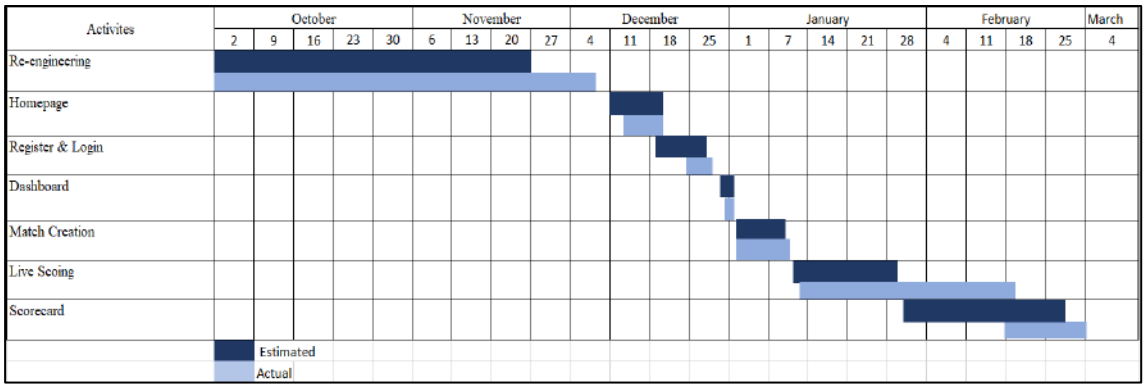


Fig. 3.3 Gantt Chart New

3.4 Hardware & software requirements

3.4.1 Hardware Requirements

- Processor: Intel Core 3.0 2.3 GHz or more.
- RAM: 4GB or more.
- Monitor: 17 CRT or LCD, Plasma, etc.
- Hard-Disk: 256 or more (SSD preferable)
- Keyboard: Normal or multimedia.
- Mouse: Compatible

3.4.2 Software Requirements

- System O.S: Window or Linux (Debian or Arch).
- Front-end: HTML, JS, CSS.

- Back-end: PHP.
- Database: MySQL.

3.5 Entity-Relationship Diagram

An entity relationship diagram shows the relationships of entity sets stored in a database. An entity in this context is an object, a component of data. An entity set is a collection of similar entities. These entities can have attributes that define its properties. In software engineering an ER model is commonly formed to represent things that a business needs to remember in order to perform business processes. Consequently, the ER model becomes an abstract data model that defines a data or information structure which can be implemented in a database, typically a relational database.

Symbol reference: Database System Concepts, "Henry F. Korth, Abraham Silberschatz, S.Sudarshan" McGraw-Hill 4th Edition.

Diagram Notations:

Name	Symbol	Description
Rectangle		Represents entity set
Ellipse		Represents attributes
Double Ellipse		Represent multivalued attributes
Diamond		Represents relationship set
Double Lines		Represents total participation
Double Rectangle		Represents weak entity

Table 3.3 ER-Diagram Notations

List of entity sets:

- User
- Match
- Player
- Team
- Score

List of relationship sets:

1. User crates a Game
2. Game has Teams
3. Team has Player
4. Player has Score

3.5.1 Entity Sets:

1. User: User is the scorer who is appointed to respective match. They will need to register and login. To register, user will need to provide details like email-id, name, password.

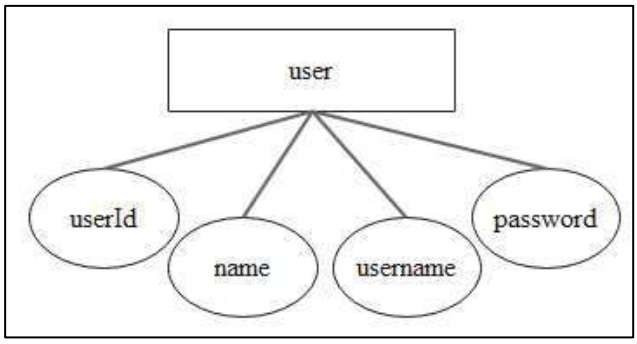


Fig. 3.4 User Entity Set

2. Game: This will include all the details of the match that is being played. Here, the user will need to provide venue, toss status, type of match, etc.

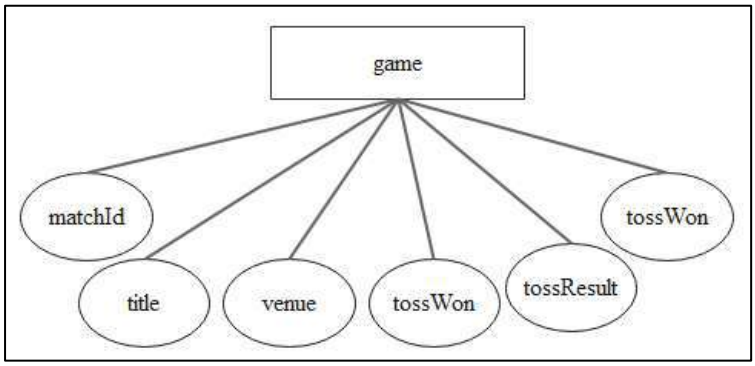


Fig. 3.5 Match Entity Set

3. Team: This will include the information related to the teams that will play the match.

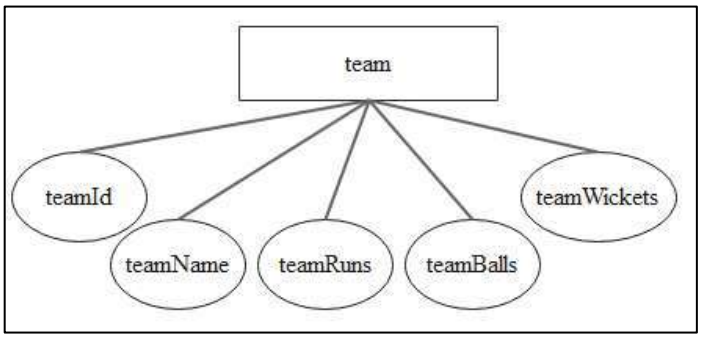


Fig. 3.6 Team Entity Set

4. Player: This will include all the information about the players will be playing that match. It will have player's name, role, etc.

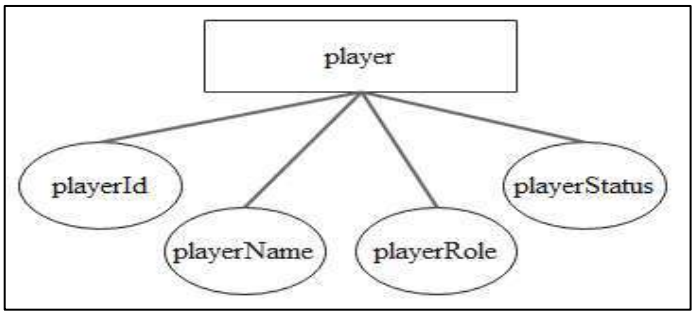


Fig. 3.7 Player Entity Set

5. Score: This will include the score of the match. It will have total score of teams and the result of the game.

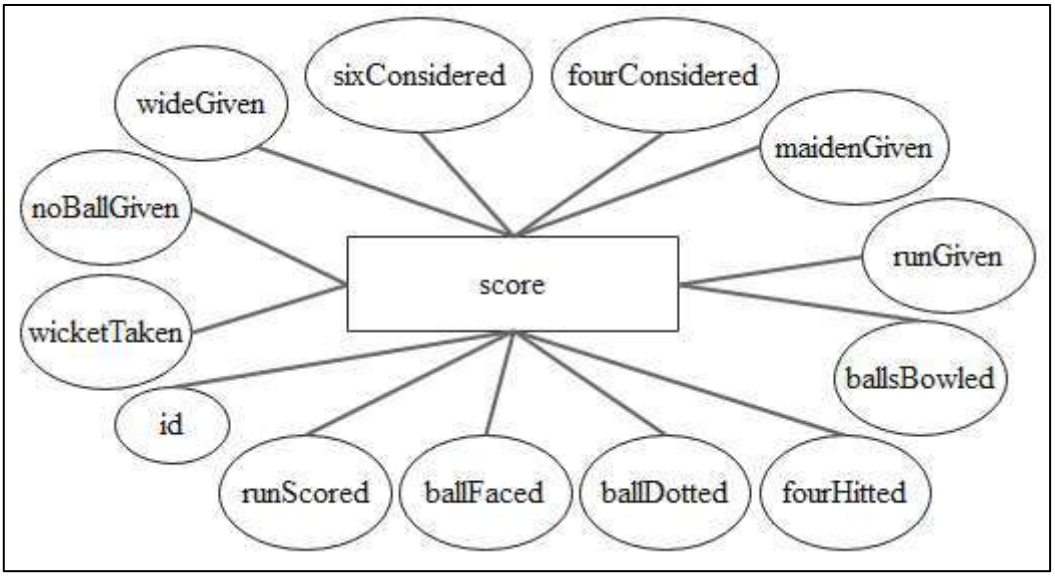


Fig. 3.8 Score Entity Set

3.5.2 Relationship Sets

A relationship is used to describe the relation between entities. Diamond or rhombus is used to represent the relationship.

1) User creates a Game
    a. User needs to create a match to do scoring. After creation of match, they can proceed with further process.
    b. Mapping Cardinality: One to one

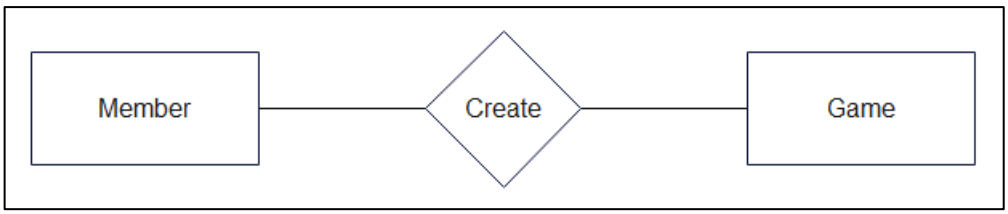


Fig. 3.9 Create Relationship Set

2) Game has Teams
    a. After creating a match, every match will have teams.
    b. Mapping Cardinality: One to Many

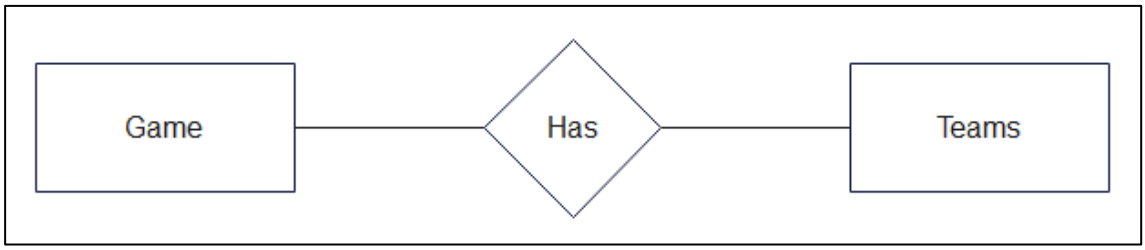


Fig. 3.10 Has Relationship Set

3) Team has players
    a. There are 2 teams in a match. Each team has 11 players.
    b. Mapping Cardinality: One to Many

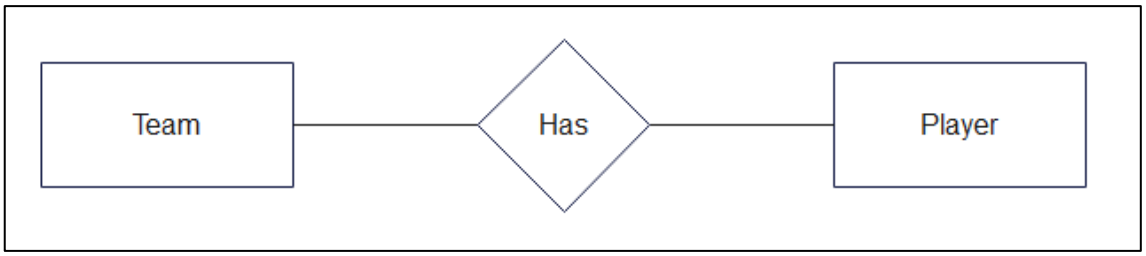


Fig. 3.11 Has Relationship Set

4) Player has Score
    a. Here a match can have only one score
    b. Mapping Cardinality: One to One

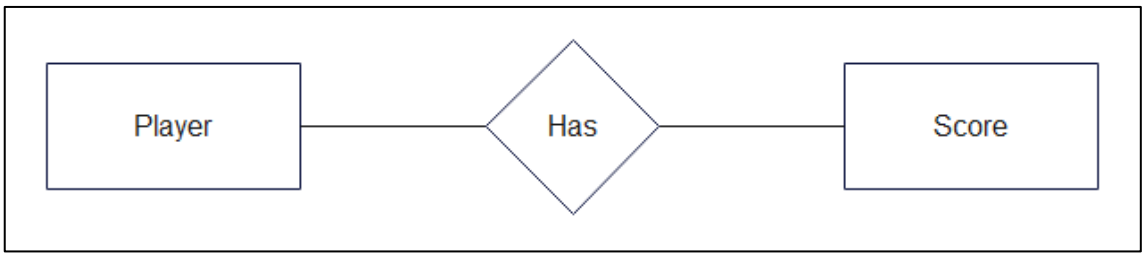


Fig. 3.12 Has relationship set

3.5.1.3 Entity-Relationship Diagram

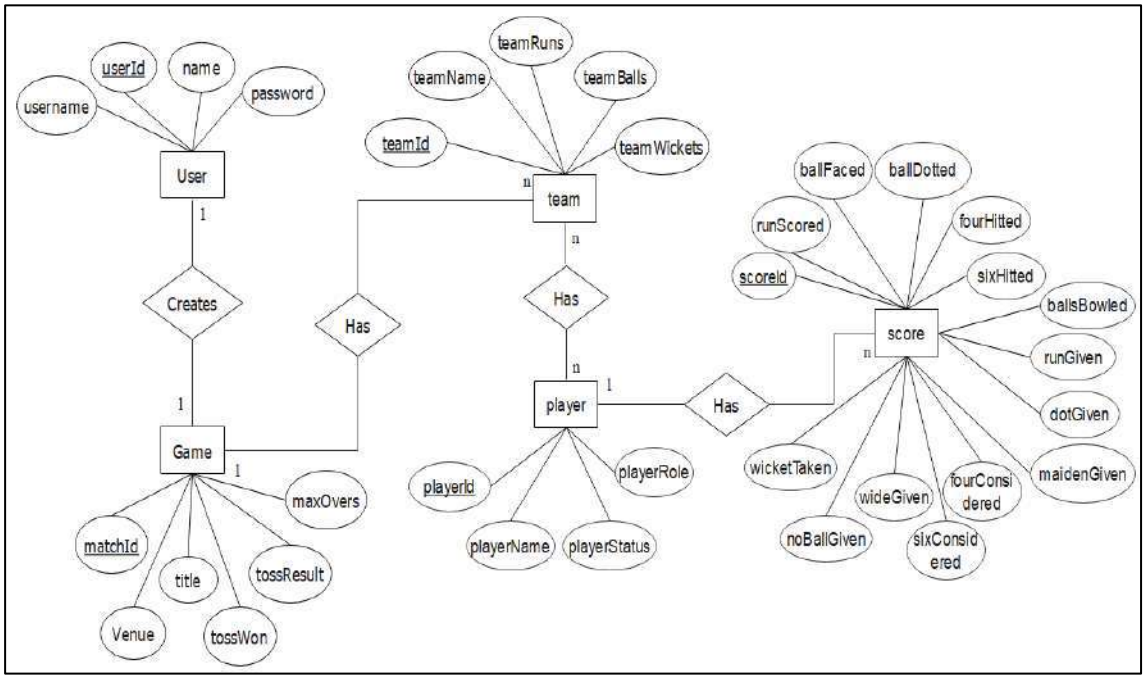


Fig. 3.13 ER Diagram

3.5.2 Schema Diagram

A database schema is the skeleton structure that represents the logical view of the entire database. It defines how the data is organised and how the relations among them are associated. It formulates all the constraints that are to be applied on the data.

Symbol reference: https://www.lucidchart.com/

Name	Symbol	Description
Table		A table is a collection of related data held in table format within a database.
Relation		In a relational database system, a one-to-one table relationship links two tables based on a Primary Key column in the child which is also a Foreign Key referencing the Primary Key of the parent table row. Therefore, we can say that the child table share the Primary Key with the parent table.

Table 3.4 Schema Diagram Notations

Diagram:

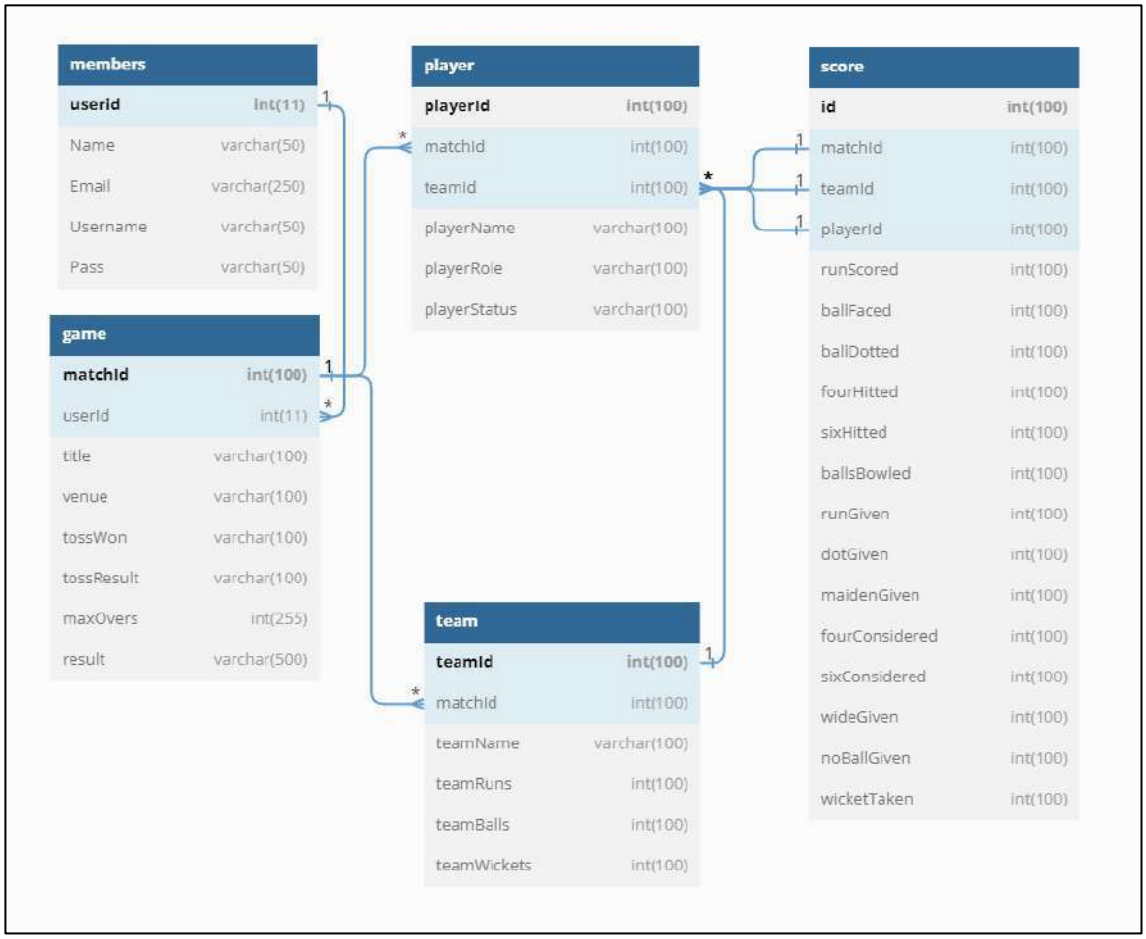


Fig. 3.14 Schema Diagram

3.5.3 Data Flow Diagram

Data flow diagrams are used to graphically represent the flow of data in a business information system. DFD describes the processes that are involved in a system to transfer data from the input to the file storage and reports generation. Data flow diagrams can be divided into logical and physical. The logical data flow diagram describes flow of data through a system to perform certain functionality of a business. The physical data flow diagram describes the implementation of the logical data flow.

Notations Reference: https://www.lucidchart.com/

Name	Symbol	Description
Process		A process transforms incoming data flow into outgoing data flow.
Database		Data stores are repositories of data in the system.
Data Flow		Data flows are pipelines through which packets of information flow. Label the arrows with the name of the data that moves through it.
External Entity		External entities are objects outside the system, with which the system communicates

Table 3.5 Data Flow Diagram Notations

Level 0 (Context Level DFD):

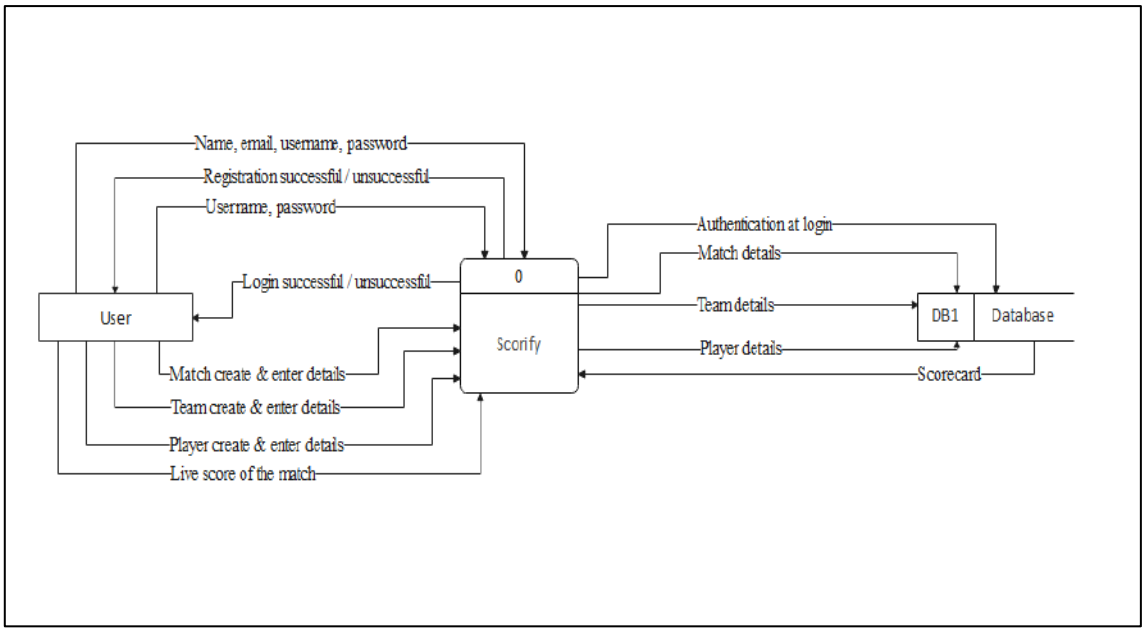


Fig.3.15 Level 0 DFD

Level 1 DFD:

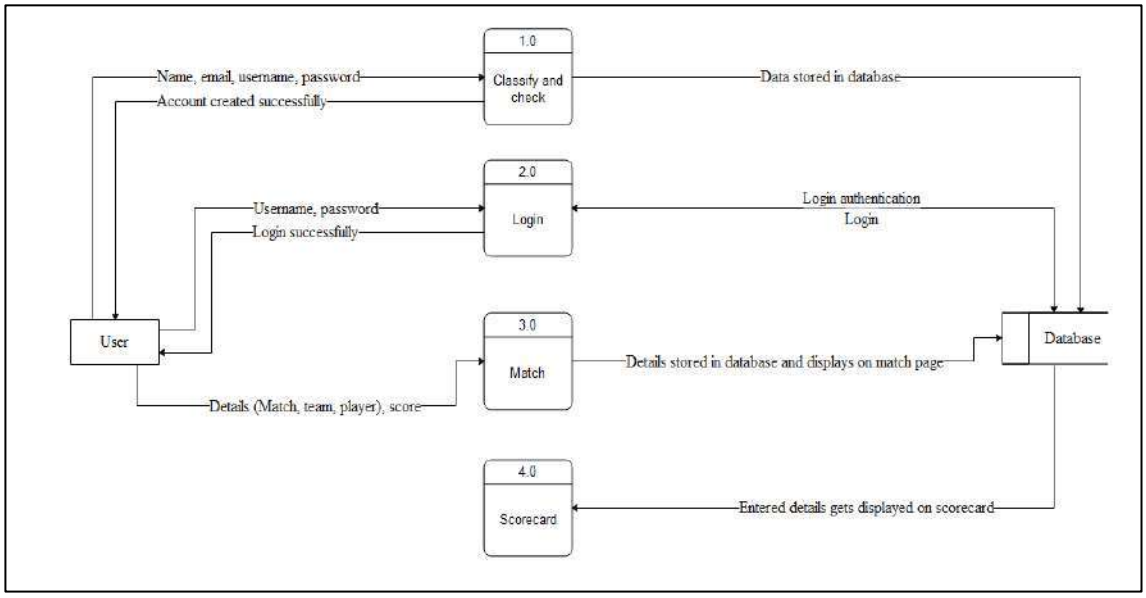


Fig. 3.16 Level 1 DFD

Level 2 DFD for Match:

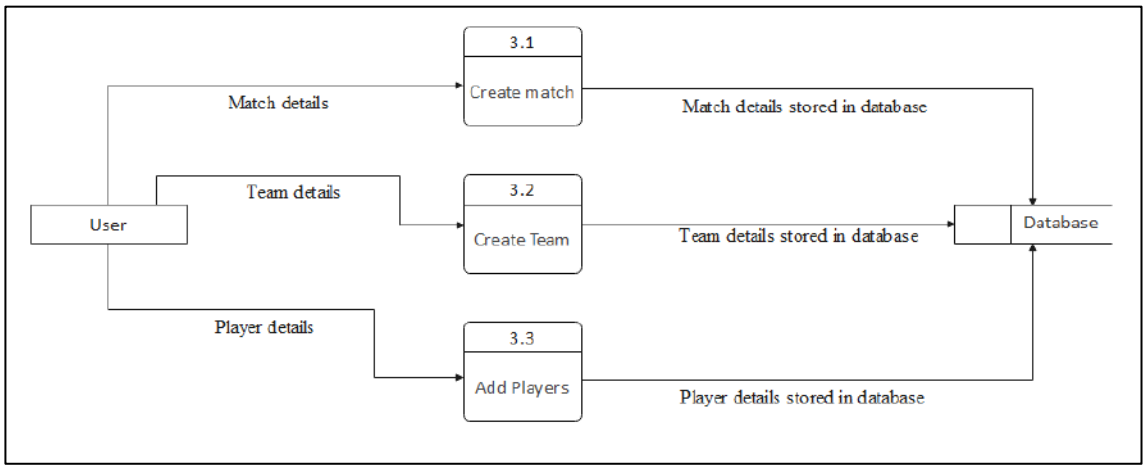


Fig. 3.17 Level 2 DFD for match

Level 2 DFD for Scorecard:

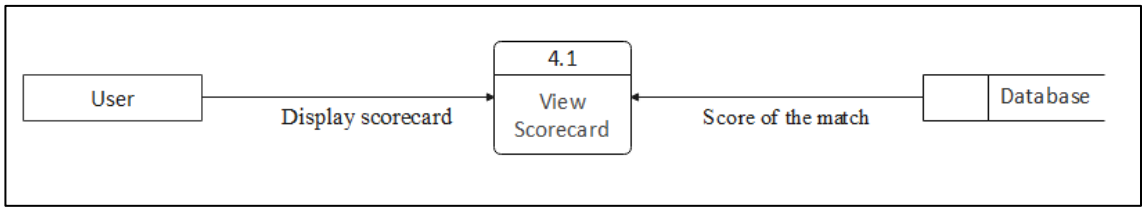


Fig. 3.18 Level 2 DFD for scorecard

3.5.4 Use Case Diagram

A use case diagram at its simplest is a representation of a user's interaction with the system that shows the relationship between the user and the different use cases in which the user is involved. A use case diagram can identify the different types of users of a system and the different use cases and will often be accompanied by other types of diagrams as well. The use cases are represented by either circles or ellipses.

Notations Reference: https://www.lucidchart.com/

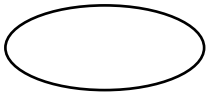
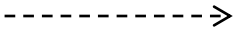
Name	Symbol	Description
Actor		Actor represents a user or another system that will interact with the system you are modelling.
Use Case		A use case is an external view of the system that represents some action the user might perform in order to complete a task.
Association		Association between use cases.
Include Relationship		Include relationship between the use cases

Table 3.6 Use Case Notation

3.5.4.1 Use Case Diagram:

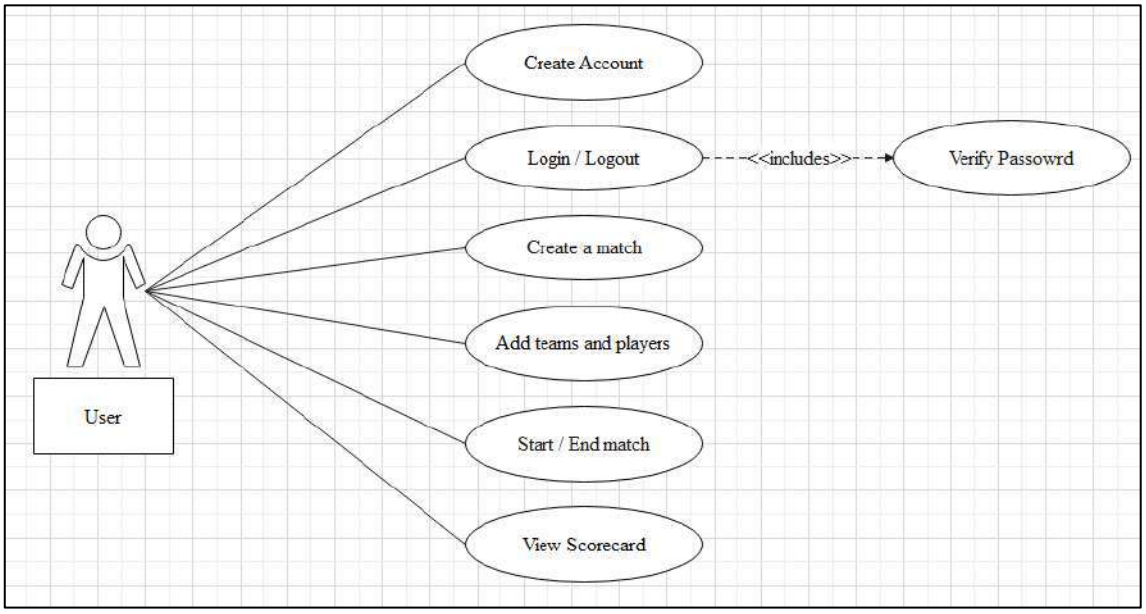


Fig. 3.19 Use Case Diagram

3.5.4.2 Use Case Diagram Description:

1. Use Case: Register
    a. Description: The user needs to register their account. They fill the necessary details and can access their account using it.
    b. Actor: User.
    c. Pre-condition: User needs to full all the details.
    d. Exception: If the format of any detail is incorrect or any required field is kept empty, registration will not be successful.
    e. Post-condition: Registered successfully. Username and password provided.
2. Use Case: Login / Logout
    a. Description: The user can sign in or sign out of their accounts.
    b. Actor: User.
    c. Pre-condition: User needs to fill the correct login details to sign-in into their account.
    d. Post-condition: -

3. Use Case: Create a match
    a. Description: The user can create a new match for scoring the live game.
    b. Actor: User.
    c. Pre-condition: User must be logged in to their account.
    d. Post-condition: User will be able to create team, add player for created match.
4. Use Case: Add team and players
    a. Description: The user can add teams and players in the teams for created match.
    b. Actor: User.
    c. Pre-condition: User must have created a match.
    d. Post-condition: -
5. Use Case: Start or end match
    a. Description: The user will be able to start and end the created match.
    b. Actor: User.
    c. Pre-condition: User must have created a match, added teams and players.
    d. Post-condition: -
6. Use Case: View Scorecard
    a. Description: The user will be able to view the scorecard of the match.
    b. Actor: User.
    c. Pre-condition: The match should have been ended.
    d. Post-condition: -

3.5.4 Sequence Diagram

A sequence diagram in a Unified Modelling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams typically are associated with use case realizations in the Logical View of the system under development.

Symbol reference: https://www.lucidchart.com/

Name	Symbol	Description
Synchronous Message		An instantaneous communication between objects that conveys information, with the expectation that an action will be initiated as a result.
Activation Box		The period during which an object is performing an action.
Object		An object that is created, performs actions, and/or is destroyed during the lifeline

Table 3.7 Sequence Diagram Notation

Sequence Diagrams:

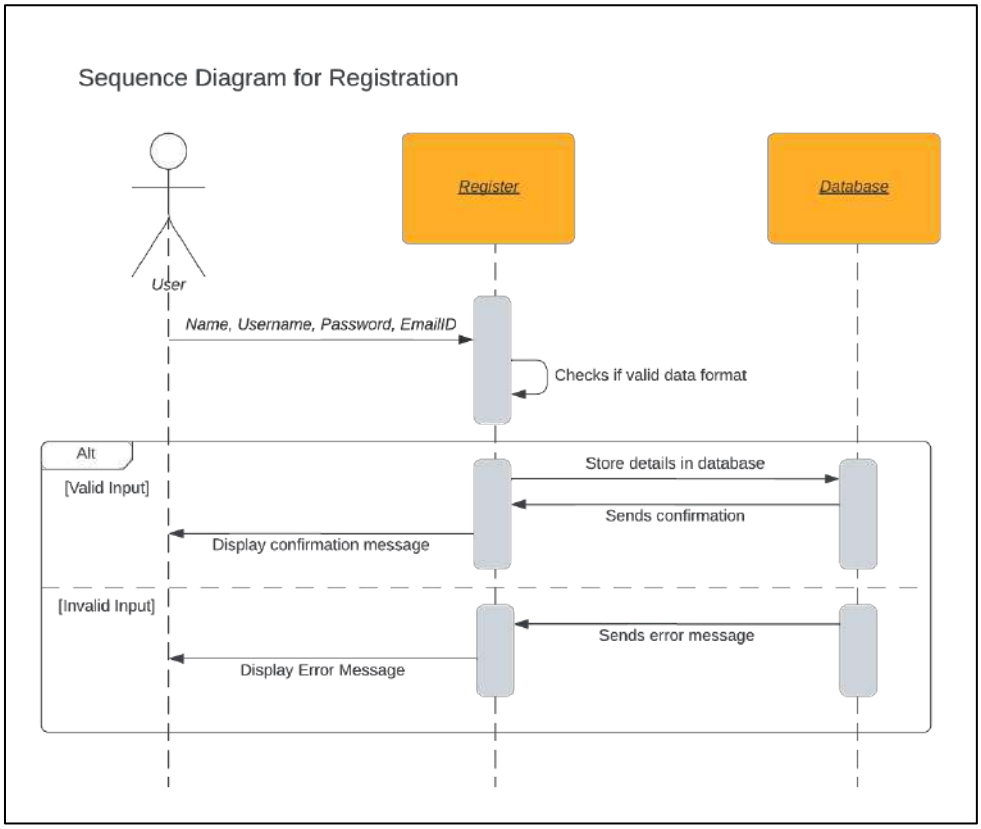


Fig. 3.20 Sequence Diagram for Registration

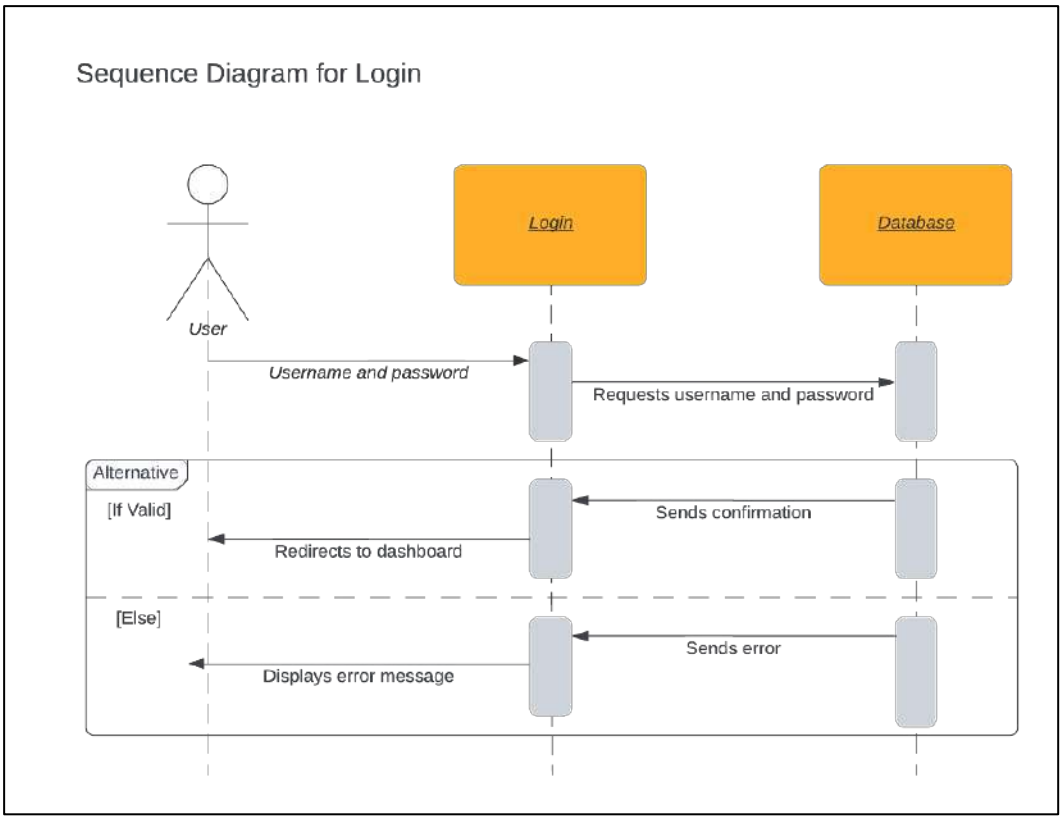


Fig. 3.21 Sequence Diagram for Login

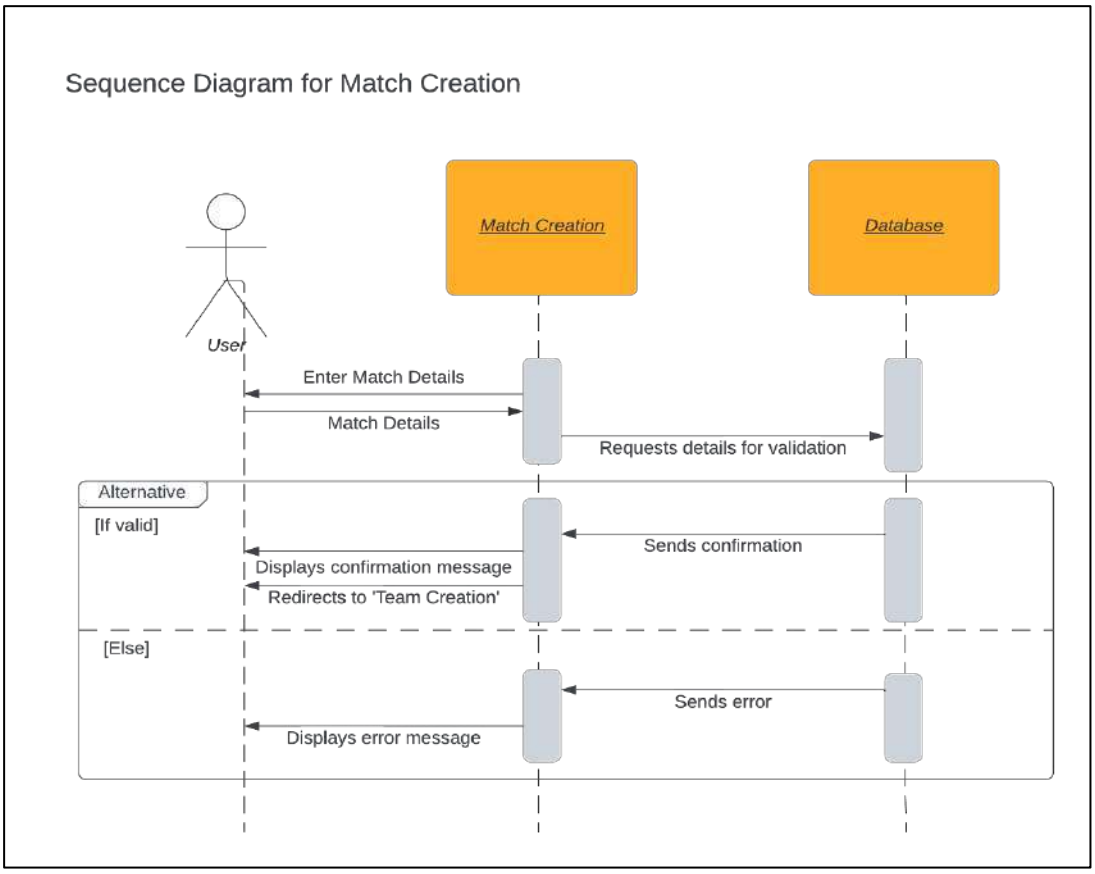


Fig. 3.22 Sequence Diagram for Match Creation

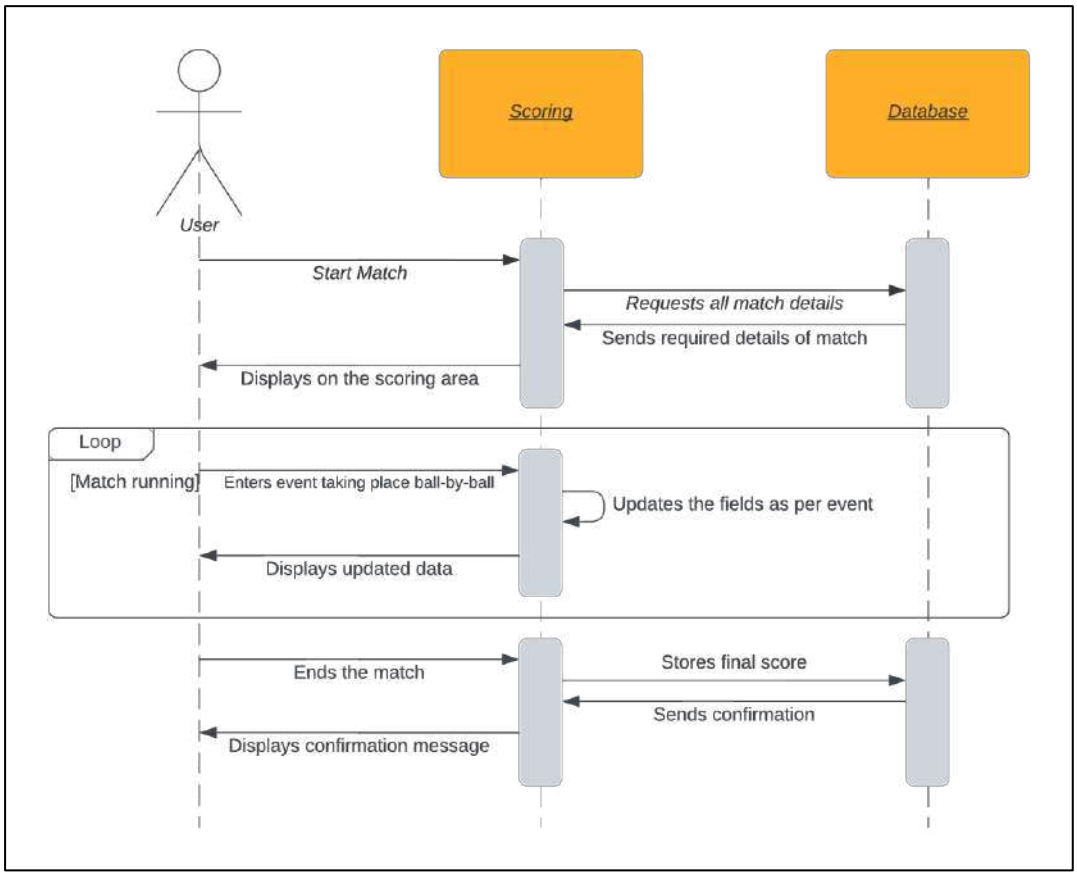


Fig 3.23 Sequence diagram for Live Scoring

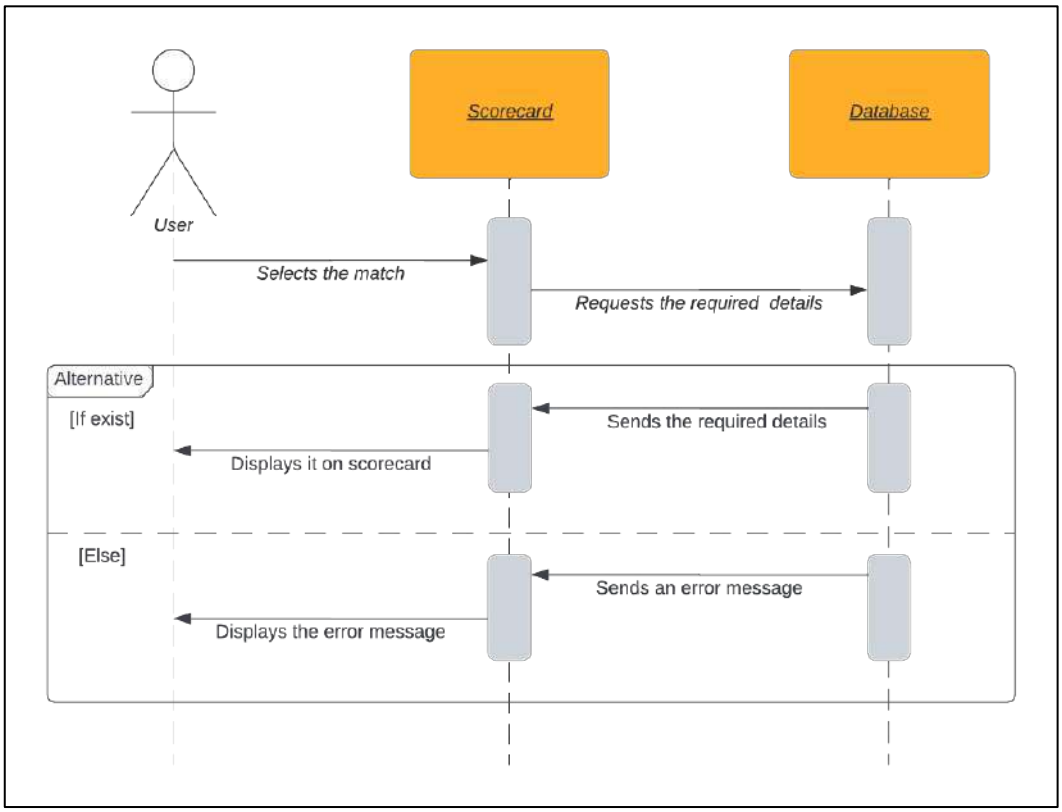


Fig. 3.24 Sequence diagram for scorecard

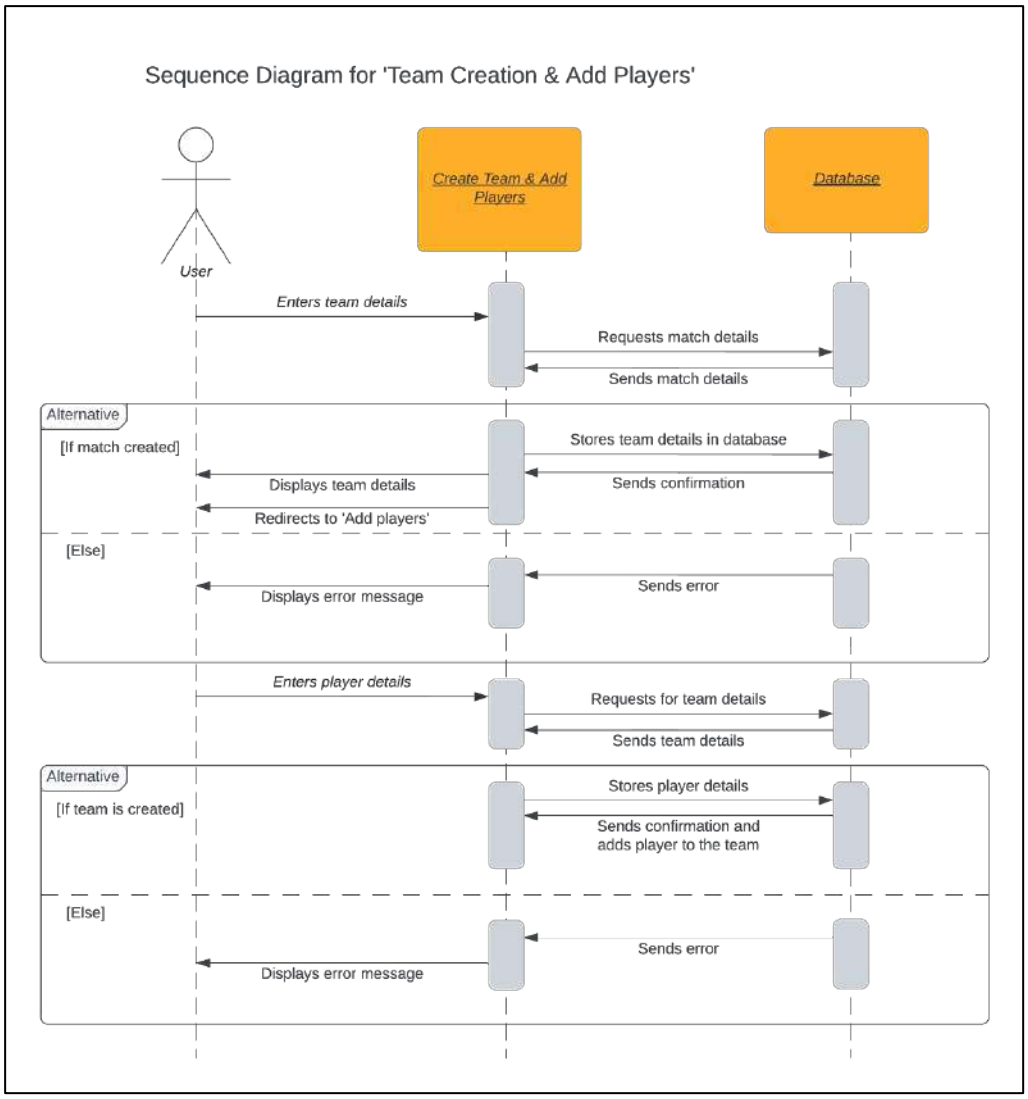


Fig. 3.25 Sequence diagram for Creation of teams and players

3.5.6 Activity Diagram

- Activity diagram is another important diagram in UML to describe the dynamic aspects of the system.
- Activity diagram is basically a flowchart to represent the flow from one activity to another activity.
- The activity can be described as an operation of the system. The control flow is drawn from one operation to another.
- This flow can be sequential, branched, or concurrent. Activity diagrams deal with all type of flow control by using different elements such as fork, join, etc.

Symbol reference: https://www.lucidchart.com/

Name	Symbol	Description
Initial State		This shows the starting point or first activity of the flow.
Final State		The end of the Activity diagram, also called as a final activity.
Action		It represents the activity to be performed.
Decision		A logic where a decision is to be made is depicted by a diamond.
Transition		A transition link represents control flow between nodes.

Table 3.8 Activity Diagram Notations

Diagram:

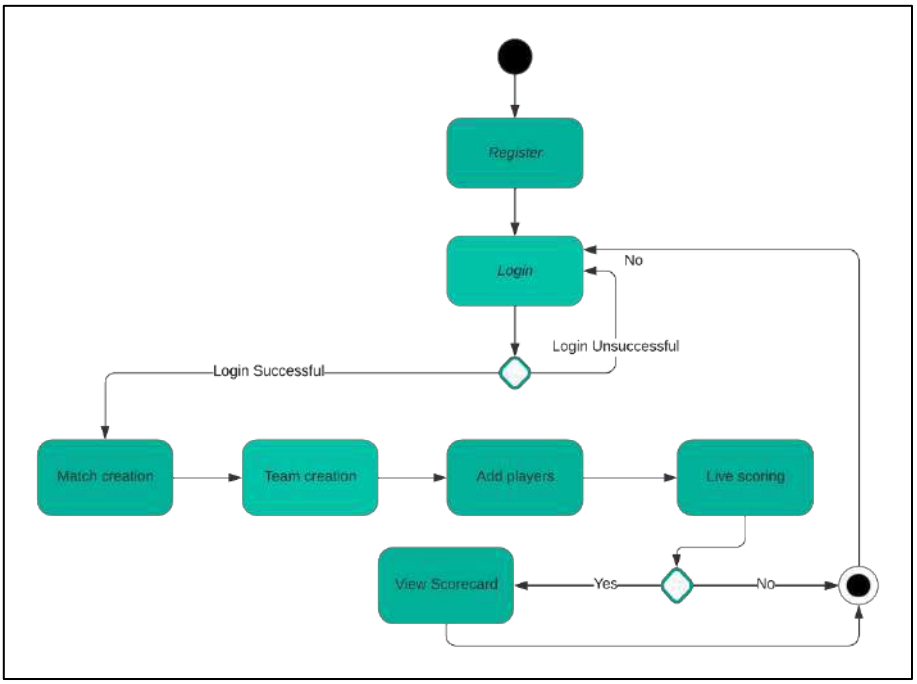


Fig 3.26 Activity Diagram

3.5.7 State-Chart Diagram

A state diagram is used to represent the condition of the system or part of the system at finite instances of time. It's a behavioural diagram and it represents the behaviour using finite state transitions. State diagrams are also referred to as State machines and State-chart Diagrams. These terms are often used interchangeably. So simply, a state diagram is used to model the dynamic behaviour of a class in response to time and changing external stimuli.

Symbol reference: https://www.lucidchart.com/

Name	Symbol	Reference
Initial State		This represents the starting of the state diagram.
Final State		This represents the final state or end of the state diagram.
Transition		This represents the change of one state into another state.
State		This represents the state of the activity.

Table 3.9 State-Chart Notations

Diagram:

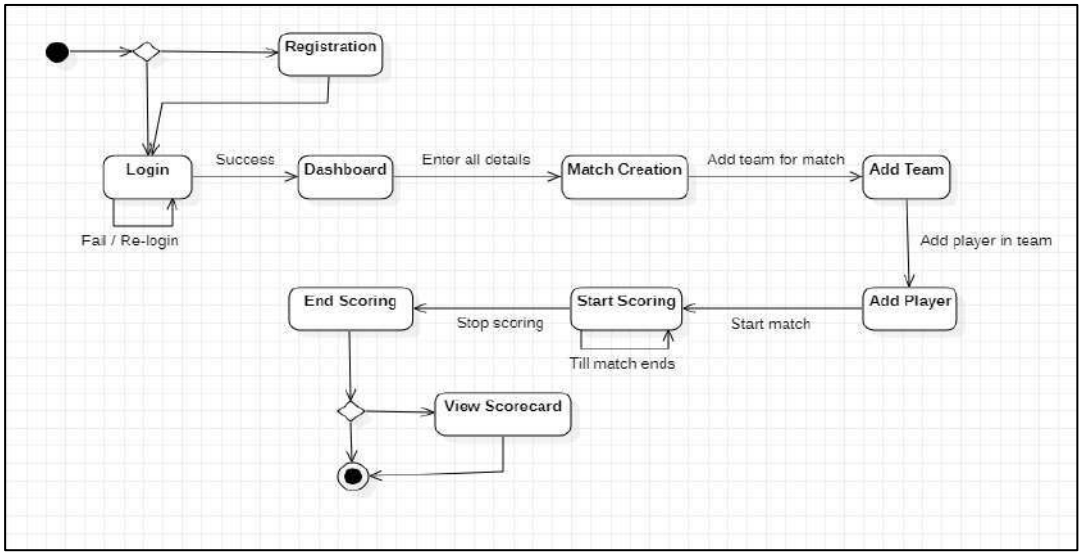


Fig. 3.27 State-Chart Diagram

Chapter 4: System Design

4.1 User Interface

1. Home Page

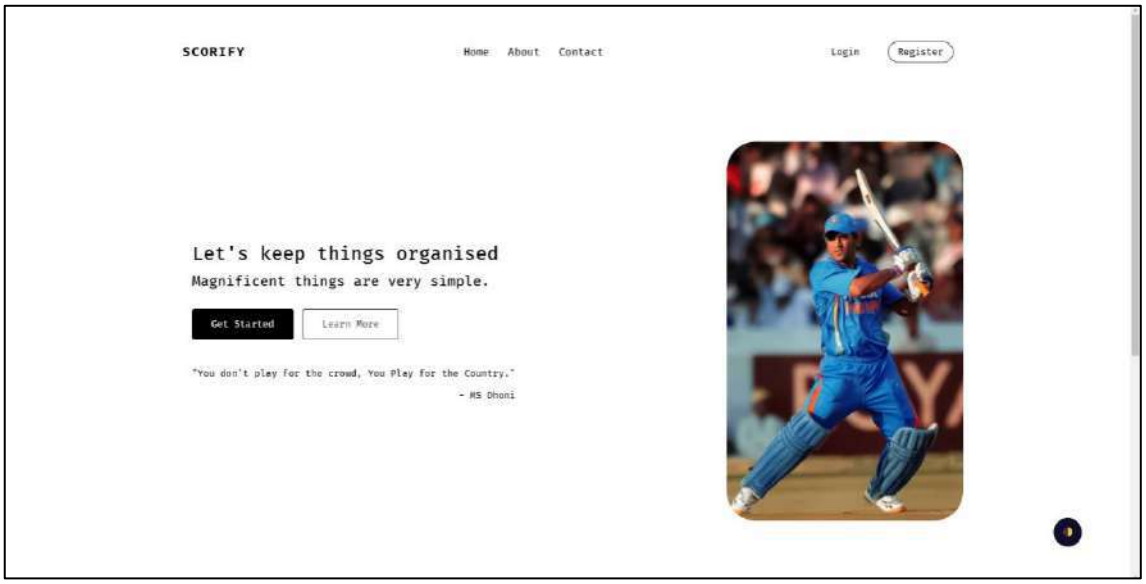


Fig. 4.1 UI for Home Page

2. Registration Page

Fig. 4.2 UI for Registration Page

3. Login Page

Fig. 4.3 UI for Login Page

4. Dashboard

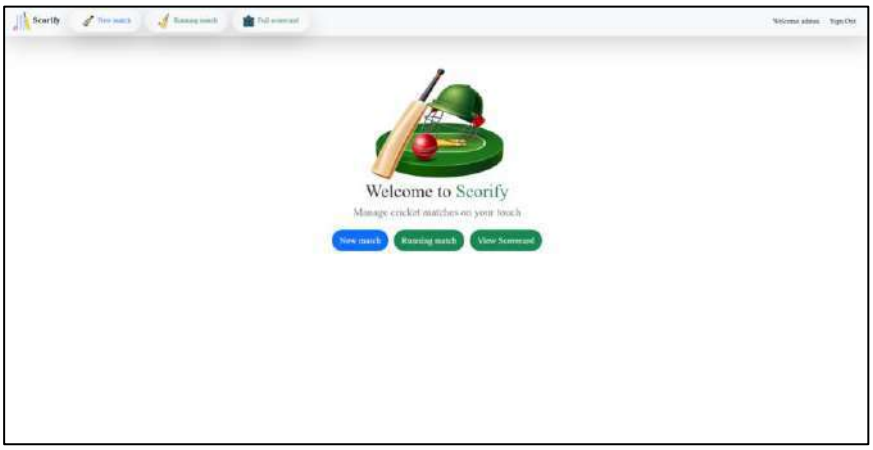


Fig. 4.4 UI for Dashboard

5. Creating Match

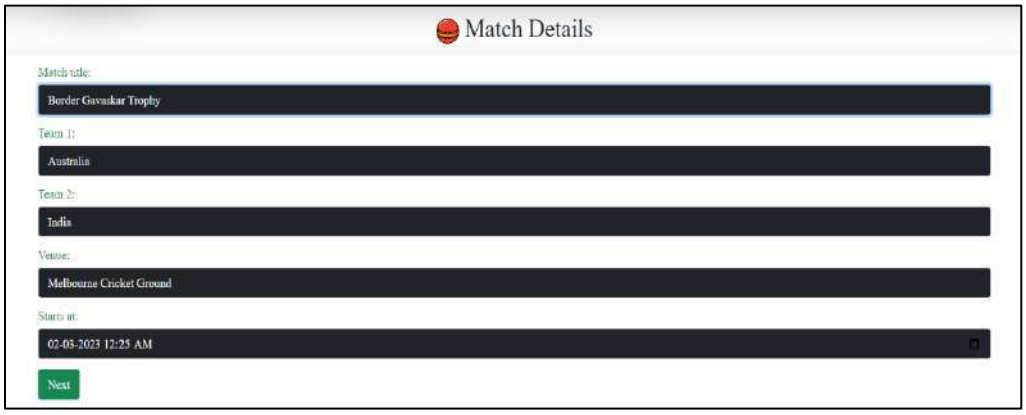


Fig. 4.5 UI For Match Creation

6. Toss Details

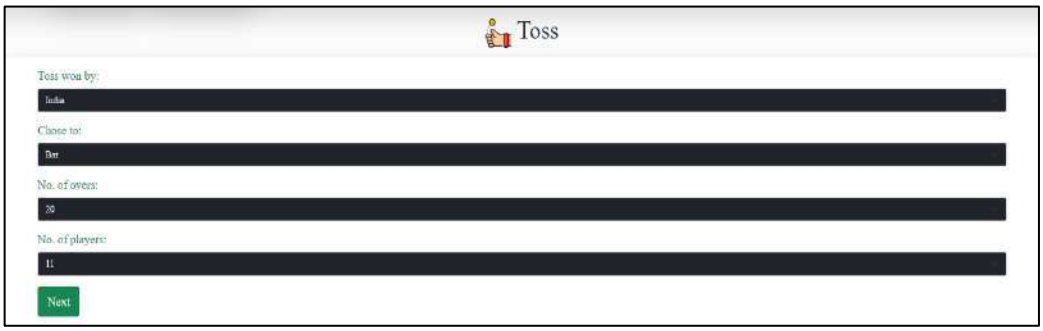


Fig. 4.6 UI For Toss Details

7. Add Player

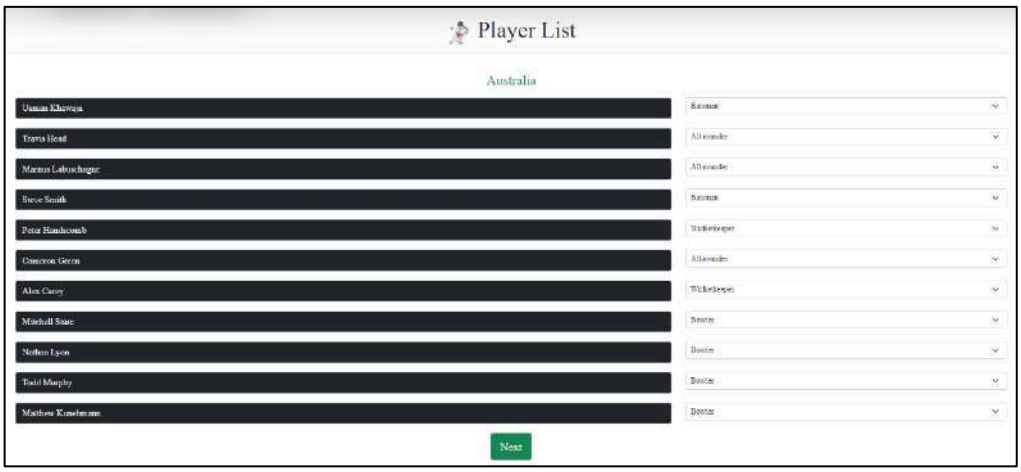


Fig. 4.7 UI For Team-wise Player List

8. Live Scoring

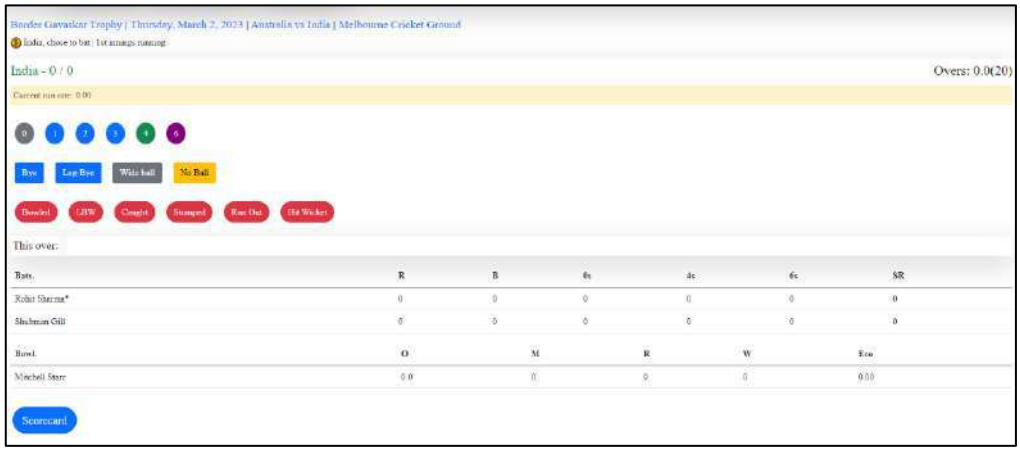


Fig. 4.8 UI for Live Scoring

9. Scorecard

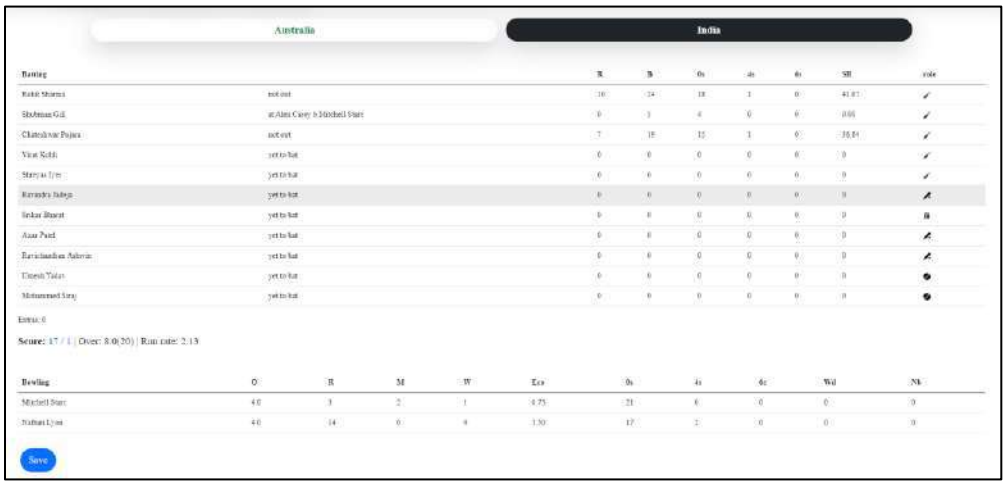


Fig. 4.9 UI for Scorecard

4.2 Test Cases

Test Case No.	Test case Description	Test Case	Expected Output	Actual Output	Remark
1.	Register for user	Name: Pushkar Email id: pushkar@gmail.com Create password: pushkar@ Re-enter password: pushkar@	User has been registered successfully.	User has been registered successfully.	Pass
2.	Register for user	Name: Pushkar Email id: Pushkar.gmail.com Create password: pushkar@ Re-enter password: pushkar@	Please enter valid emailid	Please enter valid email id	Pass

3.	Register for user	Name: Pushkar Email id: pushkar@gmail.com Create password: pushkar@ Re-enter password: pushkar	Passwords don't match	Passwords don't match	Pass
4	Login for user	Username: Pushkar Password: pushkar	Please enter correct password!	Please enter correct password!	Pass
5.	Login for user	Username: pushkar Password: pushkar@	Please enter correct username!	Please enter correct username!	Pass
6.	Login for user	Username: Pushkar Password: pushkar@	Redirects user to the dashboard.	Redirects user to the dashboard.	Pass
7.	Create Match	Click on the button	User should get redirect to team creation page	User get redirect to team creation page	Pass
8.	Team Creation	Team-A Name: India Team-B Name: Australia Venue: Hyderabad	User should get redirect to add player page	User get redirect to add player page	Pass
9.	Team Creation	Team-A Name: India Team-B Name:	Team name cannot be blank	Team name cannot be blank	Pass
10.	Team Creation	Team-A Name: India Team-B Name: India Venue: Hyderabad	Names of team cannot be same.	Names of team cannot be same.	Pass

11	Start Match	Click on the button	User should get redirect to Scoring page and display entered details.	User get redirect to Scoring page and display entered details.	Pass
12.	Scoring	Click on wide	Should add 1 run in batting team and +1 in extras.	Adds 1 run in batting team and adds 1 ball	Fail
13.	Scoring	Click on no-ball	Should add 1 run in batting team and +1 in extras.	Add 1 run in batting team and +1 in extras.	Pass
14.	Scoring	Click on 1-run	Add 1 run in striker runs and 1 run in bowler's run.	Add 1 run in striker runs and 1 run in bowler's run.	Pass
15.	Scoring	Click on 4-runs	Add 4 runs in striker runs, 4 runs in bowler's runs and +1 in batsman's 4's column.	Add 4 runs in striker runs, 4 runs in bowler's runs	Fail
16.	Scoring	Click on 0	Add 1 ball in batsman and bowler.	Add 1 ball in batsman and bowler.	Pass
17.	Scoring	Click on Wicket	End inning of batsman, add 1 wicket in batting team	End inning of batsman, add 1 wicket in batting team.	Fail

			as well as bowler.		
18.	Scoring	Selects new batsman	Display on scorecard.	Displays batsman on scorecard	Pass
19.	Scoring	Selects new bowler	Display on scorecard.	Display bowler on scorecard.	Pass
20.	Match	Selects same batsman on strike and non-strike	Both batsmen cannot be same	Redirects to scoring page	Fail
21.	Scoring	Click on Bowled	Add 1 wicket to batting team and display prompt.	Prompt displayed but no wicket was added to the batting team	Fail
22.	Scoring	Click on Stumped	Add wicket, add bowl played in batsman profile.	No wicket was added. No bowl was added.	Fail
23.	Scoring	Click on 0	Add 1 ball in batsman and bowler.	Add 1 ball in batsman and bowler.	Pass
24.	Home	Click on sign-out	User should get signed-out of their account.	User should log-out of their account and redirect to home page	Pass

Table 4.1 Test Cases

Chapter 5: Implementation and Testing

5.1 Implementation Approaches

The Live Cricket Scoring System (Scorify) project was created utilising Extreme Programming Concepts (XP), which is intended to increase software quality and responsiveness to client needs. The extreme programming approach suggests scaling up the best methods that have previously performed successfully in programme development initiatives to extreme levels. Extreme Programming (XP) is a software development process that emphasises high-quality product delivery through frequent and continuous feedback, collaboration, and adaptability. With an emphasis on rapid, iterative development and deployment, XP promotes a close working relationship between the development team, the client, and stakeholders. If the user requirements change at any time, the appropriate component can be rebuilt, reimplemented, and tested again.

The interfaces are designed and created using Visual Studio Code. After the user interfaces were created, database connectivity was performed. I connected my system to the SQL Server at the free web hosting site "infinityhost.com". The coding part of my project is done in PHP language. The project was divided into modules. These modules were created one by one and after completion of each module, unit testing was performed on that module. When the module fulfils its requirements, it was integrated into the main project. After integration, each functionality was checked which can also be said to be as integration testing. After adding all the modules to my main project, finally system testing was performed to check whether the system is working accordingly or not.

5.2 Coding and Efficiency

5.2.1 Coding:

Login.php

```
<?php
ob_start();
include("../config/connect.php");
$status = get_con();
session_start();
$status = session_status(); //1st measure
if ($status == PHP_SESSION_ACTIVE) {
```

```
session_destroy();//There is active session
}
// if session is already running, it destroys previous session and starts a new if redirected
to this page
session_start();
if (isset($_POST['Login'])) {
  $username =  $_POST['username'];
  $password = $_POST['password'];
  $con = get_con();
  $sql = "SELECT * FROM `user` WHERE Username = '$username' AND Pass =
'$password';";
  $result = mysqli_query($con, $sql);
  $result_user_type = mysqli_fetch_array($result);
  $row = mysqli_num_rows($result);
  if ($row > 0) {
    header("Location:./live/dashboard.php");
    //session set
    $_SESSION['name'] = $result_user_type['Username'];
    $_SESSION['id'] = $result_user_type['userid'];
  }
else{
    echo"<script>alert('Invalid username or password.');</script>";
  }
mysqli_close($con); // close connection
}
  // login block ends here forcheching echo $status;
ob_end_flush();
?>
```

Php code for setting session

```
<?php
ob_start();
include("../../config/connect.php");
$status = get_con();
```

```
session_start();
if (!isset($_SESSION['name'])) {
  // redirect if not set
  header("Location:../login.php");
}
$login_session = $_SESSION['name'];
ob_end_flush(); ?>
```

Route.js

```
let view = (url, fun, params) => {
    const xhr = new XMLHttpRequest();
    xhr.open("GET", url);
    xhr.send();
    xhr.addEventListener("readystatechange", () => {
        if (xhr.readyState == 4) {
            document.querySelector("#main-container").innerHTML = xhr.responseText;
            fun(params);
        }
    });
};
let loadHome = () => {
    view("home.php", () => {
        let match = JSON.parse(localStorage.getItem("match"));
        if (match && match.title) {
            document.querySelector("#running-match-nav").classList.remove("d-none");
        }
        if (match && match.onStrikeBatsman) {
            document.querySelector("#score-nav").classList.remove("d-none");
            document.querySelector("#home-rm").classList.remove("d-none");
        }
    });
};
let loadScoreCard = () => {
    view("viewcard.php", () => {
```

```
$("#match").on('change', (e) => {
    console.log(e);
    let match = $("#match").val();
    console.log(match);
    if(match == "--"){
        $("#teamOneName").hide();
        $("#teamTwoName").hide();
        $("#battingCard1").hide();
        $("#battingCard2").hide();
        $("#bowlingCard1").hide();
        $("#bowlingCard2").hide();
    }
    else{
        $("#teamOneName").show();
        $("#teamTwoName").show();
        $("#battingCard1").show();
        $("#battingCard2").show();
        $("#bowlingCard1").show();
        $("#bowlingCard2").show();
    //Team-1 Name
    $.ajax({
        url : "../ajax/db_ajaxcalls.php",
        type : "POST",
        data : {
            team1Name : "YES",
            matchId : match,
        },
        success: function (res) {
            $("#teamOneName").html(res);
        },
        error : function (err) {
            console.log(err);
        }
    })
```

```
//Team-2 Name
$.ajax({
    url : "../ajax/db_ajaxcalls.php",
    type : "POST",
    data : {
        team2Name : "YES",
        matchId : match,
    },
    success: function (res) {
        $("#teamTwoName").html(res);
    },
    error : function (err) {
        console.log(err);
    }
})
//Team-1 Batting
$.ajax({
    url : "../ajax/db_ajaxcalls.php",
    type : "POST",
    data : {
        // team1PlayerNames : "YES",
        matchId : match,
        team1batting : "YES"
    },
    success: function (res) {
        $("#battingCard1").html(res);
    },
    error : function (err) {
        console.log(err);
    }
})
//Team-2 Batting
$.ajax({
    url : "../ajax/db_ajaxcalls.php",
```

```
    type : "POST",
    data : {
        // team2PlayerNames : "YES",
        matchId : match,
        team2batting : "YES"
    },
    success: function (res) {
        //console.log(res);
        $("#battingCard2").html(res);
    },
    error : function (err) {
        console.log(err);
    }
})
//Team-1 Bowling
$.ajax({
    url : "../ajax/db_ajaxcalls.php",
    type : "POST",
    data : {
        // team1PlayerNames : "YES",
        matchId : match,
        team1bowling : "YES"
    },
    success: function (res) {
        //console.log(res);
        $("#bowlingCard1").html(res);
    },
    error : function (err) {
        console.log(err);
    }
})
//Team-2 Bowling
$.ajax({
    url : "../ajax/db_ajaxcalls.php",
```

```
            type : "POST",
            data : {
                // team2PlayerNames : "YES",
                matchId : match,
                team2bowling : "YES",
            },
            success: function (res) {
                //console.log(res);
                $("#bowlingCard2").html(res);
            },
            error : function (err) {
                console.log(err);
            }
        })
    }
    })
  })
}
let loadScoreBoardAdditional = () => {
  let match = JSON.parse(localStorage.getItem("match"));
  let options = {
    weekday: "long",
    year: "numeric",
    month: "long",
    day: "numeric",
  };
  let date_time = new Date(match.startTime);
  date_time = date_time.toLocaleDateString("en-US", options);
  document.querySelector(
    "#toss-win"
  ).innerHTML = `${match.tossWonBy}, chose to ${match.tossDecision} | `;
  let ii = match.runningInnings == 0 ? "1st": "2nd";
  document.querySelector(
    "#innings-indicator"
```

```
    ).innerHTML = `${ii} innings running`;
    document.querySelector(
        "#match-heading"
        ).innerHTML = `${match.title} <span class="text-dark fw-bold">|</span> ${date_time} <span class="text-dark fw-bold">|</span> ${match.teams[0]} vs ${match.teams[1]} <span class="text-dark fw-bold">|</span> ${match.venue}`;
    loadScore();
    if (
        !match.verdict ||
        (match.verdict &&
            !match.verdict.includes("won") &&
            !match.verdict.includes("tied"))
    ) {
        for (yy of document.querySelectorAll(".score-counter")) {
            yy.classList.remove("d-none");
        }
    }
};
let runningMatch = () => {
    if (localStorage.getItem("match") === null) {
        view("details.php", () => {});
    } else {
        match = JSON.parse(localStorage.getItem("match"));
        if (match.onStrikeBatsman) {
            view("play.php", loadScoreBoardAdditional);
        } else if (match.teamLineUp && match.teamLineUp[1].length > 0) {
            view("openers.php", setDomOpeners);
        } else if (match.teamLineUp && match.teamLineUp[0].length > 0) {
            view("lineup_1.php", setDomLineUp, 1);
        } else if (match.tossWonBy) {
            view("lineup_0.php", setDomLineUp, 0);
        } else if (match.title) {
            view("toss.php", setDomToss);
        }
```

```
    }
};
let teamFullCard = (track) => {
    view("scorecard.php", loadFullScorecard, track);
};
window.addEventListener("load", () => {
    loadHome();
});
```

Scoreboard.js (Ajax Request)

```
var player1_names = [];
    var player1_role = [];
    var player1_status = [];
    var player1_runsScored = [];
    var player1_ballfaced = [];
    var player1_ballDotted = [];
    var player1_fourHitted = [];
    var player1_sixHitted = [];
    var player1_ballsBowled = [];
    var player1_runsGiven = [];
    var player1_dotGiven = [];
    var player1_maidenGiven = [];
    var player1_fourConsidered = [];
    var player1_sixConsidered = [];
    var player1_wideGiven = [];
    var player1_noBallGiven = [];
    var player1_wicketTaken = [];

    var player2_names = [];
    var player2_role = [];
    var player2_status = [];
    var player2_runsScored = [];
    var player2_ballfaced = [];
    var player2_ballDotted = [];
```

```
var player2_fourHitted = [];
var player2_sixHitted = [];
var player2_ballsBowled = [];
var player2_runsGiven = [];
var player2_dotGiven = [];
var player2_maidenGiven = [];
var player2_fourConsidered = [];
var player2_sixConsidered = [];
var player2_wideGiven = [];
var player2_noBallGiven = [];
var player2_wicketTaken = [];
//Hidden Fields Data Bridge
$("#savedata").click(() => {
    let match = JSON.parse(localStorage.getItem("match"));
    let realmatch = match;
    console.log(realmatch);
    var team1_arr = realmatch.teamLineUp[0];
    var team2_arr = realmatch.teamLineUp[1];
    for(let i = 0; i < Number(match.noOfPlayers); i++){
        player1_names.push((team1_arr[i].name));
        player1_role.push((team1_arr[i].role));
        player1_status.push((team1_arr[i].status));
        player1_runsScored.push((team1_arr[i].runScored));
        player1_ballfaced.push((team1_arr[i].ballFaced));
        player1_ballDotted.push((team1_arr[i].ballDotted));
        player1_fourHitted.push((team1_arr[i].fourHitted));
        player1_sixHitted.push((team1_arr[i].sixHitted));
        player1_ballsBowled.push((team1_arr[i].ballBowled / 6));
        player1_runsGiven.push((team1_arr[i].runGiven));
        player1_dotGiven.push((team1_arr[i].dotGiven));
        player1_maidenGiven.push((team1_arr[i].maidenGiven));
        player1_fourConsidered.push((team1_arr[i].fourConsidered));
        player1_sixConsidered.push((team1_arr[i].sixConsidered));
        player1_wideGiven.push((team1_arr[i].wideGiven));
```

```
        player1_noBallGiven.push((team1_arr[i].noBallGiven));
        player1_wicketTaken.push((team1_arr[i].wicketTaken));
    }
     tempteam1 = [player1_names, player1_role, player1_status, player1_runsScored,
player1_ballfaced, player1_ballDotted, player1_fourHitted, player1_sixHitted,
player1_ballsBowled, player1_runsGiven, player1_dotGiven, player1_maidenGiven,
player1_fourConsidered, player1_sixConsidered, player1_wideGiven,
player1_noBallGiven, player1_wicketTaken];
    console.log(tempteam1);
    for(let i = 0; i < Number(match.noOfPlayers); i++){
        player2_names.push((team2_arr[i].name));
        player2_role.push((team2_arr[i].role));
        player2_status.push((team2_arr[i].status));
        player2_runsScored.push((team2_arr[i].runScored));
        player2_ballfaced.push((team2_arr[i].ballFaced));
        player2_ballDotted.push((team2_arr[i].ballDotted));
        player2_fourHitted.push((team2_arr[i].fourHitted));
        player2_sixHitted.push((team2_arr[i].sixHitted));
        player2_ballsBowled.push((team2_arr[i].ballBowled / 6));
        player2_runsGiven.push((team2_arr[i].runGiven));
        player2_dotGiven.push((team2_arr[i].dotGiven));
        player2_maidenGiven.push((team2_arr[i].maidenGiven));
        player2_fourConsidered.push((team2_arr[i].fourConsidered));
        player2_sixConsidered.push((team2_arr[i].sixConsidered));
        player2_wideGiven.push((team2_arr[i].wideGiven));
        player2_noBallGiven.push((team2_arr[i].noBallGiven));
        player2_wicketTaken.push((team2_arr[i].wicketTaken));
    }
    $.ajax({
        url : "../ajax/db_ajaxcalls.php",
        type : "POST",
        data : {
            flag : "YES",
            //Match Table
```

```
noOfPlayers : match.noOfPlayers,
title : match.title,
venue : match.venue,
result : match.verdict,
tossWon : match.tossWonBy,
tossResult : match.tossDecision,
maxOvers : match.noOfOvers,
//Team Table
teamOneName : match.teams[0],
teamTwoName : match.teams[1],
teamOneRuns : match.teamScoreboard[0].totalRunScored,
teamTwoRuns : match.teamScoreboard[1].totalRunScored,
teamOneBalls : match.teamScoreboard[0].ballsPlayed,
teamTwoBalls : match.teamScoreboard[1].ballsPlayed,
teamOneWickets : match.teamScoreboard[0].wicketFall,
teamTwoWickets : match.teamScoreboard[1].wicketFall,
//Score Table
team1: [player1_names, player1_role, player1_status, player1_runsScored, player1_ballfaced, player1_ballDotted, player1_fourHitted, player1_sixHitted, player1_ballsBowled, player1_runsGiven, player1_dotGiven, player1_maidenGiven, player1_fourConsidered, player1_sixConsidered, player1_wideGiven, player1_noBallGiven, player1_wicketTaken],
team2: [player2_names, player2_role, player2_status, player2_runsScored, player2_ballfaced, player2_ballDotted, player2_fourHitted, player2_sixHitted, player2_ballsBowled, player2_runsGiven, player2_dotGiven, player2_maidenGiven, player2_fourConsidered, player2_sixConsidered, player2_wideGiven, player2_noBallGiven, player2_wicketTaken]
},
success: function (res) {
console.log(res);
},
error : function (err) {
console.log(err);
}
```

```
});
});
};
```

Scorecard.php

```
<?php
ob_start();
include("../../config/connect.php");
$status = get_con();
session_start();
if (!isset($_SESSION['name'])) {
  header("Location:../login.php");  // redirect if not set
}
$userId = $_SESSION['id'];;
ob_end_flush();
?>
<br>
<label class="form-label text-success h5" for="match">Choose a match:</label>
        <select class="form-select bg-dark text-white" name="match" id="match">
        <?php
                //get ref of user for match
                $query = "SELECT * FROM `game` WHERE `userId` = '$userId'";
                $result = mysqli_query($status, $query);
                echo "<option value=" . "--" . ">" . "--" . "</option>";
                while ($row3 = $result ->fetch_assoc()) {
                        echo "<option value=" . $row3["matchId"] . ">" . $row3["title"] .
"</option>";
                }
 ?>
</select>
```

5.2.2 Coding Efficiency

I have tried to keep the codes as short as possible but functionalities and reliability aren't compromised. Efficiency is an important aspect of the system as the usability by reducing the complexity. Wherever there was repetition of code, I used functions. So, the functions were the called instead of writing the whole code again and again. Also, I have tried to implement Ajax in part of system so that the response was quick. I have also used local storage for main module of the system i.e. Live Scoring by which even if there is internet connectivity issue or by mistake the page reloads the data won't be lost as it will be stored in local storage and in the end the user has an option to save the data if they wish to view it in future.

5.3 Testing Approaches

The Testing Approach for the project was solely based on the reliability of the components implemented through Several Testing Phases to ensure the quality of the system is up to the requirements specified.

Here I have used manual testing technique. Manual testing is a type of software testing in which testers manually execute test cases without the use of automation tools. This involves a human tester performing a set of predefined steps and observations to evaluate the functionality, usability, and performance of a software application. Manual testing is often used in the early stages of the development cycle and is an effective way to identify issues and defects that may have been missed during automated testing or development. It can also provide valuable feedback on the user experience and overall quality of the application.

Also, I made a prototype and gave it to my school to test it in a real-life cricket match and testing was done accordingly.

5.3.1 Unit Testing

Unit testing is a type of software testing in which individual units or components of a software application are tested in isolation from the rest of the system to ensure they are functioning as intended. This involves writing and executing test cases for each unit of code to validate that it meets its specifications and produces the expected output. It is an important practice for ensuring the reliability, maintainability, and overall quality of software applications.

Test Cases:

Test Case No.	Test case Description	Test Case	Expected Output	Actual Output	Remark
1.	Register for user	Name: Pushkar Email id: pushkar@gmail.com Create password: pushkar@ Re-enter password: pushkar@	User has been registered successfully.	User has been registered successfully.	Pass
2.	Register for user	Name: Pushkar Email id: Pushkar.gmail.com Create password: pushkar@ Re-enter password: pushkar@	Please enter valid email id	Please enter valid email id	Pass
3.	Register for user	Name: Pushkar Email id: pushkar@gmail.com Create password: pushkar@ Re-enter password: pushkar	Passwords don't match	Passwords don't match	Pass

Table 5.1 User Registration

4	Login for user	Username: Pushkar Password: pushkar	Please enter correct password!	Please enter correct password!	Pass
5.	Login for user	Username: pushkar Password: pushkar@	Please enter correct username!	Please enter correct username!	Pass

6.	Login for user	Username: Pushkar Password: pushkar@	Redirects user to the dashboard.	Redirects user to the dashboard.	Pass

Table 5.2 User Login

7.	Create Match	Click on the button	User should get redirect to team creation page	User get redirect to team creation page	Pass
8.	Team Creation	Team-A Name: India Team-B Name: Australia Venue: Hyderabad	User should get redirect to add player page	User get redirect to add player page	Pass
9.	Team Creation	Team-A Name: India Team-B Name:	Team name cannot be blank	Team name cannot be blank	Pass
10.	Team Creation	Team-A Name: India Team-B Name: India Venue: Hyderabad	Names of team cannot be same.	Names of team cannot be same.	Pass

Table 5.3 Match and Team Details

11.	Start Match	Batsman 1: Rohit Sharma Batsman 2: Rohit Sharma Bowler: Mitchell Starc	Both batters cannot be same.	Both batters cannot be same.	Pass
12.	Start Match	Batsman 1: Rohit Sharma Batsman 2: Virat Kohli Bowler: Mitchell Starc	Redirect to live scoring	Redirect to live scoring	Pass
13.	Start Match	Click on no-ball	Should add 1 run in batting	Add 1 run in batting team	Pass

			team and +1 in extras.	and +1 in extras.	

Table 5.4 Player Details

14.	Start Match	Click on the button	User should get redirect to Scoring page and display entered details.	User get redirect to Scoring page and display entered details.	Pass
15.	Scoring	Click on wide	Should add 1 run in batting team and +1 in extras.	add 1 run in batting team and +1 in extras and Strike rotate	Fail
16.	Scoring	Click on no-ball	Should add 1 run in batting team and +1 in extras.	Add 1 run in batting team and +1 in extras.	Pass
17.	Scoring	Click on 1-run	Add 1 run in striker runs and 1 run in bowler's run.	Add 1 run in striker runs and 1 run in bowler's run.	Pass
18.	Scoring	Click on 4-runs	Add 4 runs in striker runs, 4 runs in bowler's runs and +1 in batsman's 4's column.	Add 4 runs in striker runs, 4 runs in bowler's runs	Pass
19.	Scoring	Click on 0	Add 1 ball in batsman and bowler.	Add 1 ball in batsman and bowler.	Pass

20.	Scoring	Click on Wicket	End inning of batsman, add 1 wicket in batting team as well as bowler.	End inning of batsman, add 1 wicket in batting team.	Fail
21.	Scoring	Selects new batsman	Display on scorecard.	Displays batsman on scorecard	Pass
22.	Scoring	Selects new bowler	Display on scorecard.	Display bowler on scorecard.	Pass
23.	Match	Selects same batsman on strike and non-strike	Both batsmen cannot be same	Redirects to scoring page	Pass
24.	Scoring	Click on Bowled	Add 1 wicket to batting team and display prompt.	Prompt displayed but no wicket was added to the batting team	Fail
25.	Scoring	Click on Stumped	Add wicket, add bowl played in batsman profile.	No wicket was added. No bowl was added.	Fail
26.	Scoring	Click on 0	Add 1 ball in batsman and bowler.	Add 1 ball in batsman and bowler.	Pass

Table 5.5 Live Scoring

27.	Scorecard	Click on Save	Data should get stored in database	No error but data didn't get stored	Fail

28.	Scorecard	Select the match	Displays the score of respective matches.	Displays the score of respective matches.	Fail

Table 5.6 Scorecard

Following changes were made to fix the errors

Test Case: 15

As there is generally 1 run for a wide ball the strike shouldn't rotate. By adding line given below it doesn't rotate strike as the last batsman remains as On-Strike batsman.

Solution:

```
//Batting team
match.teamScoreboard[track].runsFromExtras++;
//Bowler
match.teamLineUp[1 - track][bowlerId].wideGiven++;
//Strike doesn't change
match.lastBatsman = match.onStrikeBatsman;
```

15.	Scoring	Click on wide	Should add 1 run in batting team and +1 in extras.	Add 1 run in batting team and +1 in extras.	Pass

Test Case: 20

Here a wicket should be added to the batting team and the bowler should get 1 wicket except if it is a run-out.

Solution:

```
if (x == "bowled") {
      match.lastWicketFallMessage = `Last batsman:
<b>${match.onStrikeBatsman}</b> b <b>${match.onStrikeBowler}</b>`;
      match.teamLineUp[track][batsmanId].status = `b ${match.onStrikeBowler}`;
   } else if (x == "lbw") {
      match.lastWicketFallMessage = `Last batsman:
<b>${match.onStrikeBatsman}</b> lbw <b>${match.onStrikeBowler}</b>`;
      match.teamLineUp[track][batsmanId].status = `lbw ${match.onStrikeBowler}`;
   } else {
```

```
        match.lastWicketFallMessage = `Last batsman:
<b>${match.onStrikeBatsman}</b> hit-wicket b <b>${match.onStrikeBowler}</b>`;
        match.teamLineUp[track][
            batsmanId
        ].status = `hit-wicket b ${match.onStrikeBowler}`; }
```

20.	Scoring	Click on Wicket	End inning of batsman, add 1 wicket in batting team as well as bowler.	End inning of batsman, add 1 wicket in batting team as well as bowler.	Pass

Test Case: 24

Here the team should get 1 wicket but it didn't show and batsman wasn't also shown as out.

Solution:

```
// batting team scoreboard
    match.teamScoreboard[track].ballsPlayed++;
    match.teamScoreboard[track].wicketFall++;
    match.teamScoreboard[track].curOver.push("W");
 // batsman profile
    match.teamLineUp[track][batsmanId].hasBatted = true;
    match.teamLineUp[track][batsmanId].ballFaced++;
    match.teamLineUp[track][batsmanId].gotOut = true;
// bowler profile
    match.teamLineUp[1 - track][bowlerId].ballBowled++;
    match.teamLineUp[1 - track][bowlerId].dotGiven++;
    match.teamLineUp[1 - track][bowlerId].wicketTaken++;
```

24.	Scoring	Click on Bowled	Add 1 wicket to batting team and display prompt.	Add 1 wicket to batting team and display prompt.	Pass

Test Case: 25

Here, when the user clicks on stumped it should add wicket, add bowl in batsman profile who got out.

Solution:
```
match.lastWicketFallMessage = `Last batsman: <b>${
    match.onStrikeBatsman
}</b> st <b>${document.querySelector("#stumpedByOption").value}</b> b <b>${
    match.onStrikeBowler
}</b>`;
match.teamLineUp[track][batsmanId].status = `st ${
    document.querySelector("#stumpedByOption").value
} b ${match.onStrikeBowler}`;
// batting team scoreboard
match.teamScoreboard[track].ballsPlayed++;
match.teamScoreboard[track].wicketFall++;
match.teamScoreboard[track].curOver.push("W");
// batsman profile
match.teamLineUp[track][batsmanId].ballFaced++;
match.teamLineUp[track][batsmanId].gotOut = true;
// bowler profile
match.teamLineUp[1 - track][bowlerId].ballBowled++;
match.teamLineUp[1 - track][bowlerId].dotGiven++;
match.teamLineUp[1 - track][bowlerId].wicketTaken++;
match.lastBatsman = match.onStrikeBatsman;
```

25.	Scoring	Click on Stumped	Add wicket, add bowl played in batsman profile.	Add wicket, add bowl played in batsman profile.	Pass

Test Case: 27

When the game finishes, and user wishes to save the data. They will click on save button. The data should be stored in database but blank data was entered.

```
$getPlayerId2 = "SELECT * FROM `player` WHERE `teamId` = '$teamId'";
   $result3 = $con->query($getPlayerId2);
   $j = 0;
   while ($row3 = $result3 -> fetch_assoc()) {
      $playerId = $row3['playerId'];
      //Insert into Score Table for Team-1
      $score1 = "INSERT INTO `score` (`matchId`, `teamId`, `playerId`, `runScored`,
`ballFaced`, `ballDotted`, `fourHitted`, `sixHitted`, `overBowled`, `runGiven`,
`dotGiven`, `maidenGiven`, `fourConsidered`, `sixConsidered`, `wideGiven`,
`noBallGiven`, `wicketTaken`)
         VALUES ('$matchId', '$teamId', '$playerId', '$playerRunScored2[$j]',
'$playerBallFaced2[$j]', '$playerBallDotted2[$j]', '$playerFourHitted2[$j]',
'$playerSixHitted2[$j]', '$playeroverBowled2[$j]', '$playerRunsGiven2[$j]',
         '$playerDotGiven2[$j]', '$playerMaidenGiven2[$j]',
'$playerFourConsidered2[$j]', '$playerSixConsidered2[$j]', '$playerWideGiven2[$j]',
'$playerNoBallGiven2[$j]', '$playerWicketTaken2[$j]')";
      if ($con->query($score1) === TRUE) {
         echo "\nScore Inserted";
      }
      else {
         echo "Error 500";
      }
      $j = $j+1;
   }
```

27.	Scorecard	Click on Save	Data should get stored in database	No error but data didn’t get stored	Fail

Solution 2:

```
// Insert into Player table of Team-1
   $playerNames1 = $team2Players[0];
```

```
    $playerRole1 = $team2Players[1];
    $playerStatus1 = $team2Players[2];
    $playerRunScored1 = $team2Players[3];
    $playerBallFaced1 = $team2Players[4];
    $playerBallDotted1 = $team2Players[5];
    $playerFourHitted1 = $team2Players[6];
    $playerSixHitted1 = $team2Players[7];
    $playeroverBowled1 = $team2Players[8];
    $playerRunsGiven1 = $team2Players[9];
    $playerDotGiven1 = $team2Players[10];
    $playerMaidenGiven1 = $team2Players[11];
    $playerFourConsidered1 = $team2Players[12];
    $playerSixConsidered1 = $team2Players[13];
    $playerWideGiven1 = $team2Players[14];
    $playerNoBallGiven1 = $team2Players[15];
    $playerWicketTaken1 = $team2Players[16];
$getPlayerId1 = "SELECT * FROM `player` WHERE `teamId` = '$teamId'";
    $result2 = $con->query($getPlayerId1);
    $j = 0;
    while ($row2 = $result2 -> fetch_assoc()) {
        $playerId = $row2['playerId'];
        //Insert into Score Table for Team-1
        $score1 = "INSERT INTO `score` (`matchId`, `teamId`, `playerId`, `runScored`,
`ballFaced`, `ballDotted`, `fourHitted`, `sixHitted`, `overBowled`, `runGiven`,
`dotGiven`, `maidenGiven`, `fourConsidered`, `sixConsidered`, `wideGiven`,
`noBallGiven`, `wicketTaken`)
            VALUES ('$matchId', '$teamId', '$playerId', '$playerRunScored1[$j]',
'$playerBallFaced1[$j]', '$playerBallDotted1[$j]', '$playerFourHitted1[$j]',
'$playerSixHitted1[$j]', '$playeroverBowled1[$j]', '$playerRunsGiven1[$j]',
            '$playerDotGiven1[$j]', '$playerMaidenGiven1[$j]',
'$playerFourConsidered1[$j]', '$playerSixConsidered1[$j]', '$playerWideGiven1[$j]',
'$playerNoBallGiven1[$j]', '$playerWicketTaken1[$j]')";
        if ($con->query($score1) === TRUE) {
            echo "\nScore Inserted";
```

```
        }
        else {
            echo "Error 500";
        }
        $j = $j+1;
    }
```

27.	Scorecard	Click on Save	Data should get stored in database	Data get stored in database	Pass

Test Case: 28

Here when the user selects the match it should show the score of respective matches.

```
//Display Team-1 Name (Same for Team-2 Name)
if(isset($_POST['team1Name'])){
    $matchId = $_POST['matchId'];
    $teams = array();
    $query = "SELECT * FROM `team` WHERE `matchId` = '$matchId'";
    $result = mysqli_query($status, $query);
        while ($row3 = $result -> fetch_assoc()) {
            array_push($teams, $row3['teamName']);
        }
        $team = "SELECT * FROM `team` WHERE `teamName` = '$teams[0]'";
        $result2 = mysqli_query($status, $team);
        if ($row5 = $result2 -> fetch_assoc()) {
            echo "<span>" . "<div id=\"T\"><p>" . $row5["teamName"] . "</p></div>" .
"</span>";
        }
}
//Display Team-1 Player's Batting Data (Same for Team-2 Player's Batting Data)
if(isset($_POST['team1batting'])){
    $matchId = $_POST['matchId'];
    $teams = array();
    $query = "SELECT * FROM `team` WHERE `matchId` = '$matchId'";
    $result = mysqli_query($status, $query);
```

```
        while ($row3 = $result -> fetch_assoc()) {
            array_push($teams, $row3['teamId']);
        }
        $player = "SELECT * FROM `player` WHERE `teamId` = '$teams[0]'";
        $result2 = mysqli_query($status, $player);
        while ($row4 = $result2 -> fetch_assoc()) {
            echo '<tr>';
            echo "<td>" . "<div id=\"T\"><p>" . $row4["playerName"] . "</p></div>" .
"</td>";
            $playerid = $row4["playerId"];
            $getscore = "SELECT * FROM `score` WHERE `playerId` = '$playerid'";
            $result3 = mysqli_query($status, $getscore);
            $result3 = $result3 -> fetch_assoc();
            echo "<td>" . "<div id=\"T\"><p>" . $result3["runScored"] . "</p></div>" .
"</td>";
            echo "<td>" . "<div id=\"T\"><p>" . $result3["ballFaced"] . "</p></div>" .
"</td>";
            echo "<td>" . "<div id=\"T\"><p>" . $result3["ballDotted"] . "</p></div>" .
"</td>";
            echo "<td>" . "<div id=\"T\"><p>" . $result3["fourHitted"] . "</p></div>" .
"</td>";
            echo "<td>" . "<div id=\"T\"><p>" . $result3["sixHitted"] . "</p></div>" .
"</td>";
            echo '</tr>';
        }
}
/Display Team-1 Player Bowling Data (Same for Team-2 Player Bowling Data)
if(isset($_POST['team1bowling'])){
    $matchId = $_POST['matchId'];
    $teams = array();
    $query = "SELECT * FROM `team` WHERE `matchId` = '$matchId'";
    $result = mysqli_query($status, $query);
                while ($row3 = $result -> fetch_assoc()) {
                        array_push($teams, $row3['teamId']);
```

```
                }
        $player = "SELECT * FROM `player` WHERE `teamId` = '$teams[0]'";
        $result2 = mysqli_query($status, $player);
                while ($row4 = $result2 -> fetch_assoc()) {
            echo '<tr>';
                        echo "<td>" . "<div id=\"T\"><p>" . $row4["playerName"] .
"</p></div>" . "</td>";
            $playerid = $row4["playerId"];
            $getscore = "SELECT * FROM `score` WHERE `playerId` = '$playerid'";
            $result3 = mysqli_query($status, $getscore);
            $result3 = $result3 -> fetch_assoc();
            echo "<td>" . "<div id=\"T\"><p>" . $result3["overBowled"] . "</p></div>" .
"</td>";
            echo "<td>" . "<div id=\"T\"><p>" . $result3["runGiven"] . "</p></div>" .
"</td>";
            echo "<td>" . "<div id=\"T\"><p>" . $result3["maidenGiven"] . "</p></div>" .
"</td>";
            echo "<td>" . "<div id=\"T\"><p>" . $result3["wicketTaken"] . "</p></div>" .
"</td>";
            echo "<td>" . "<div id=\"T\"><p>" . $result3["wideGiven"] . "</p></div>" .
"</td>";
            echo "<td>" . "<div id=\"T\"><p>" . $result3["noBallGiven"] . "</p></div>" .
"</td>";
            echo '</tr>';
                }
}
```

28.	Scorecard	Select the match	Displays the score of respective matches.	Displays the score of respective matches.	Pass

5.3.2 Beta Testing

The first step would be to identify the scope of the testing and the objectives to be achieved. Then, a prototype was made and was given to the user i.e. scorer who scored a live match using this system. On that basis following test cases were prepared.

Test Case No.	Description	Test Case	Expected Output	Actual Output	Remark
1.	Scoring	Click on 0	Add ball to batsman, bowler and teams.	Adds ball to batsman, bowler and teams.	Pass
2.	Scoring	Click on 1	Add 1 ball to batsman, bowler and team and 1 run to bowler and on-strike batsman and strike rotate.	Adds 1 ball to batsman, bowler and team and 1 run to bowler and on-strike batsman. No strike rotates.	Fail
3.	Scoring	Click on 2	Add 1 ball to batsman, bowler and team and 2 run to bowler and on-strike batsman.	Add 1 ball to batsman, bowler and team and 2 run to bowler and on-strike batsman.	Pass
4.	Scoring	Click on 3	Add 1 ball to batsman, bowler and team and 3 run to bowler and on-strike	Adds 1 ball to batsman, bowler and team and 3 run to bowler and on-strike	Fail

			batsman and strike rotate.	batsman. No strike rotates.	
5.	Scoring	Click on 4	Add 1 ball to batsman, bowler and team and 4 runs to bowler and on-strike batsman.	Add 1 ball to batsman, bowler and team and 4 runs to bowler and on-strike batsman.	Pass
6.	Scoring	Click on 6	Add 1 ball to batsman, bowler and team and 6 runs to bowler and on-strike batsman.	Add 1 ball to batsman, bowler and team and 6 runs to bowler and on-strike batsman.	Pass
7.	Scoring	Click on wide-ball	Add runs to team and bowler, no ball count.	Add runs to team and bowler, no ball count	Pass
8.	Scoring	Click on no-ball	Add runs to batsman, bowler and team, no ball count.	Add runs to batsman, bowler and team, no ball count.	Pass
9.	Scoring	Click on bowled	Batsman out, add wicket to team and bowler.	Batsman out, add wicket to bowler and not team.	Pass

10	Scoring	Click on LBW	Batsman out, add wicket to team and bowler.	Batsman out, add wicket to bowler and not team.	Pass
11.	Scoring	Click on Run Out	Should ask which batsman was out on which end	Wicket of on strike batsman.	Fail

Table 5.7 Beta-Testing Table

Following were the solutions done for correction of errors:

Test Case 2, 4:

Here as the batsman has taken 1 and 3 runs the strike should rotate.

Solution:

```
// change strike
  if (runTaken % 2 == 1) {
    [match.onStrikeBatsman, match.nonStrikeBatsman] = [
      match.nonStrikeBatsman,
      match.onStrikeBatsman,
    ];
  }
```

2.	Scoring	Click on 1	Add 1 ball to batsman, bowler and team and 1 run to bowler and on-strike batsman and strike rotate.	Adds 1 ball to batsman, bowler and team and 1 run to bowler and on-strike batsman. No strike rotates.	Pass
4.	Scoring	Click on 3	Add 1 ball to batsman, bowler and team and 3	Add 1 ball to batsman, bowler and team and 3	Pass

			run to bowler and on-strike batsman and strike rotate.	run to bowler and on-strike batsman and strike rotate.	

Test Case 11:

Here as the wicket is in the form of run-out. System should ask which batsman was out and on which end.

Solution:

```
elevenOption +=`<option
value="${match.onStrikeBatsman}">${match.onStrikeBatsman}</option>`;
   elevenOption += `<option
value="${match.nonStrikeBatsman}">${match.nonStrikeBatsman}</option>`;
   document.querySelector("#whoGotOutOption_r").innerHTML = elevenOption;
   elevenOption = "";
   match.teamLineUp[1 - track].forEach((e) => {
      elevenOption += `<option value="${e.name}">${e.name}</option>`; });
```

11.	Scoring	Click on Run Out	Should ask which batsman was out on which end	Prompt which asks which batsman was out on which end	Pass

Chapter 6: Result and Discussion

6.1 Test Report

The testing phase of project development is critical. The testing step allows you to determine whether all of the capabilities are being executed correctly. The testing phase began with the creation of test cases for each module as well as the design of the modules themselves. The process for integrating test cases was completed. Each module was then examined, and test cases were created based on the findings. The test cases provided input and the expected result after entering the values. After constructing the test cases, they were validated by actually entering the inputs and determining if the estimated and actual outputs were the same or not. If the estimated output corresponded to the actual output then the test cases were remarked to be passed else, they were remarked as a failure. Not all values were tried and tested but the process made sure the system would be able to cope up with any values. After performing all the testcases, it was concluded that there were no errors. So, no further modifications were needed in the respective modules.

Based on the performed test cases and modifications the problems such as creating team, adding players, live scoring, getting scorecard were capable of tackling the problem defined for the project objectives and commercial small-scale use.

6.2 User Documentation

- Homepage

  The First Page the user can see when they visit the website.

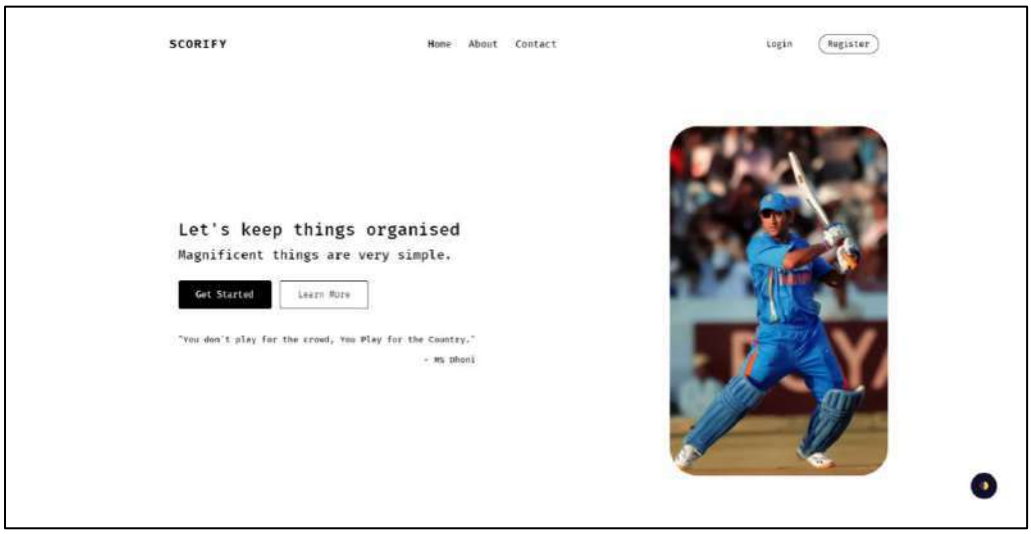


Fig. 6.1 Home Page

- Register

  The user can create their account from here which will be used for logging in to the system.

Fig. 6.2 Registration Page

- Login Page

  This is login page that will allow registered users to login themselves in to the system and use the system.

Fig. 6.3 Login Page

- Dashboard

  This is the first page the user sees after logging in

Fig. 6.4 Dashboard Page

- Create Match

  This is the first page for creating a new match. User has to fill all the match details here.

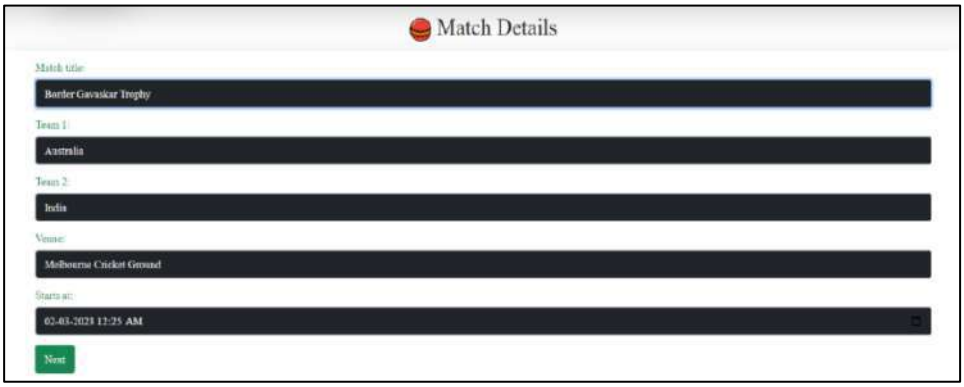


Fig. 6.5 Match Details Page

- Toss Details

  Here the user needs to enter the toss details as per which batting and bowling will be selected.

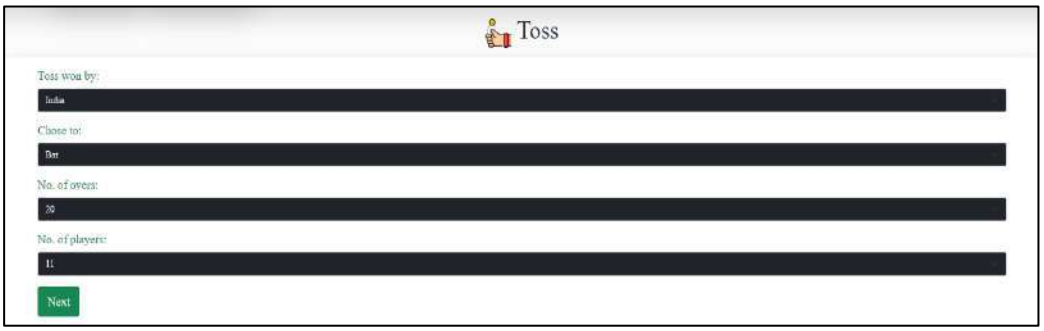


Fig 6.6 Toss Page

- Add Player

  In here the user will have to enter player names team-wise.

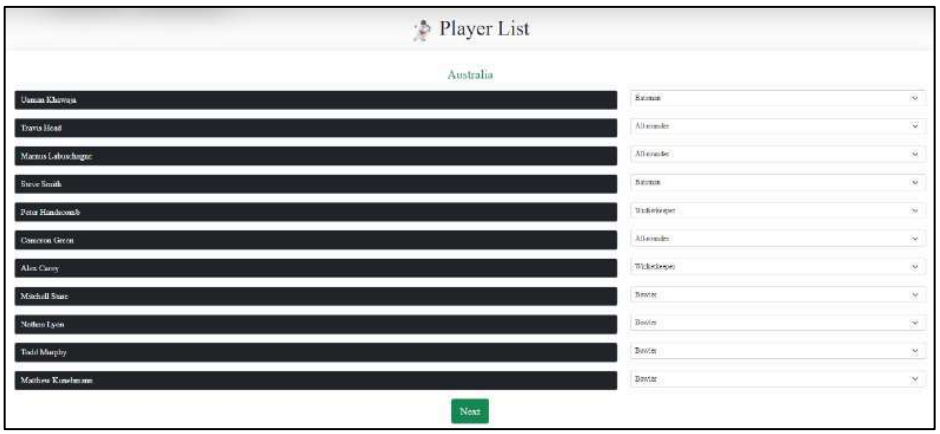


Fig 6.7 Player List Page

- Live Scoring

  This is the page where you will have to record the scores signalled by the umpires.

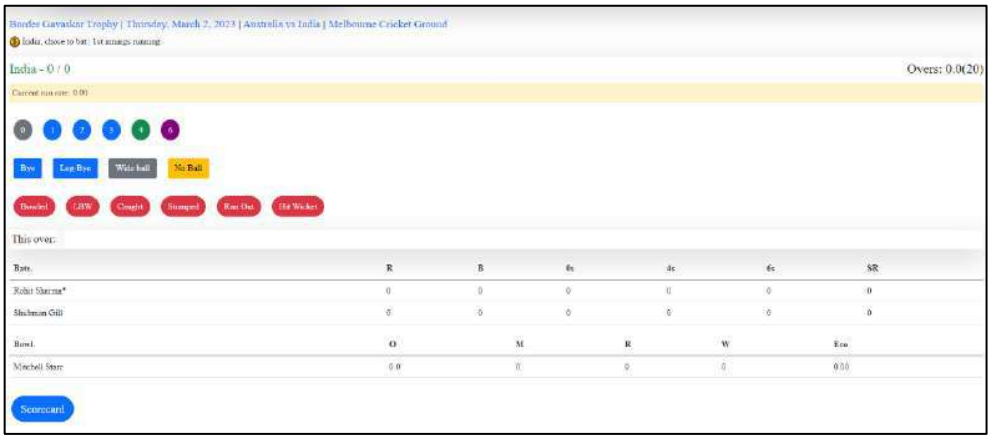


Fig 6.8 Live Scoring Page

- Scorecard

  This is the page where the user will get the scorecard of the respective match.

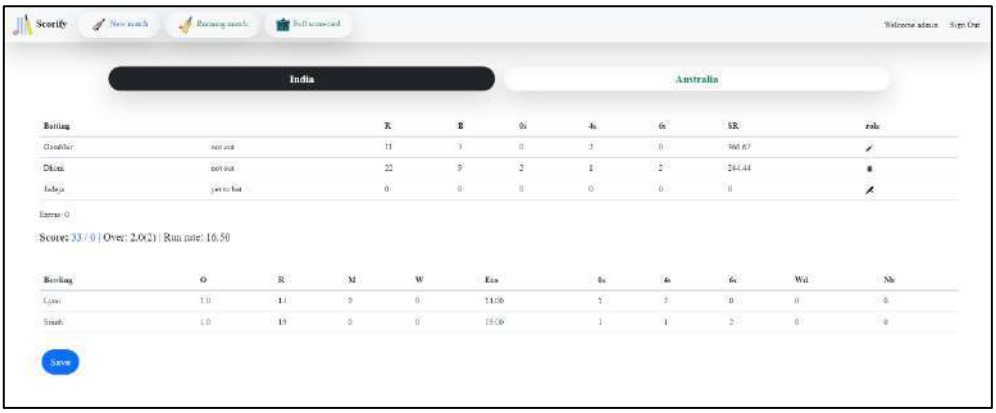


Table 6.9 Scorecard Page

Chapter 7: Conclusions

7.1 Conclusion

The most thing I learnt was time management, if we manage our time properly then we can finish anything before the deadline. I also learnt that designing and planning are one of the important things. This project took me through the various phases of project development and gave me real insight into the world of software engineering.

In conclusion, developing a live cricket scoring system using PHP and MySQL for our college project was a great learning experience. Through this project, we were able to gain practical experience in designing and implementing a full-stack application that can be used to score on-going cricket match.

PHP is a widely used open-source programming language that is easy to learn, has a large community, and is particularly suited for web development, making it an ideal choice for building dynamic and interactive web applications. Some of the benefits of using PHP include its flexibility, security, scalability, and compatibility with various platforms and databases.

Also, MySQL proves to be easy to configure unlike another database. Also, there is no need for any other data storage for same database while using different device. Maintenance of data proves to be very easy. Also, during power or server shutdown the corruption of data takes place which is eliminated in MSSQL by having features for data recovery and restoration.

Also, I've used JavaScript (JS) which is a versatile and powerful programming language that is widely used in web development. Some benefits of using JS include its ability to add interactivity and dynamic features to web pages, its compatibility with a wide range of browsers, its large community of developers and resources, and its ability to work with various frameworks and libraries.

I've also tried to implement partial load in the project. It is implemented in the main module i.e. from match creation till viewing scorecard.

In here, the user can create a new match and by inserting the details and can get started with scoring the match. The user will get to see all the details which are needful on the page. In, the end the scorecard of entire match is generated wherein the user will get option to save it by which they'll have an option to save the score of the match. If they save the data can be accessed any time when they want.

7.2 Limitations of the System

- The system can't be used to score Test / Multi-Day matches.
- Once the score is recorded it cannot be reversed.
- It can be used only on browser.

7.3 Future Scope of the System

- Mobile app.
- Display scores of matches for general people (Like cricbuzz).
- Fully responsive design.
- This Project is only capable to Handle Moderate Traffic as the hosting solution is based on 1 CPU core and 2GB RAM.

Bibliography

- **Websites:**(As of 17-06-2022)
  - https://cricheroes.in/
  - https://play-cricket.ecb.co.uk/hc/en-us/sections/115001082765-Play-Cricket-Scorer-Instructions-tablet-phone
  - https://www.php.net/manual/en/langref.php/
  - https://www.tutorialspoint.com/
  - https://www.javatpoint.com/uml-activity-diagram
  - https://www.w3schools.in/
  - https://www.lucidchart.com/
  - https://www.geeksforgeeks.org/
  - https://app.diagrams.net/
  - https://online.visual-paradigm.com/drive/#diagram:proj=0&type=GanttChart&workspace=mhujsaxv&id=2
  - UML Diagrams Full Course (Unified Modeling Language) - YouTube

- **Reference Books** (Referred from 17-06-2022)
  - Software Engineering, "Ian Somerville", 8th Edition, Pearson Education.
  - Database System Concepts, "Henry F. Korth, Abraham Silberschatz. S.Sudarshan" McGrawHill 4th Edition
  - The Unified Modeling Language Reference Manual, 2nd Edition, "James Rumbaugh, Ivar Jacobson, Grady Booch"
  - Learning PHP, MySQL, JavaScript, & CSS: A Step-by-Step Guide to Creating Dynamic Websites,Second EditionRobin Nixon, O'Reilly
  - Fundamentals of Creating a Great UI/UX Author: - Creative Tim as of 23/08/22