**Table Of Contents:**

|  |  |  |
| --- | --- | --- |
| Chapter No. | Chapter Name | Page No. |
|  | Synopsis | 1 |
| 1 | Introduction   * 1. Background   2. Objective   3. Purpose   4. Application   5. Scope   6. Achievements | 3  3  3  3  3  3 |
| 2 | Survey of Technology | 4 |
| 3 | Requirement And Analysis  3.1 Problem Definition  3.1.1Sub-systems  3.1.2 Problem Description  3.2 Requirement Specification  3.2.1 Requirement Gathering  3.2.2 Requirement Analysis  3.2.3Functional Requirements  3.2.4 Non-functional Requirements  3.2.5 System Requirements  3.3 Planning And Scheduling  3.3.1 Activity Table  3.3.2 Gantt Chart  3.4 Hardware And Software Requirements  3.4.1 Hardware Requirements  3.4.2 Software Requirements  3.5 Concept Models  3.5.1 Entity Relationship Diagram  3.5.1.1 Entity Sets  3.5.1.2 Relationship Sets  3.5.1.3 ER-Diagram  3.5.2 Schema Diagram  3.5.3 Data Flow Diagram  3.5.3.1 Level 0 (Context Level)  3.5.3.2 Level 1  3.5.3.3 Level 2  3.5.4 Use Case Diagram  3.5.4.1 Use Case Diagram  3.5.4.2 Use Case Description  3.5.5 Sequence Diagram  3.5.6 Activity Diagram  3.5.7 State-Chart Diagram | 7  7  7  8  10  10  10  12  13  14  14  17  18  20  25  22  23  23  25  26  27  29  32  34 |
| 4 | System Design  4.1 User Interface  4.2 Test Cases | 35  38 |
|  | Bibliography  Websites  Reference Books | 41  41 |

**List Of Figures:**

|  |  |  |
| --- | --- | --- |
| Fig. No. | Figure Name | Page No. |
| 1.1 | Architecture | 1 |
| 3.1 | Requirement Gathering | 8 |
| 3.2 | Gantt Chart | 11 |
| 3.3 | User Entity Set | 16 |
| 3.4 | Match Entity Set | 16 |
| 3.5 | Team Entity Set | 17 |
| 3.6 | Player Entity Set | 17 |
| 3.7 | Score Entity Set | 18 |
| 3.8 | Create Relationship Set | 18 |
| 3.9 | Has Relationship Set | 19 |
| 3.10 | ER Diagram | 19 |
| 3.11 | Schema Diagram | 20 |
| 3.12 | Level 0 DFD | 23 |
| 3.13 | Level 1 DFD | 23 |
| 3.14 | Level 2 DFD for Match | 24 |
| 3.15 | Level 2 DFD for Scorecard | 24 |
| 3.16 | Use Case Diagram | 25 |
| 3.17 | Sequence Diagram for Registration | 28 |
| 3.18 | Sequence Diagram for Login | 28 |
| 3.19 | Sequence Diagram for Match Creation | 29 |
| 3.20 | Sequence Diagram for Scoring | 29 |
| 3.21 | Sequence Diagram for Scorecard | 30 |
| 3.22 | Sequence Diagram for Create Team & Player | 30 |
| 3.23 | Activity Diagram | 32 |
| 3.24 | State-Chart Diagram | 33 |
| 4.1 | UI for Home Page | 33 |
| 4.2 | UI for Registration Page | 34 |
| 4.3 | UI for Login | 34 |
| 4.4 | UI for Dashboard | 34 |
| 4.5 | UI for Scoring | 35 |
| 4.6 | UI for Scorecard | 35 |
| 4.7 | UI for Create Team & Player | 36 |

**List Of Tables:**

|  |  |  |
| --- | --- | --- |
| Table No. | Table Name | Page No. |
| 3.1 | Activity Table | 13 |
| 3.2 | ER Diagram Notation | 15 |
| 3.3 | Schema Diagram Notation | 10 |
| 3.4 | Data Flow Diagram Notation | 22 |
| 3.5 | Use Case Diagram Notation | 24 |
| 3.6 | Sequence Diagram Notation | 27 |
| 3.7 | Activity Diagram Notation | 31 |
| 3.8 | State-Chart Diagram Notation | 32 |
| 4.1 | Test Cases | 36 |

## 

## **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.**

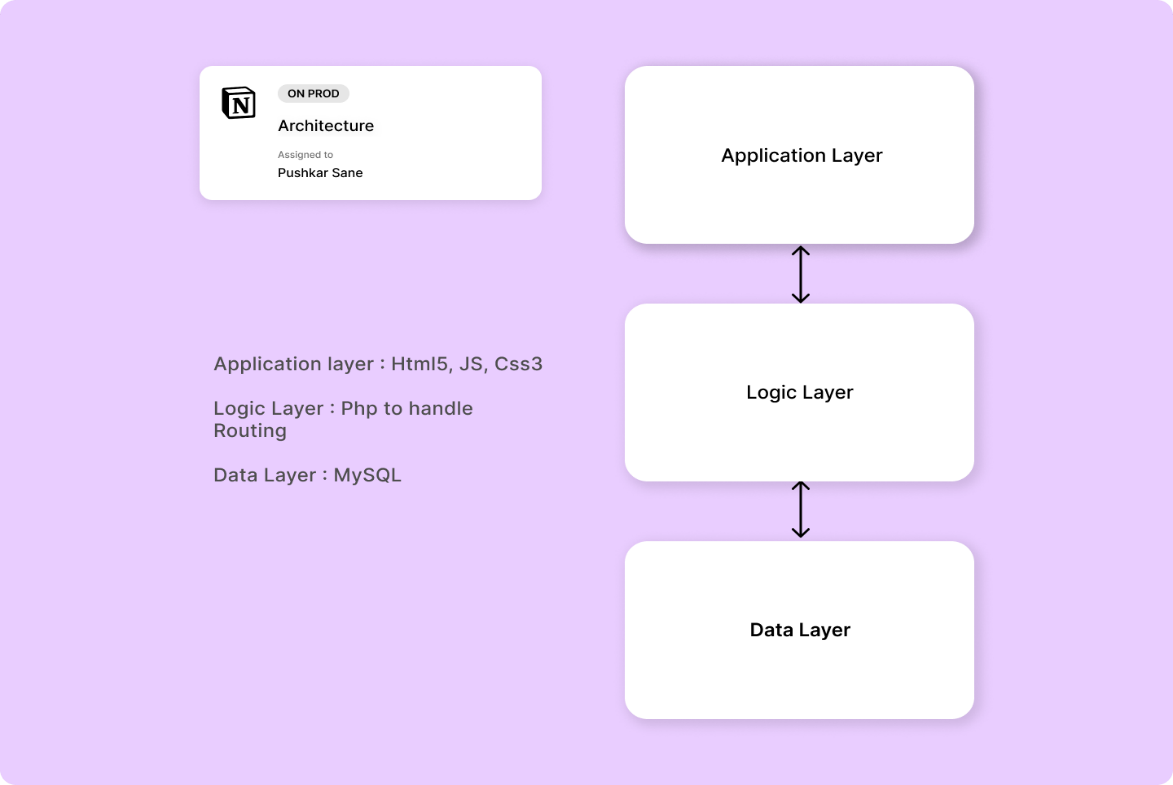
* To reduce multiple manual calculations during the match such as net run rate, batsman’s analysis, bowler’s analysis.
* To provide ball-by-ball update of the match.
* To provide detailed scorecard.

## **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.

## **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.

## **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.

## **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.

## **Application**

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

## **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.

## **Achievements**

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

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

1. **Front-end Languages:**
   1. HTML 5
   2. CSS
   3. JavaScript
   4. ASP.Net
   5. Bootstrap
   6. Python
2. **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.
3. **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.
4. **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.
5. **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.
6. **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.
7. **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.
8. **Back-end Languages:**
   * PHP
   * MySQL
   * MongoDB
9. **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.
10. **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.
11. **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.

## **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.

### **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.

### **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.

## **Requirements Specification**

### **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.

### **Requirement analysis**

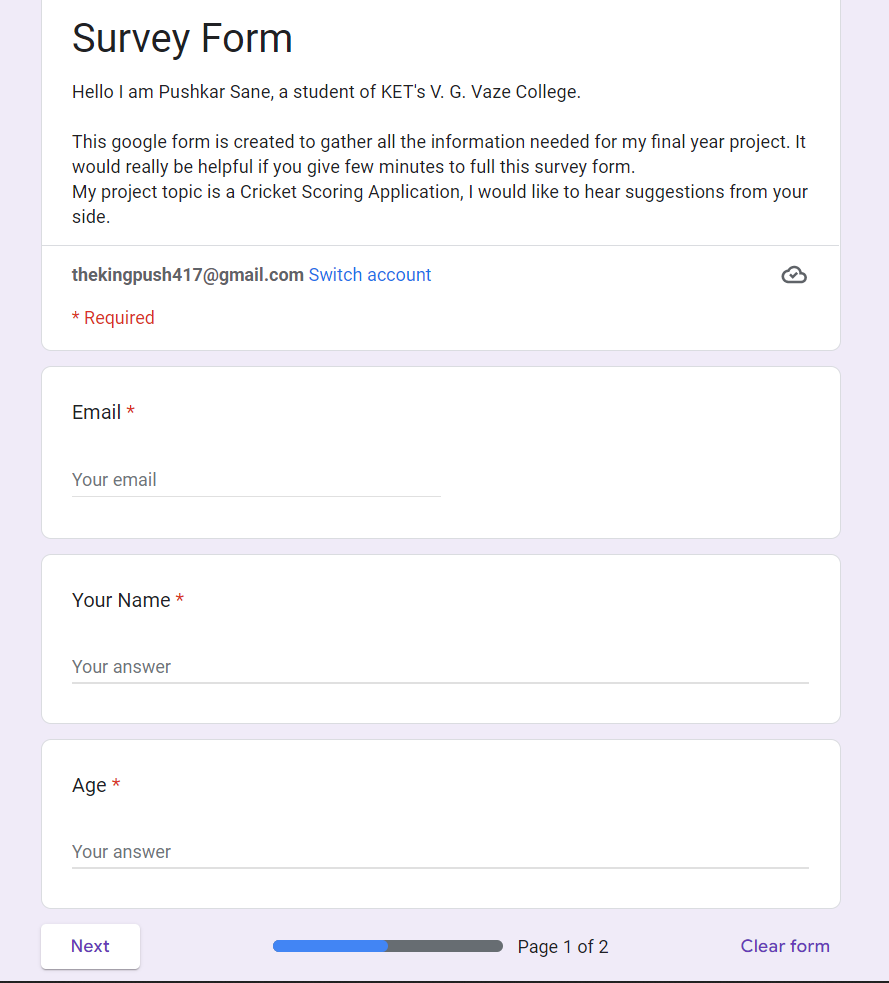
Identify stakeholders i.e., in case the people who are going to use this site.

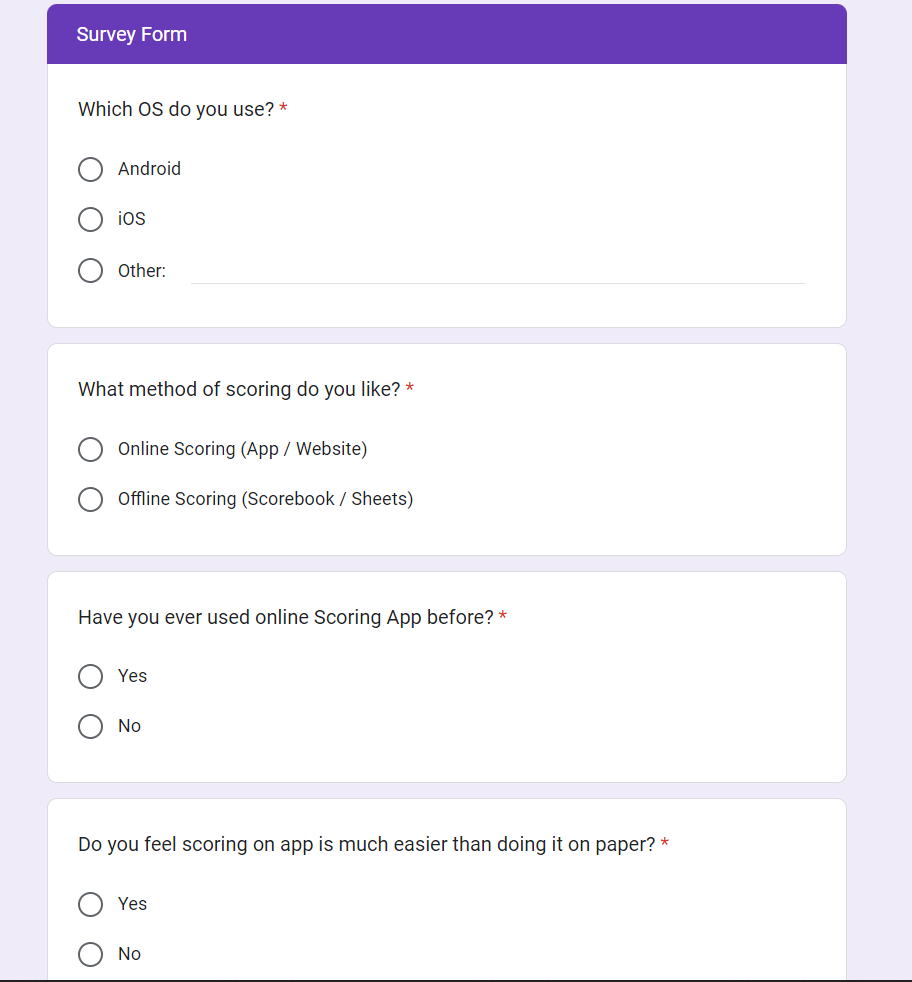
* Capture requirements
* Holding One-One interviews
* Conduct Group Workshops
* Get Feedbacks
* Build Small Prototypes

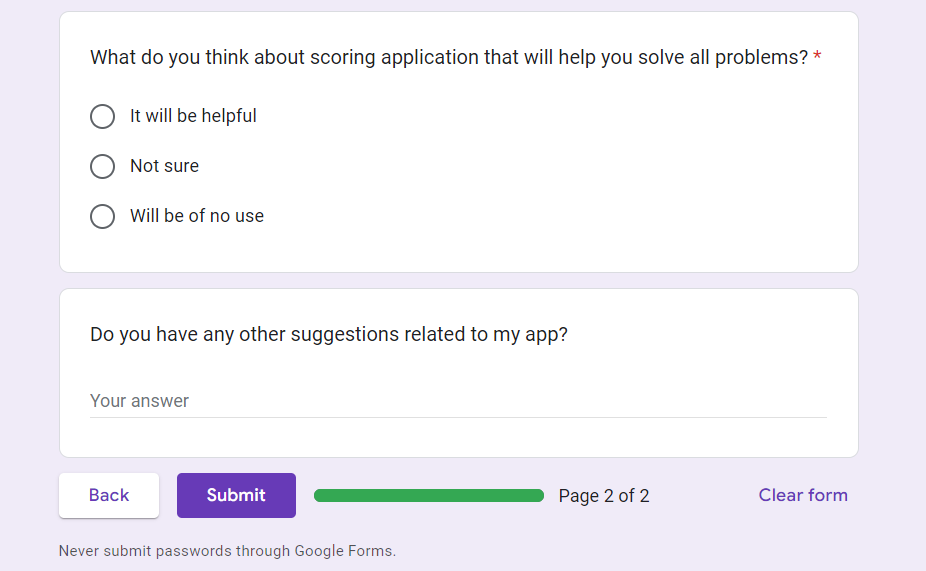
For the current situation, I used the Feedback method to identify the requirements for the project using Google Forms as a means to collect the data.

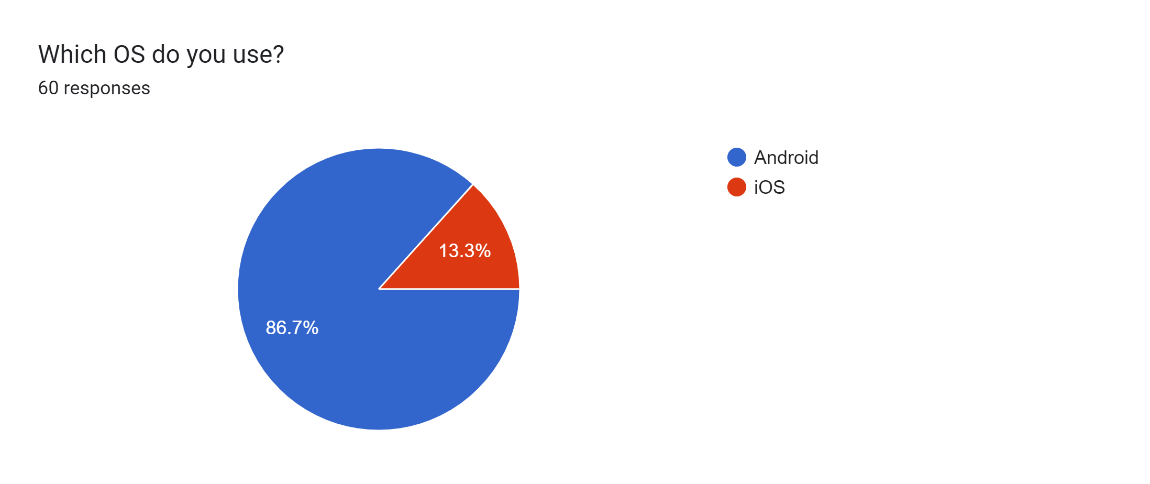
The link for spreadsheet of responses I got is below:

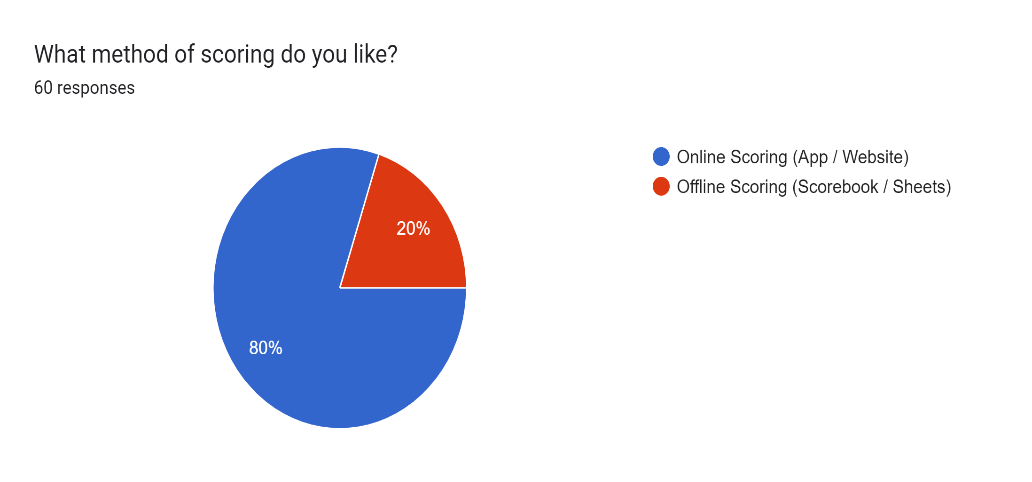
<https://docs.google.com/spreadsheets/d/1TcOC6hnWNdawqq9xfcP0FMfusH_fRYPg2GTdW_PTRbY/edit?usp=sharing\>

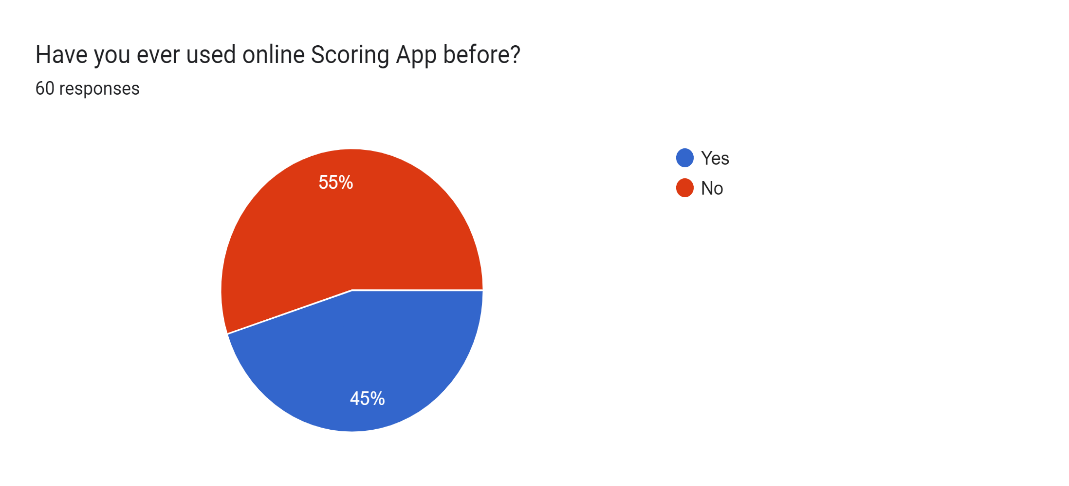
The below figures are the collected data that was generated. 

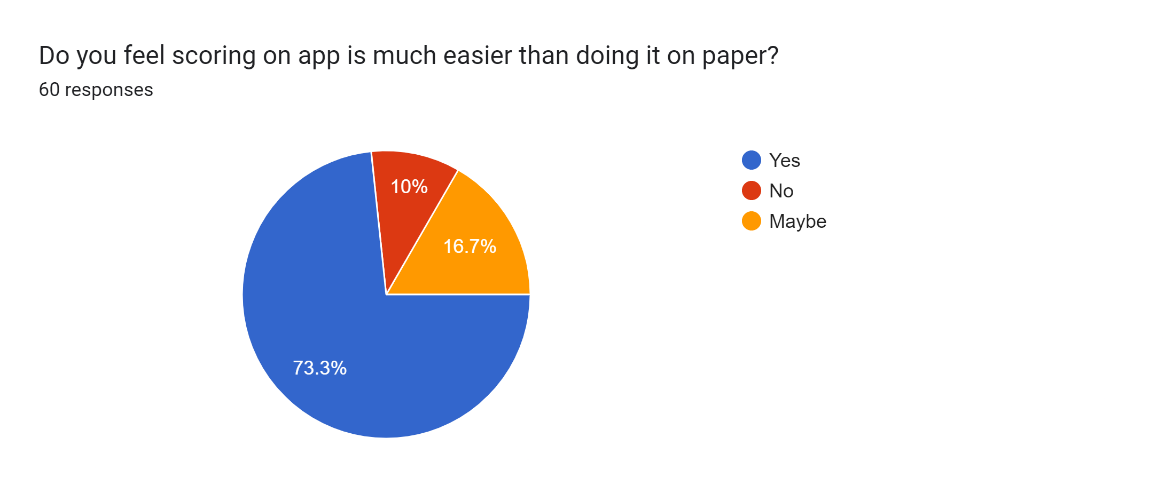


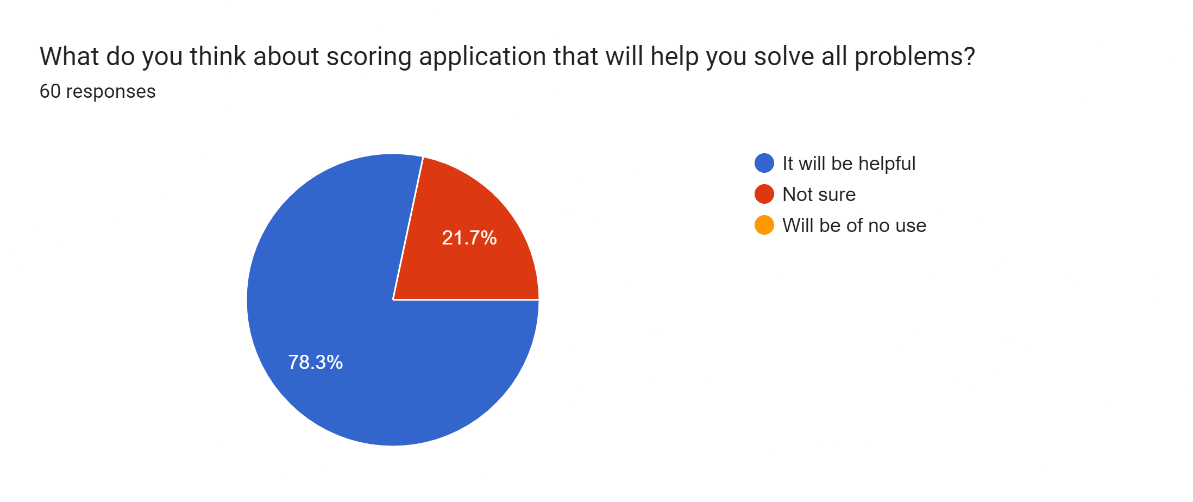












***Fig. 3.1 Requirement Gathering***

### **3.2.3 Functional Requirements**

* 1. 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.
  2. Ball-to-ball Scoring: The scorer should be able to maintain the ball-to-ball score of the ongoing match.
  3. Professional Scorecard: After the match ends, the users should be able to view scorecard which includes batting analysis, bowling analysis, etc.
  4. 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**

* 1. Usability: Should be user-friendly and only required detail should be shown in a minimal way.
  2. Reliability: The system should be user friendly to use.
  3. Flexibility: can run on any Platform.

### **System Requirements**

1. Login:
   1. Description: The user will be able to login to their respective accounts.
   2. Input: Username, password
   3. Source: User
   4. Output: Gets logged in to the system.
   5. Destination: -
   6. Action: After entering username and password, the user will get redirected to the dashboard,
   7. Pre-condition: The user must have an account
   8. Post-condition: -
2. Register:
   1. Description: The user will be able to create a new account.
   2. Input: Name, Email, Username, Password.
   3. Source: User
   4. Output: Account gets created.
   5. Destination: Entered data will get stored in database.
   6. Action: After registering, the account of user gets created.
   7. Pre-condition: User must provide the required details.
   8. Post-condition: User can login to their account with registered username and password.
3. Create a match:
   1. Description: User will be able to create a new match.
   2. Input: Details of match, teams and players.
   3. Source: User.
   4. Output: New match is created.
   5. Destination: Data will be displayed and added on scorecard.
   6. Action: Match gets created as per the given details.
   7. Pre-condition: User must be logged in to their account:
   8. Post-condition: User will be able to add team and players.
4. Live Scoring:
   1. Description: User will be able to maintain score of ongoing matches.
   2. Input: Event happening on each ball.
   3. Source: User
   4. Output: Updates the score.
   5. Destination: User interface.
   6. Action: Score gets updated after each ball.
   7. Pre-condition: Match should be created/started.
   8. Post-condition: -
5. Scorecard (View):
   1. Description: User will be able to view the scorecard of completed match.
   2. Input: Select the match.
   3. Source: Database.
   4. Output: Scorecard is displayed.
   5. Destination: -
   6. Action: User gets to view the scorecard.
   7. Pre-condition: Match should be finished.
   8. Post-condition: -

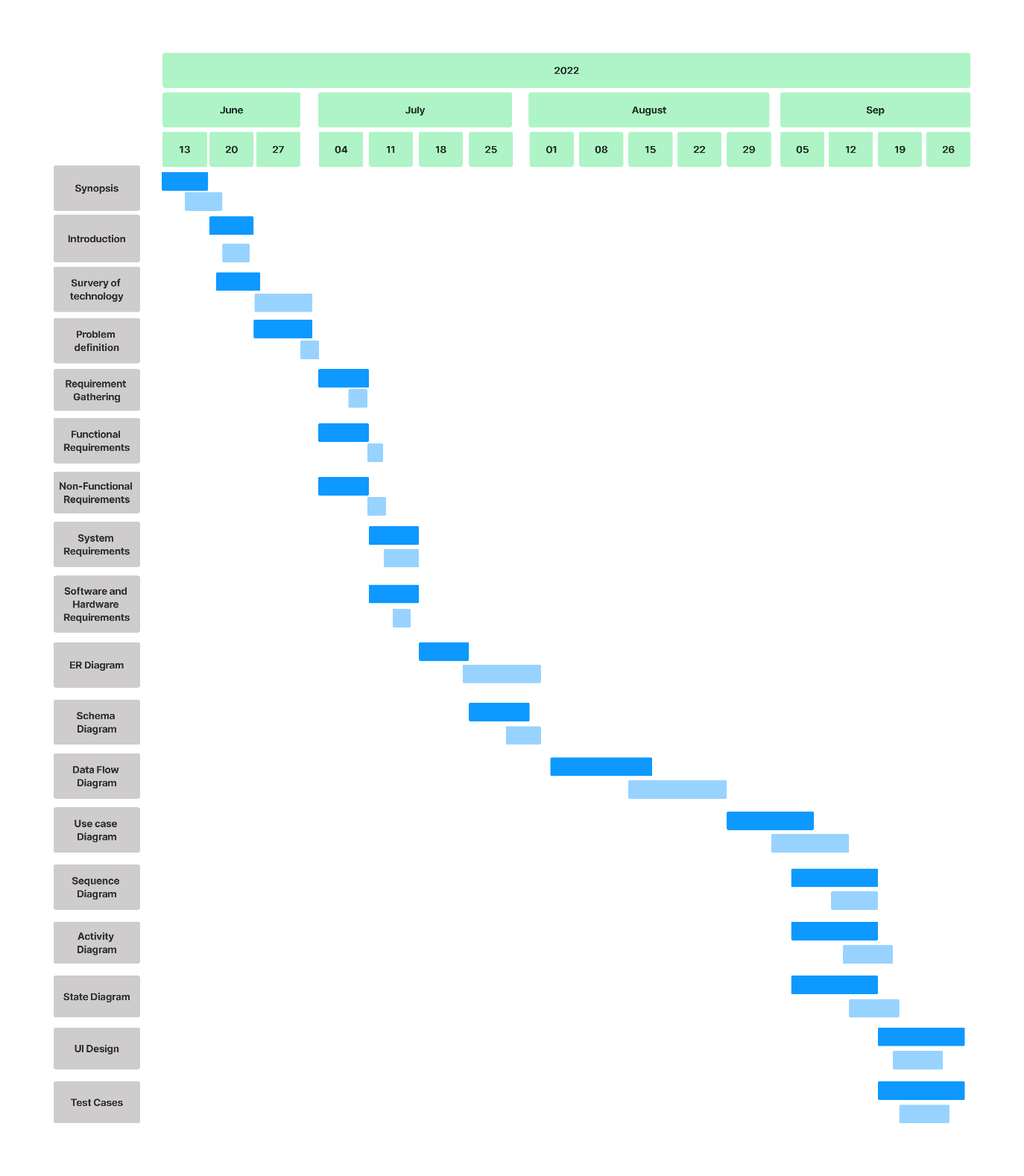
## **3.3 Planning and Scheduling**

### 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 * Sequence Diagram * Activity Diagram * State Diagram | 18-07-2022  25-07-2022  01-08-2022  29-08-2022  12-09-2022  12-09-2022  12-09-2022 | 23-07-2022  30-07-2022  13-08-2022  10-09-2022  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***

### **Gantt Chart**



***Fig. 3.2 Gantt Chart***

* 1. **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 itsproperties. 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 |

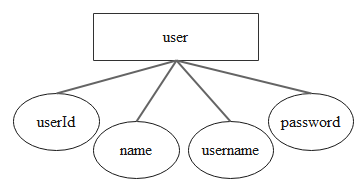
***Table 3.2 ER-Diagram Notations***

List of entity sets:

* User
* Match
* Player
* Team
* Score

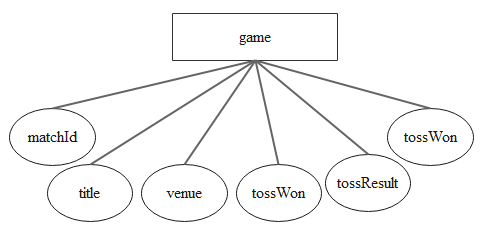
List of relationship sets:

1. User crates a Game
2. Game has Teams
3. Team has Player
4. Player has Score  
   * 1. **Entity Sets:**
5. 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.



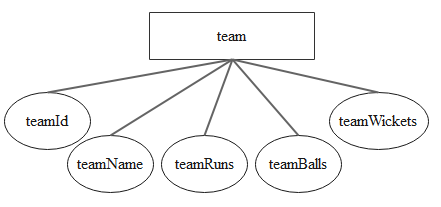
***Fig. 3.3 User Entity Set***

1. 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.



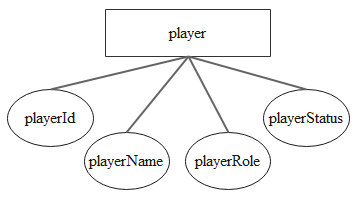
***Fig. 3.4 Match Entity Set***

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



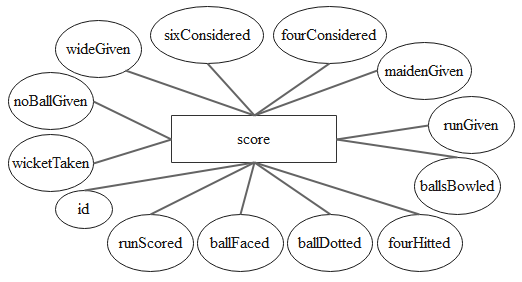
***Fig. 3.5 Team Entity Set***

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



***Fig. 3.6 Player Entity Set***

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

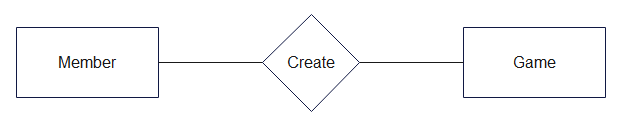


***Fig. 3.7 Score Entity Set***

* + 1. **Relationship Sets**

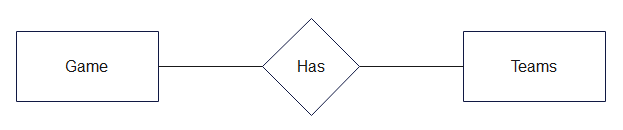
A relationship is used to describe the relation between entities. Diamond or rhombus is used to represent the relationship.

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



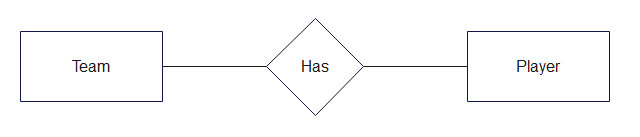
***Fig. 3.8 Create Relationship Set***

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



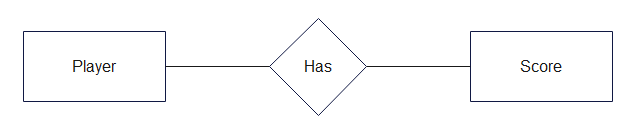
***Fig. 3.9 Has Relationship Set***

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



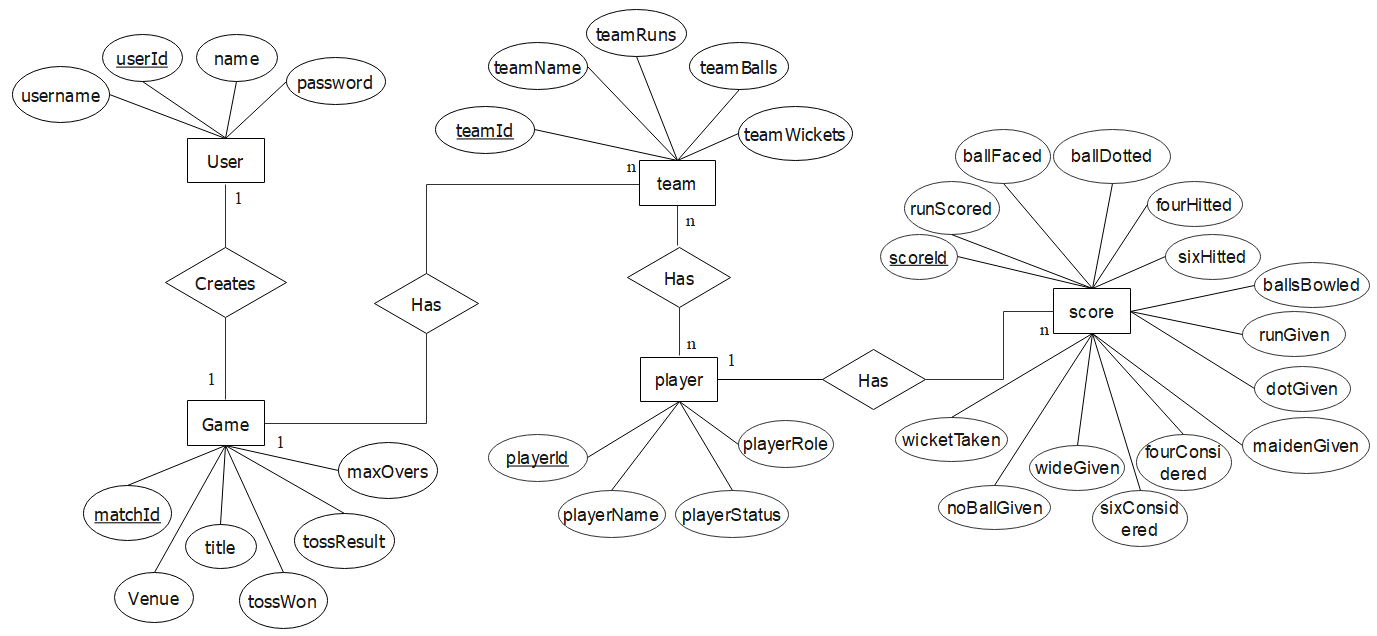
***Fig. 3.10 Has Relationship Set***

1. Player has Score
   1. Here a match can have only one score
   2. Mapping Cardinality: One to One



***Fig. 3.11 Has relationship set***

**3.5.1.3 Entity-Relationship Diagram**



***Fig. 3.12 ER Diagram***

* + 1. **Schema Diagram**

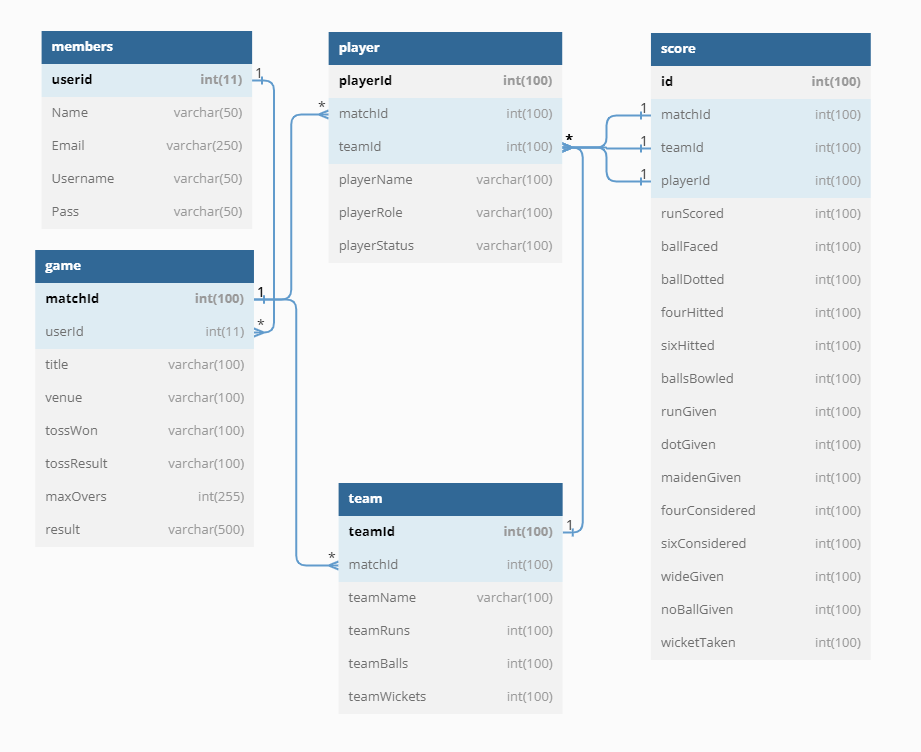
A database schema is the skeleton structure that represents the logical view of the entire database. It defines how the data is organised and how the relations among them are associated. It formulates all the constraints that are to be applied on the data.

**Symbol reference:** <https://www.lucidchart.com>[/](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 3.3 Schema Diagram Notations***

**Diagram:**



***Fig. 3.13 Schema Diagram***

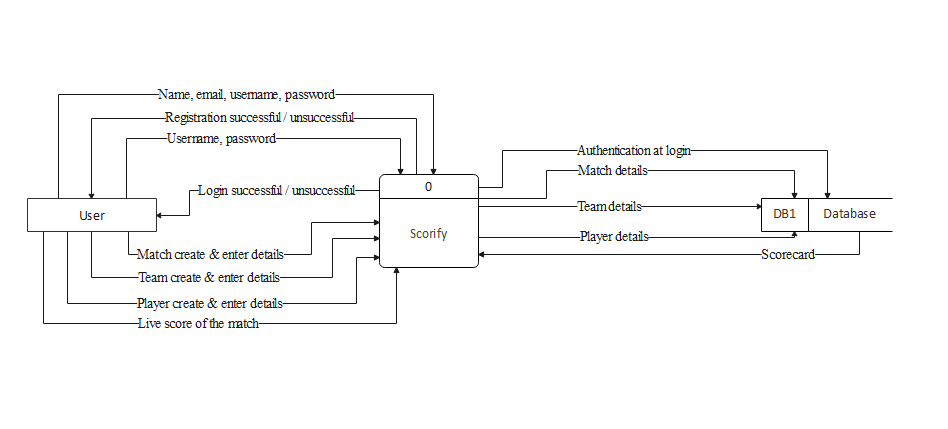
**3.5.3 Data Flow Diagram**

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/>

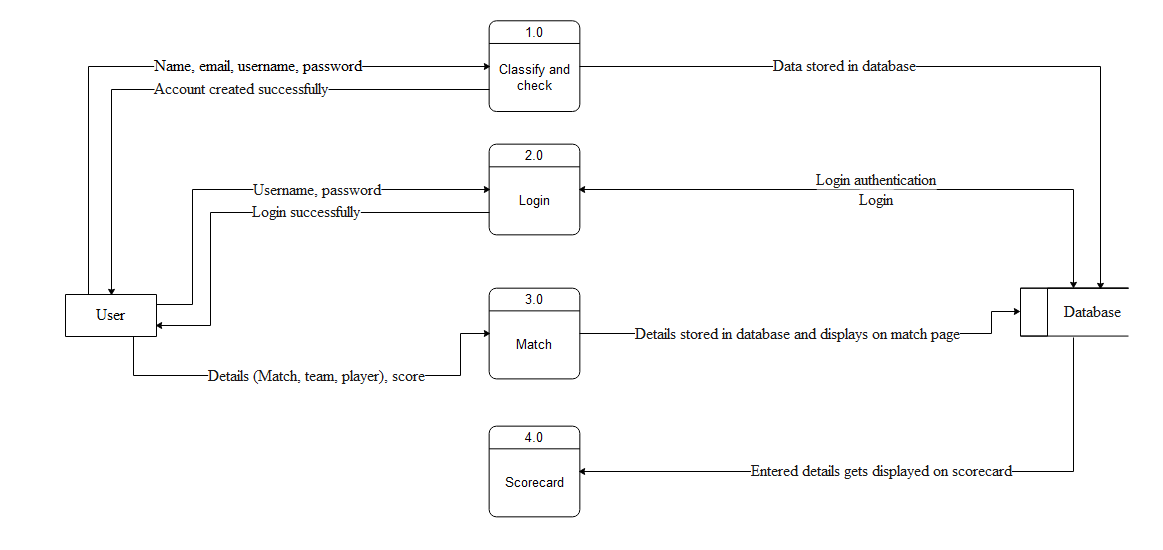
|  |  |  |
| --- | --- | --- |
| Name | Symbol | Description |
| Process |  | A process transforms incoming data flow into outgoing data flow. |
| Database |  | Data stores are repositories of data in the system. |
| Data Flow |  | Data flows are pipelines through which packets of information flow. Label 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 3.3 Data Flow Diagram Notations***

**Level 0 (Context Level DFD):**

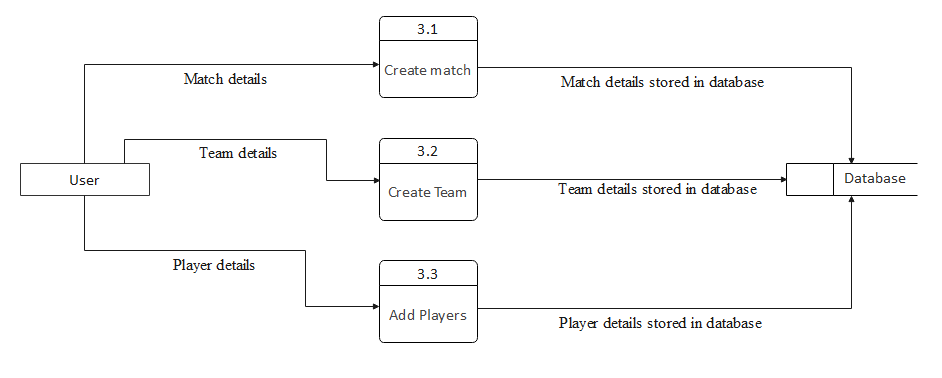
***Fig.3.14 Level 0 DFD***

**Level 1 DFD:**



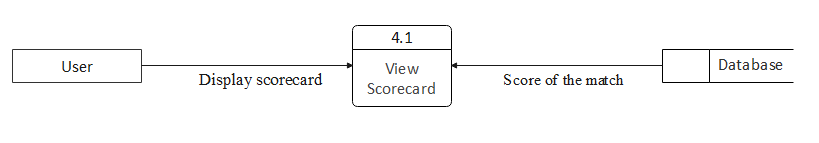
***Fig. 3.15 Level 1 DFD***

**Level 2 DFD for Match:**



***Fig. 3.16 Level 2 DFD for match***

**Level 2 DFD for Scorecard:**



***Fig. 3.17 Level 2 DFD for scorecard***

**3.5.4 Use Case Diagram**

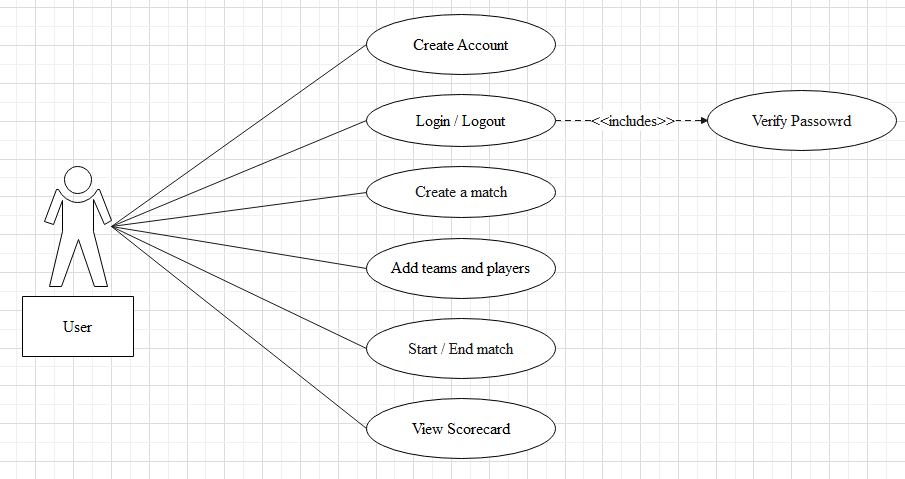
A use case diagram at its simplest is a representation of a user's interaction with the system that shows the relationship between the user and the different use cases in which the user is involved. A use case diagram can identify the different types of users of a system and the different use cases and 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 3.5 Use Case Notation***

**3.5.4.1 Diagram:**



***Fig. 3.18 Use Case Diagram***

**3.5.4.2 Use Case Diagram Description:**

1. Use Case: Register
   1. Description: The user needs to register their account. They fill the necessary details and can access their account using it.
   2. Actor: User.
   3. Pre-condition: User needs to full all the details.
   4. Exception: If the format of any detail is incorrect or any required field is kept empty, registration will not be successful.
   5. Post-condition: Registered successfully. Username and password provided.
2. Use Case: Login / Logout
   1. Description: The user can sign in or sign out of their accounts.
   2. Actor: User.
   3. Pre-condition: User needs to fill the correct login details to sign-in into their account.
   4. Post-condition: -
3. Use Case: Create a match
   1. Description: The user can create a new match for scoring the live game.
   2. Actor: User.
   3. Pre-condition: User must be logged in to their account.
   4. Post-condition: User will be able to create team, add player for created match.
4. Use Case: Add team and players
   1. Description: The user can add teams and players in the teams for created match.
   2. Actor: User.
   3. Pre-condition: User must have created a match.
   4. Post-condition: ­-
5. Use Case: Start or end match
   1. Description: The user will be able to start and end the created match.
   2. Actor: User.
   3. Pre-condition: User must have created a match, added teams and players.
   4. Post-condition: -
6. Use Case: View Scorecard
   1. Description: The user will be able to view the scorecard of the match.
   2. Actor: User.
   3. Pre-condition: The match should have been ended.
   4. Post-condition: -

**3.5.4 Sequence Diagram**

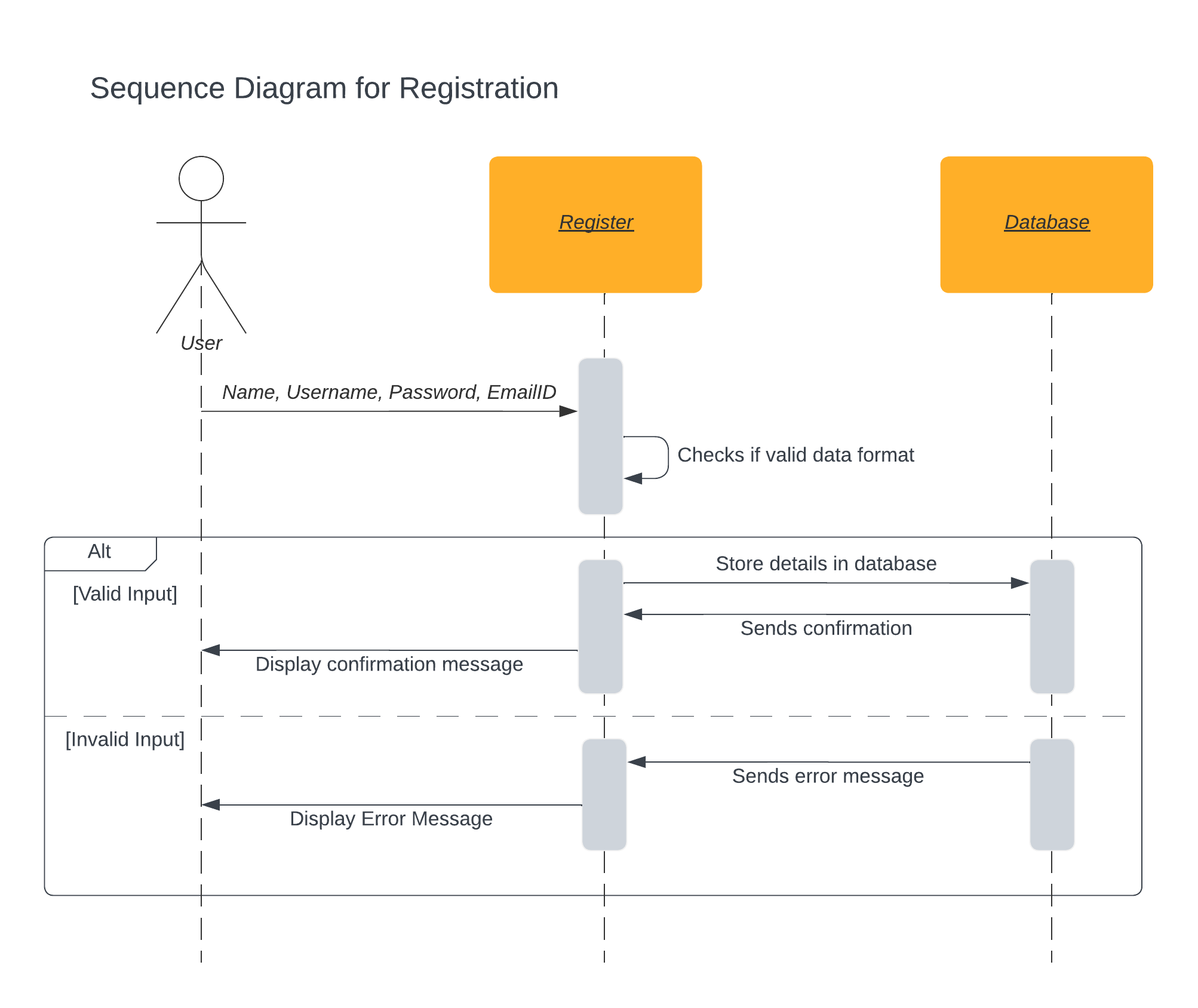
A sequence diagram in a Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence 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>[/](https://www.lucidchart.com/)

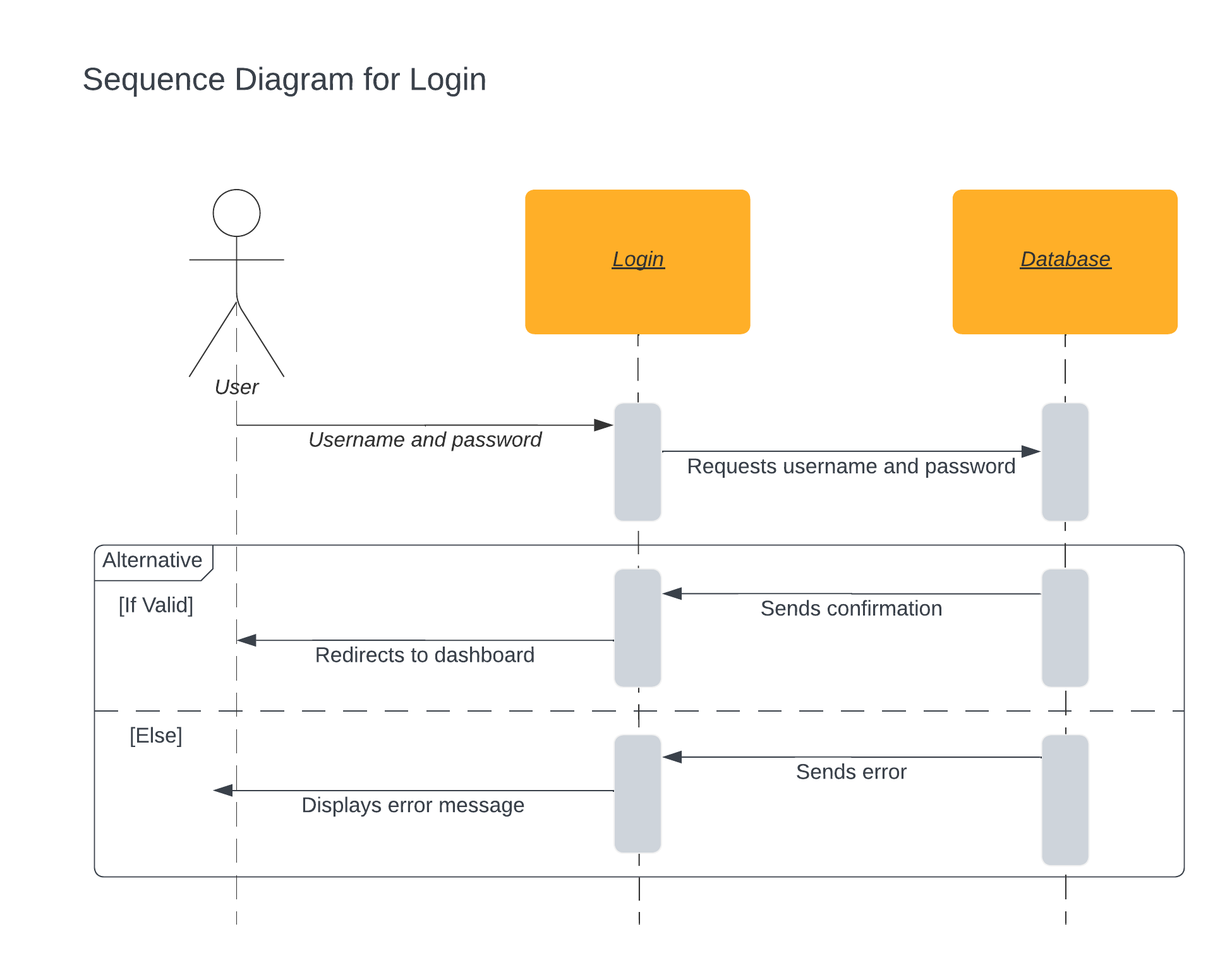
|  |  |  |
| --- | --- | --- |
| Name | Symbol | Description |
| Synchronous Message |  | An instantaneous communication between objects that conveys information, with the expectation that an action will be initiated as a result. |
| Activation Box |  | The period during which an object is performing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

***Table 4.5 Sequence Diagram Notation***

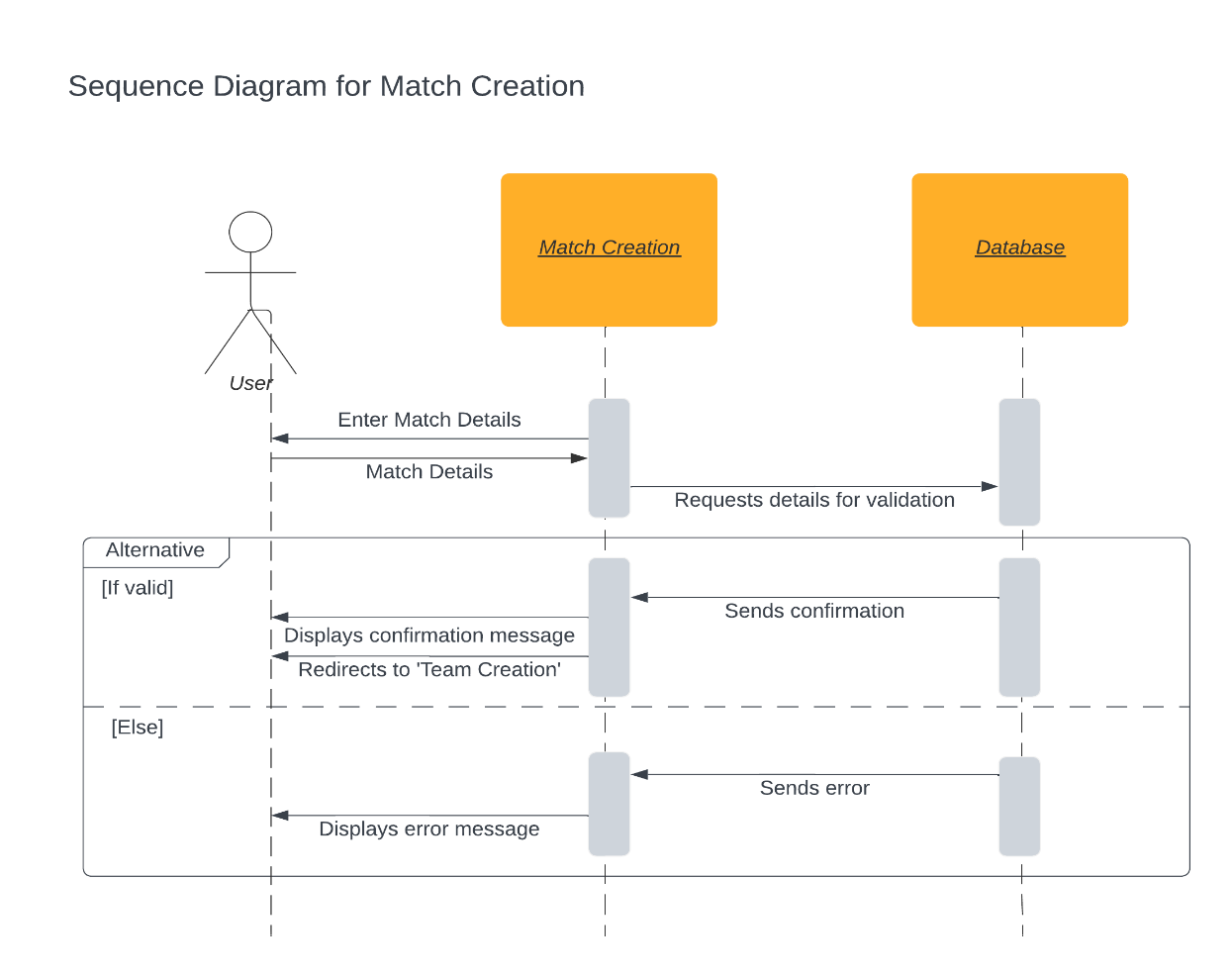
**Sequence Diagrams:**



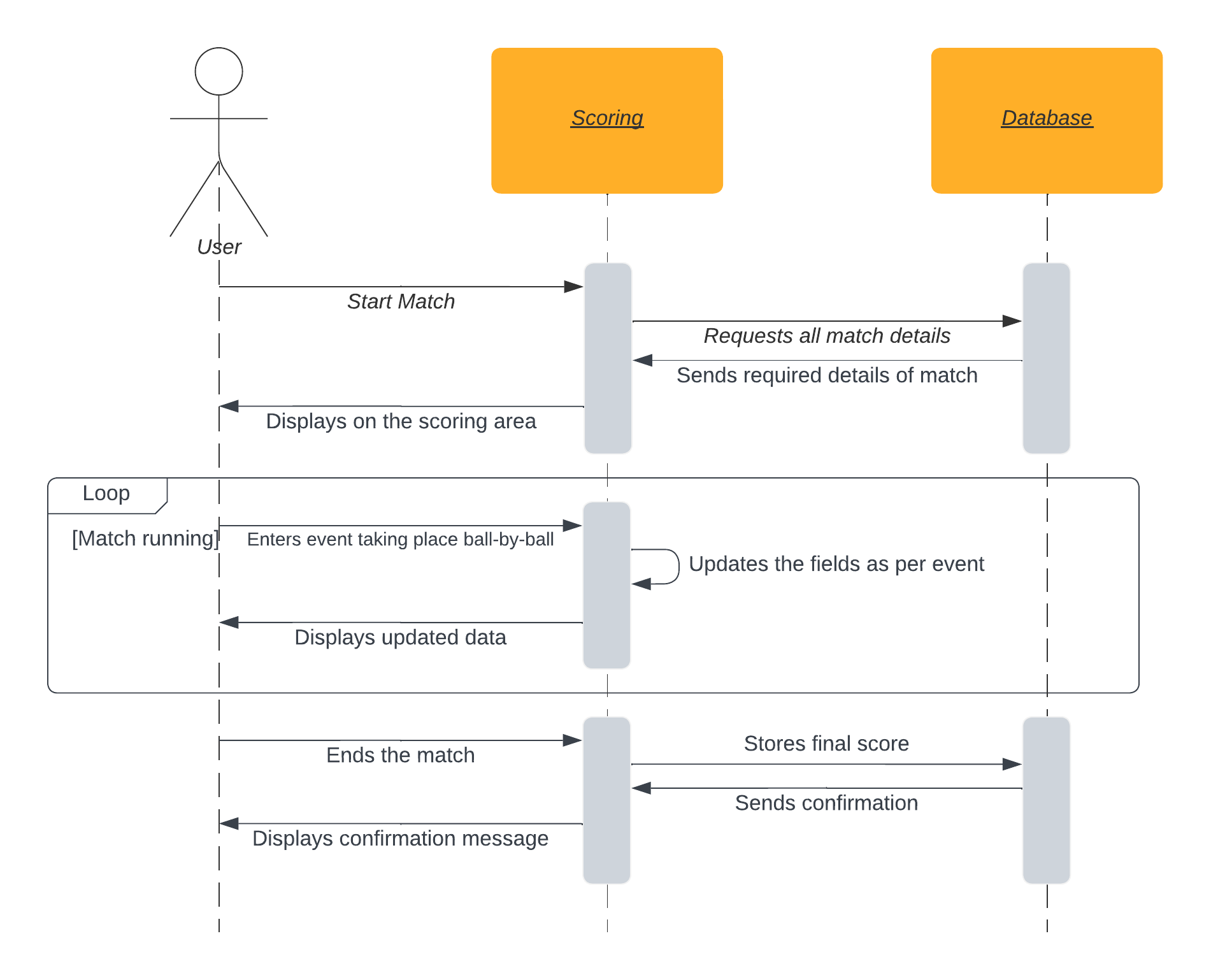
***Fig. 3.19 Sequence Diagram for Registration***



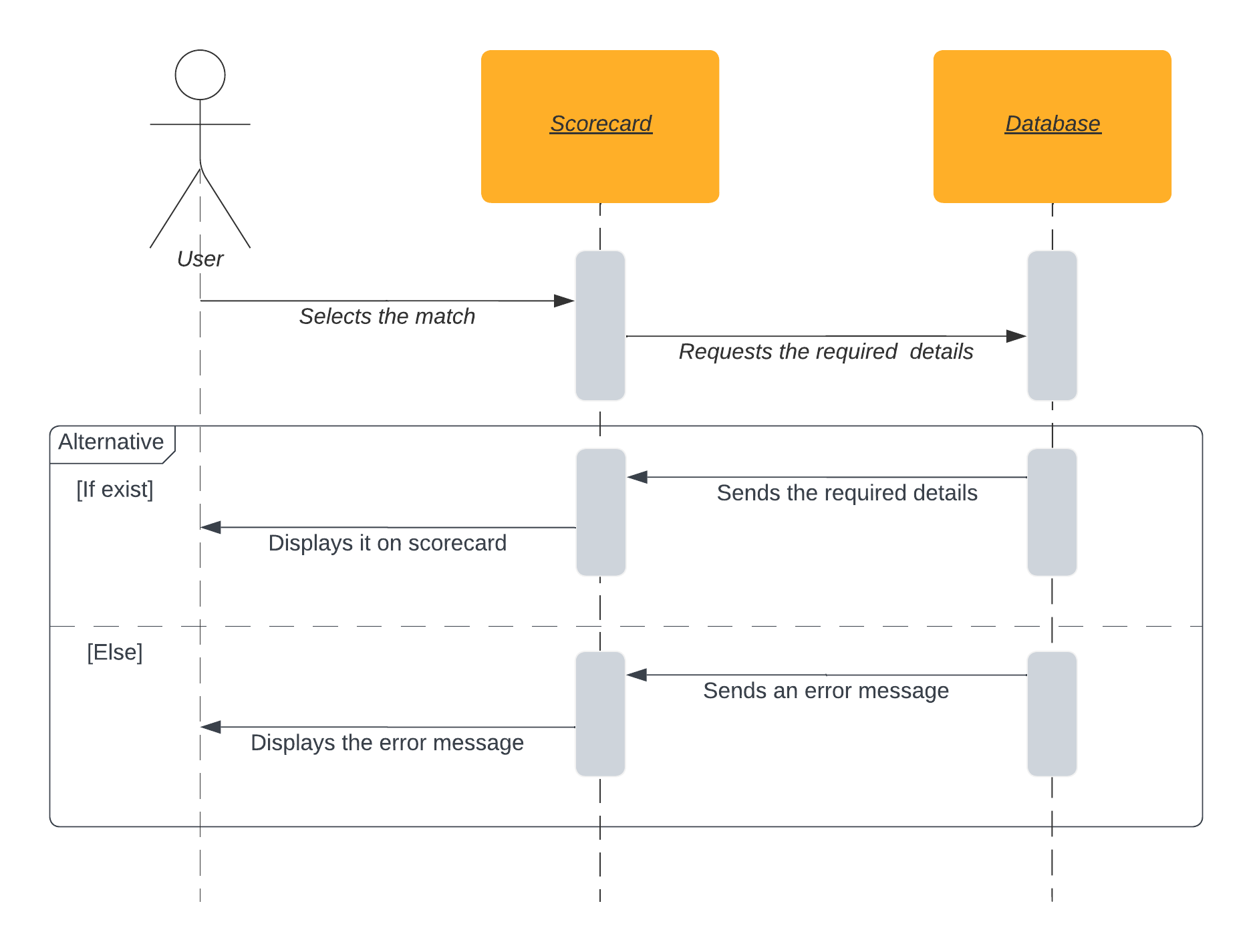
***Fig. 3.20 Sequence Diagram for Login***



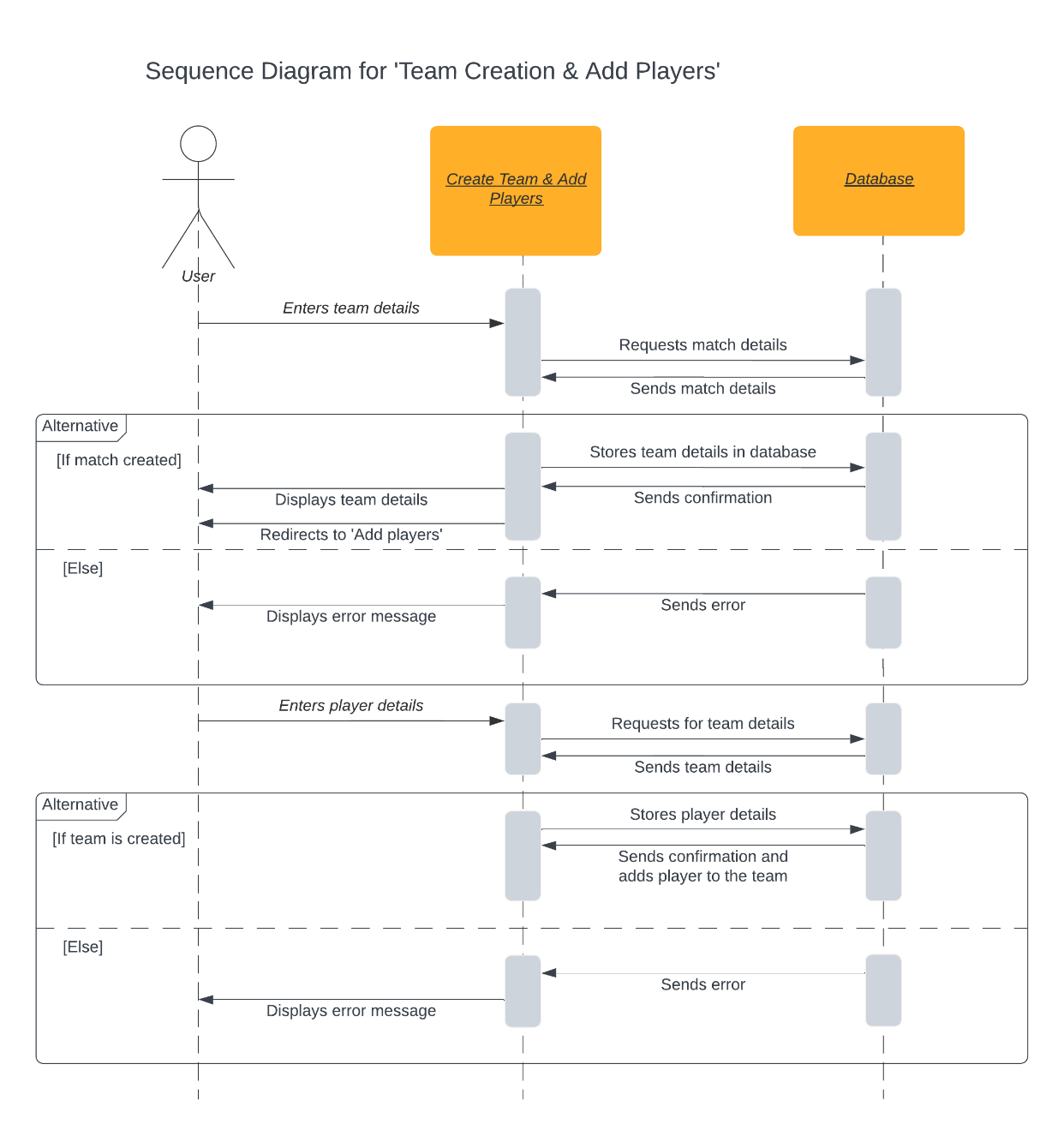
***Fig. 3.21 Sequence Diagram for Match Creation***



***Fig 3.22 Sequence diagram for Match Scoring***



***Fig. 3.23 Sequence diagram for scorecard***



***Fig. 3.24 Sequence diagram for Creation of teams and players***

**3.5.6 Activity Diagram**

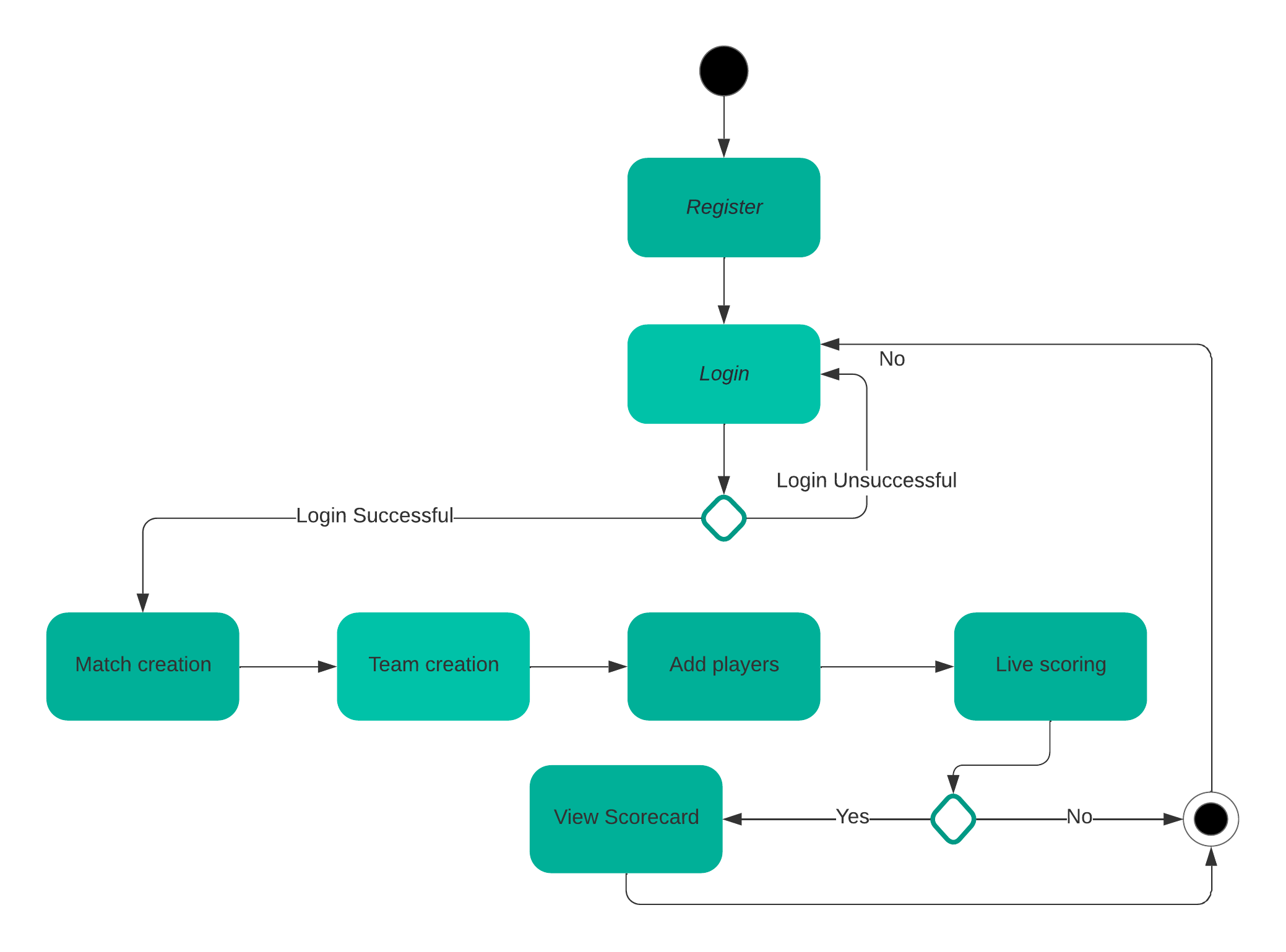
* Activity diagram is another important diagram in UML to describe the dynamic aspects of the system.
* Activity diagram is basically a flowchart to represent the flow from one activity to another activity.
* The activity can be described as an operation of the system. The control flow is drawn from 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>[/](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 3.7 Activity Diagram Notations***

**Diagram:**



***Fig 3.23 Activity Diagram***

**3.5.7 State-Chart Diagram**

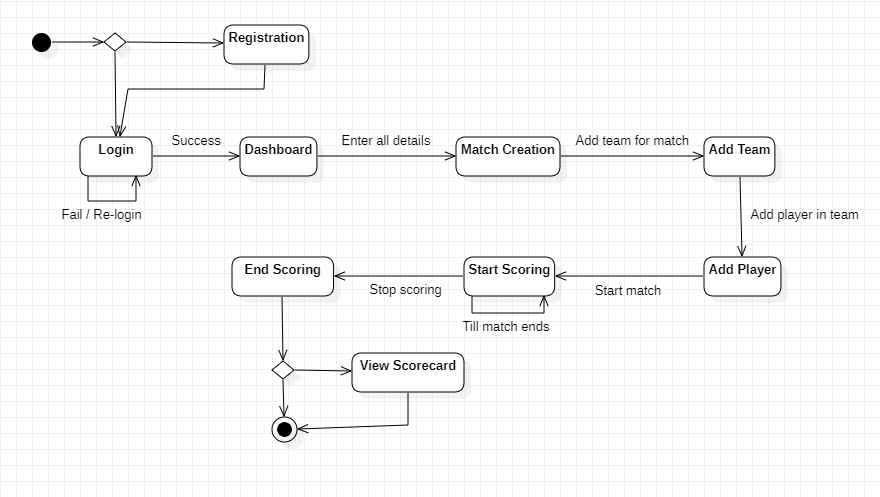
A state diagram is used to represent the condition of the system or part of the system at finite instances of time. It’s a behavioural diagram and it represents the behaviour using finite state transitions. State diagrams are also referred to as State machines and State-chart Diagrams. These terms are 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 3.8 State-Chart Notations***

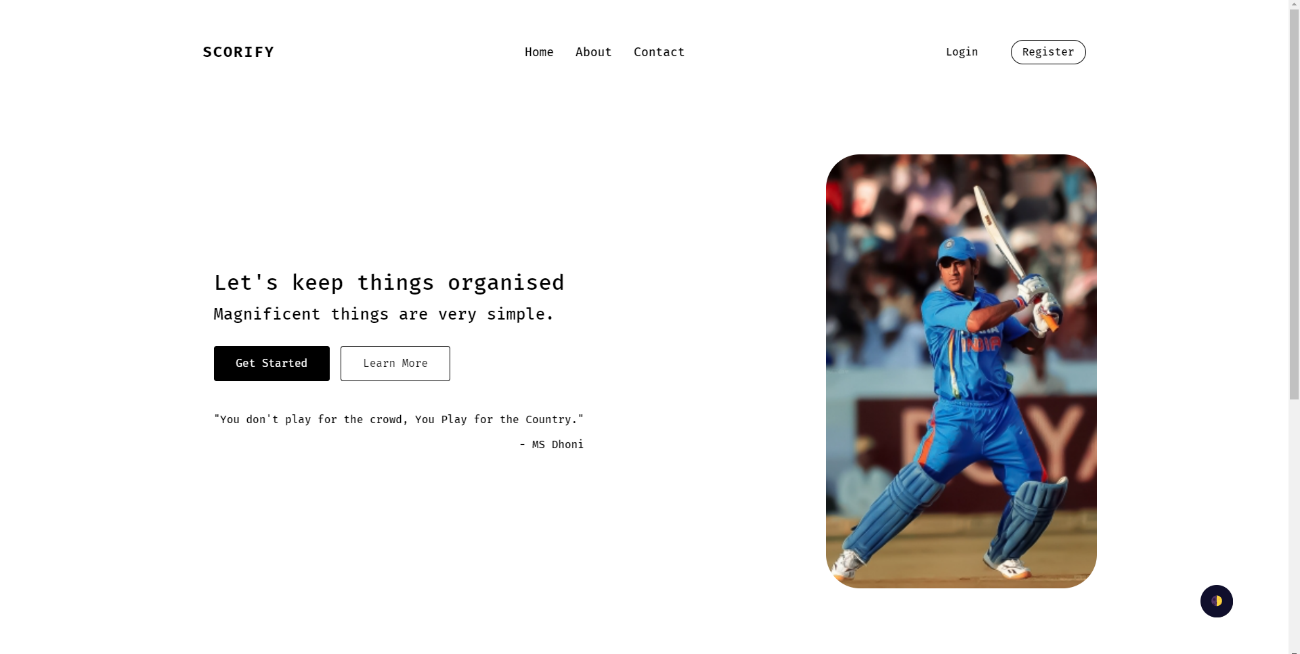
**Diagram:**



***Fig. 3.24 State-Chart Diagram***

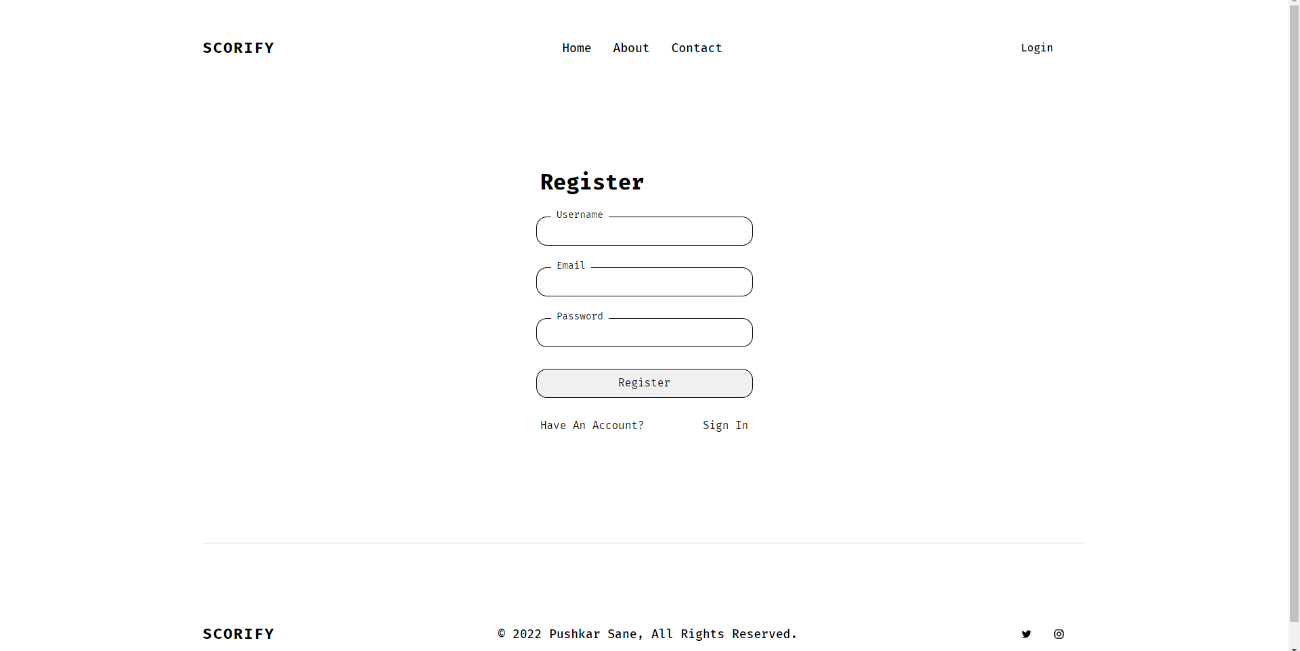
**4.1 User Interface**

1. Home Page



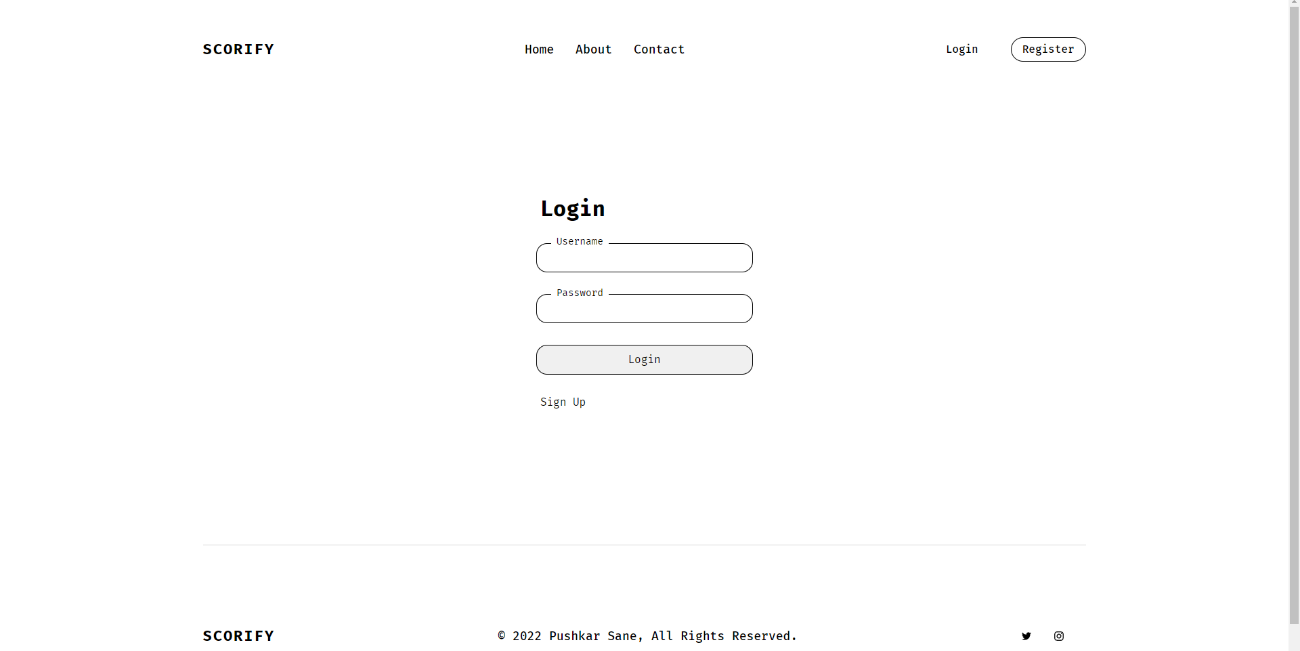
***Fig. 4.1 UI for Home Page***

1. Registration Page



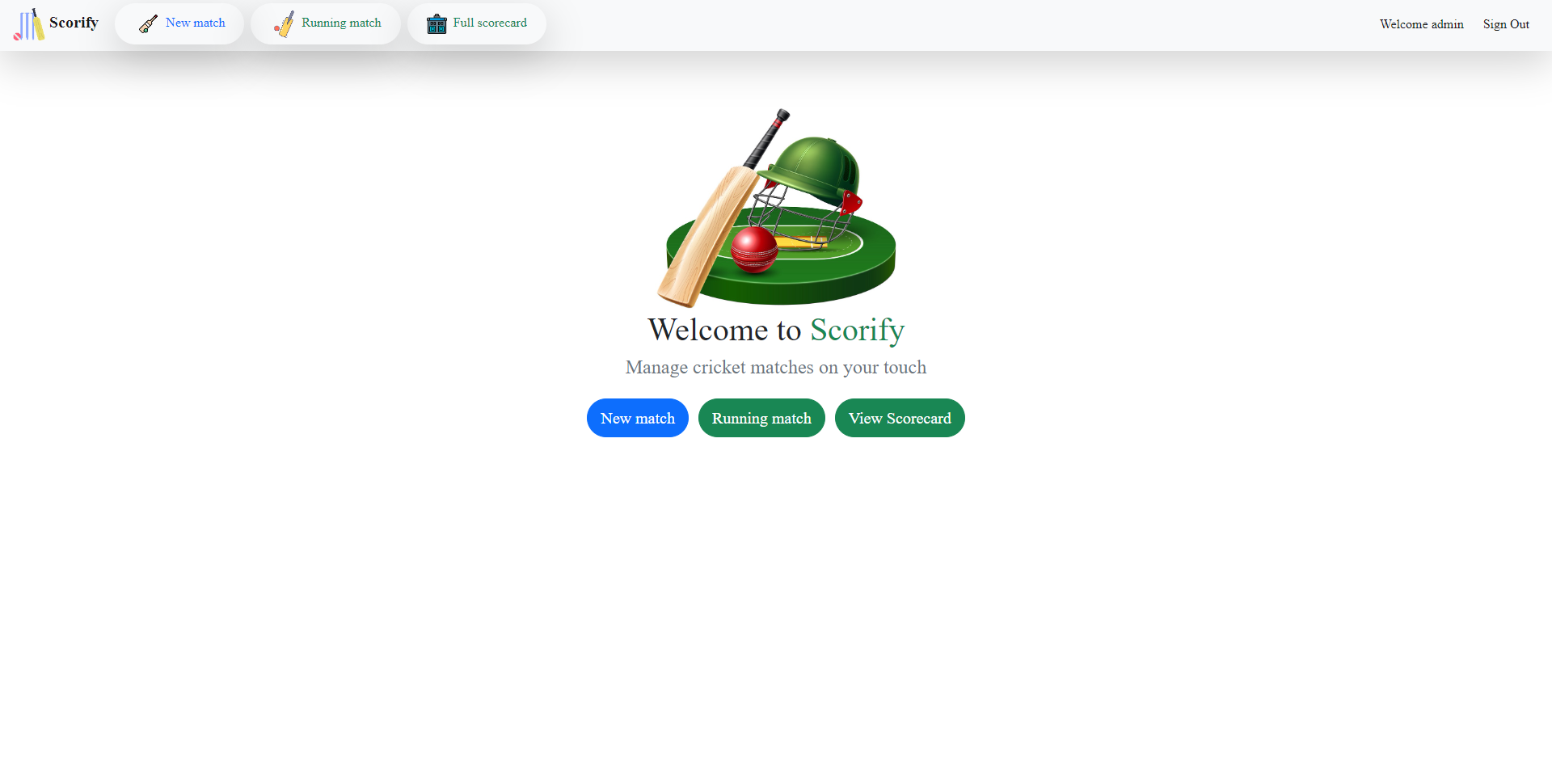
***Fig. 4.2 UI for Registration Page***

1. Login Page



***Fig. 4.3 UI for Login Page***

1. Dashboard



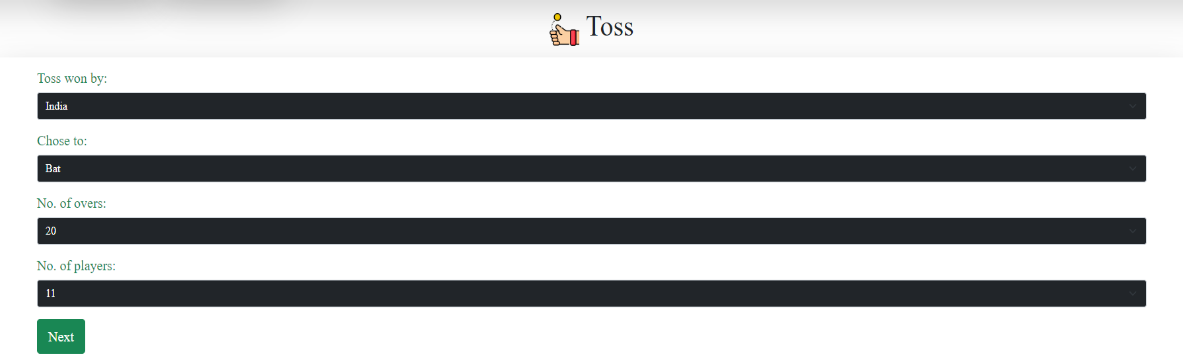
***Fig. 4.4 UI for Dashboard***

1. Creating Match



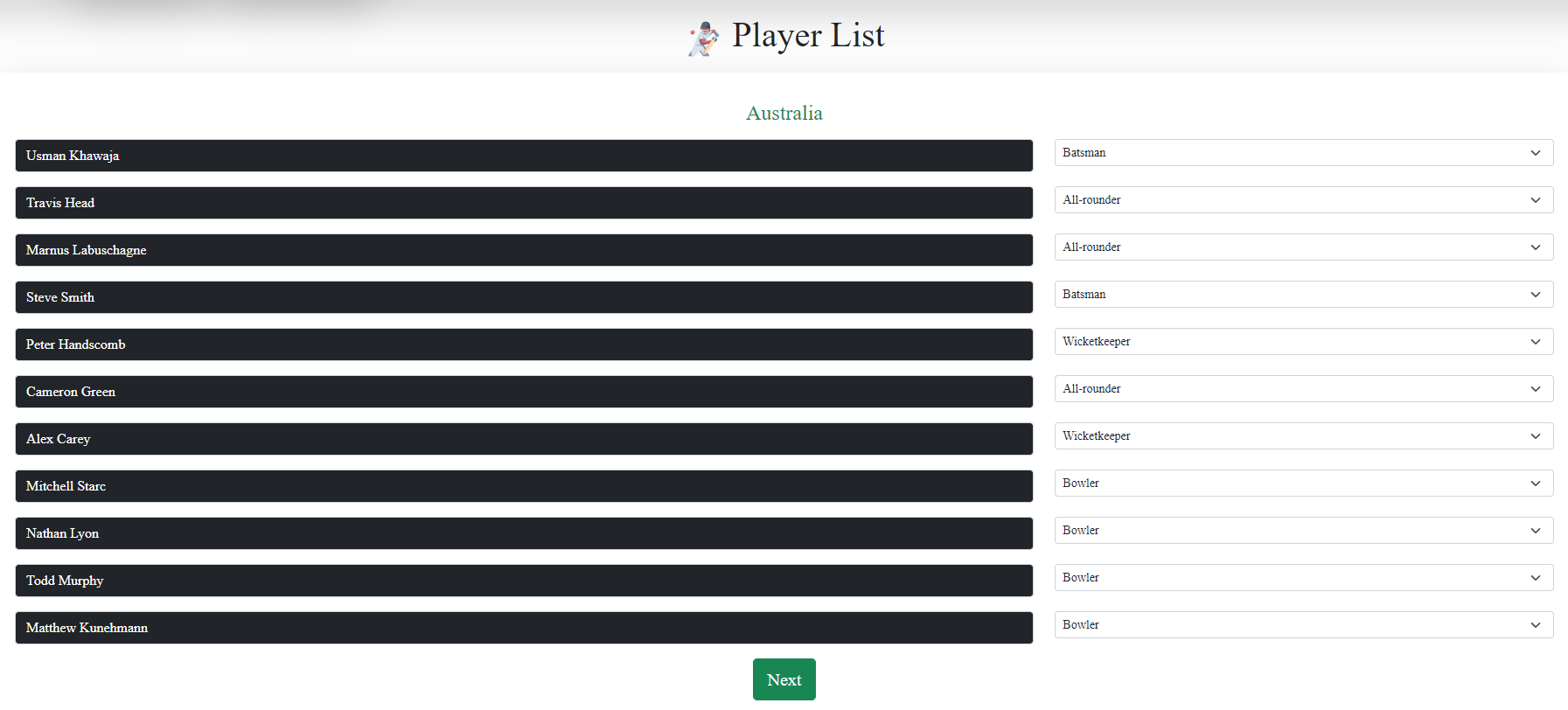
***Fig. 4.5 UI For Creating match, team and players***

1. Toss Details



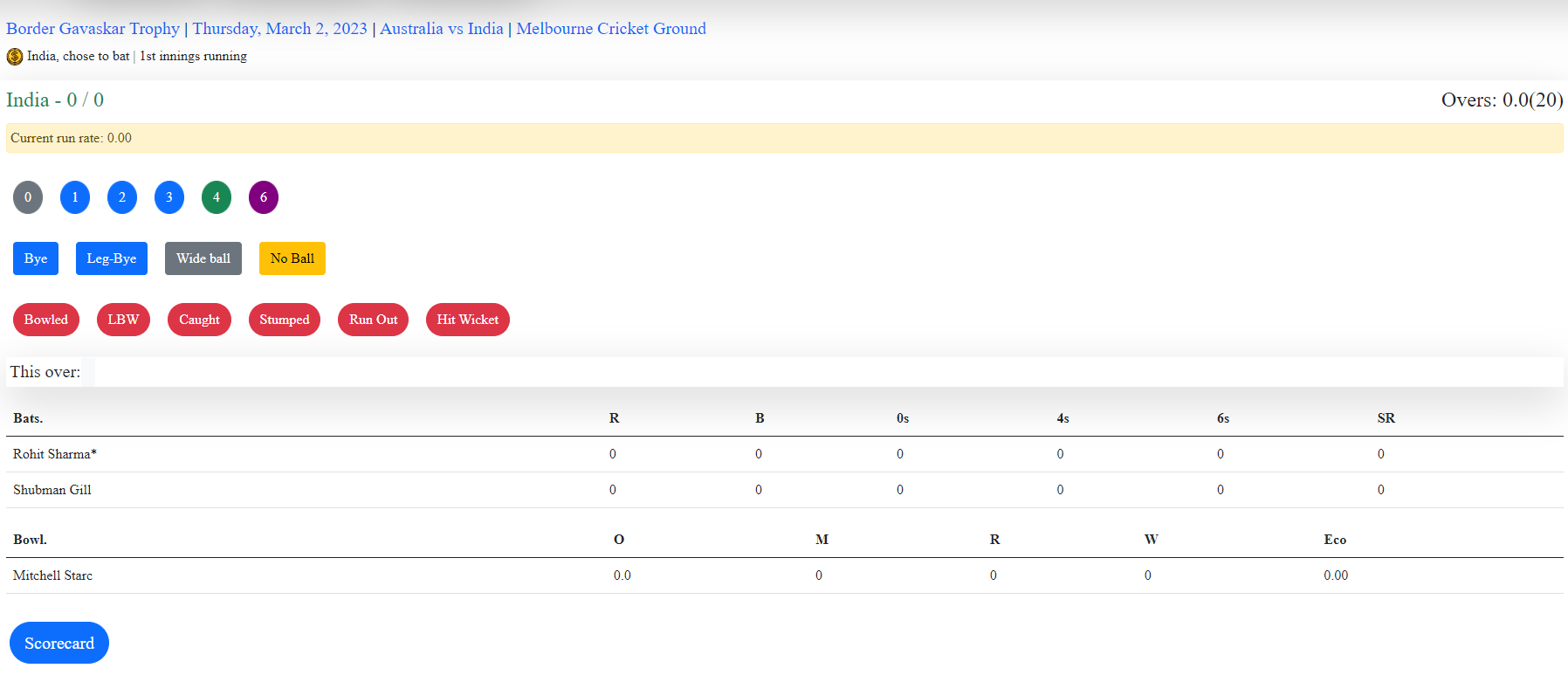
***Fig. 4.6 UI For Toss Details***

1. Add Player



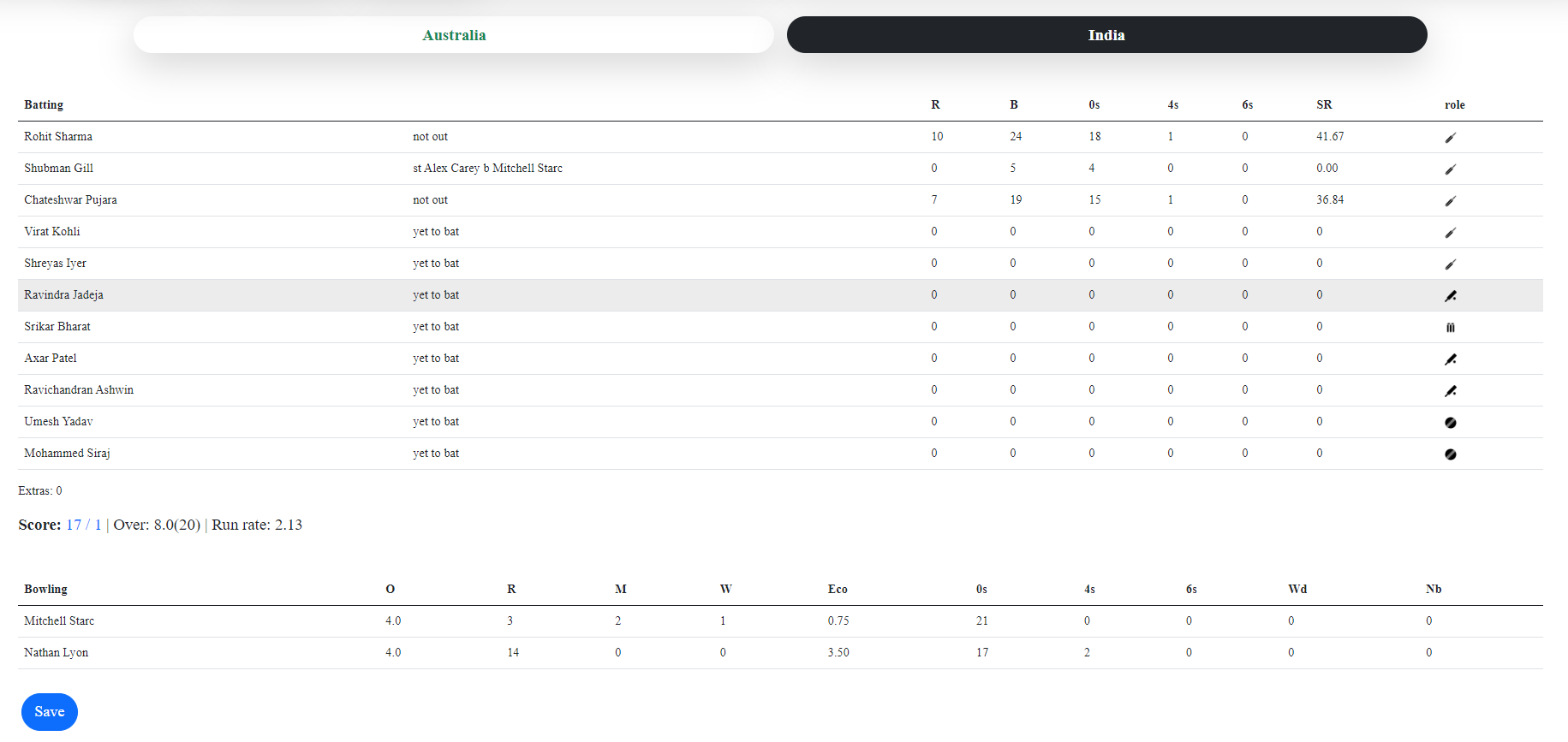
***Fig. 4.6 UI For Player List (Same for both teams)***

1. Live Scoring



***Fig. 4.6 UI for Live Scoring***

1. Scorecard



**Fig. 4.7 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 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 |
| 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 as well as bowler. | End inning of batsman, add 1 wicket in batting team. | Fail |
| 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***

**5.1 Implementation Approaches**

The Vaccine Management System (Le-Vaccine) 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>";

}

// close connection

mysqli\_close($con);

}

// login block ends here for cheching echo $status;

ob\_end\_flush();

?>

<!DOCTYPE html>

<html lang="en">

<head>

<title> Scorify </title>

<link rel="icon" type="image/x-icon" href="../imgs/vector-logo.png">

<meta property="og:title" content="Scorify" />

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

<meta charset="utf-8" />

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

<link rel="stylesheet" href="./css/register.css" />

</head>

<body>

<div>

<link href="./css/home.css" rel="stylesheet" />

<div class="contact-container">

<header data-role="Header" class="contact-header">

<p class="contact-text">

<a href="../index.php"> Scorify </a></p>

<div class="contact-nav">

<nav class="navigation-links2-nav navigation-links2-root-class-name10">

<span class="navigation-links2-text">

<a href="../index.php"> Home </a></span>

<span class="navigation-links2-text1">

<a href="./about.php"> About </a></span>

<span class="navigation-links2-text2">

<a href="./contact.php"> Contact </a></span>

</nav>

</div>

<div class="contact-btn-group">

<button class="contact-login button">

<a href="./login.php"> Login </a></button>

<button class="contact-register button">

<a href="./register.php"> Register </a></button>

</div>

<div data-role="BurgerMenu" class="contact-burger-menu">

<svg viewBox="0 0 1024 1024" class="contact-icon">

<path d="M128 554.667h768c23.552z"></path>

</svg>

</div>

<div data-role="MobileMenu" class="contact-mobile-menu">

<div class="contact-nav1">

<div class="contact-container1">

<span class="contact-text1">

<a href="../index.php"> Scorify </a></span>

<div data-role="CloseMobileMenu" class="contact-menu-close">

<svg viewBox="0 0 1024 1024" class="contact-icon02">

</svg>

</div>

</div>

<nav class="navigation-links2-nav navigation-links2-root-class-name11">

<span class="navigation-links2-text">

<a href="../index.php"> Home </a> </span>

<span class="navigation-links2-text1">

<a href="./about.php"> About </a> </span>

<span class="navigation-links2-text2"><span>

<a href="./contact.php"> Contact </a></span>

</span>

</nav>

<div class="contact-container2">

<button class="contact-button button">

<a href="./src/login.php"> Login </a></button>

<button class="contact-button1 button">

<a href="./src/register.php"> Register </a></button>

</div>

</div>

<div>

<a href=<https://www.twitter.com/SurturG> target="\_blank">

<svg viewBox="0 0 950.8571428571428 1024" class="contact-icon04">

<path d="M925.714 233.143c-25.143 36.571-56.571"></path>

</svg> </a>

<a href=<https://www.instagram.com/pushkar.png> target="\_blank">

<svg viewBox="0 0 877.7142857142857 1024" class="contact-icon06">

<path d="M585.143 512c0-80.571-65.714-146.286-146.286 146.286s"></path>

</svg>

</a>

</div>

</div>

</header>

<div class="height-set">

<form class="form" method="POST">

<div class="head">

<h1>Login</h1>

</div>

<p id="username-msg"></p>

<div class="input-contain">

<input type="text" name="username" id="username" autocomplete="on">

<label class="placeholder-text" for="username" id="placeholder-fname">

<div class="text">Username</div>

</label>

</div>

<p id="pass-msg"></p>

<div class="input-contain">

<input type="password" name="password" id="password" autocomplete="on">

<label class="placeholder-text" for="password" id="placeholder-fname">

<div class="text">Password</div>

</label>

</div>

<div class="div">

<input id="signup" type="submit" name="Login" value="Login">

</div>

<div class="sub">

<p> <a href="./register.php"> Sign Up </a></p>

</div>

</div>

<footer class="contact-footer">

<div class="contact-separator"></div>

</footer>

<footer class="contact-footer1">

<p class="contact-text2">

<a href="../index.php"> Scorify </a></p>

<span class="contact-text3">

© 2022 Pushkar Sane, All Rights Reserved.

</span>

<div class="contact-icon-group1">

<a href=<https://www.twitter.com/SurturG> target="\_blank" rel="noreferrer noopener" class="contact-link2">

<svg viewBox="0 0 950.8571428571428 1024" class="contact-icon08">

<path d="M925.714 233.143c-25.143 "></path>

</svg>

</a>

<a href=<https://www.instagram.com/pushkar.png> target="\_blank" rel="noreferrer noopener" class="contact-link3">

<svg viewBox="0 0 877.7142857142857 1024" class="contact-icon10">

<path d="M585.143512c0-80.571-65"></path>

</svg>

</a>

</div>

</footer>

</div>

</div>

<script data-section-id="navbar" src="https://unpkg.com/@teleporthq/teleport-custom-scripts"></script>

</body>

</html>

**Dashboard.php**

<?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();

?>

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

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

<title> Scorify </title>

<link rel="icon" href="assets/images/logo.png">

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

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

</head>

<body>

<link rel="stylesheet" href="../css/home.css">

<nav class="navbar sticky-top navbar-expand-lg navbar-light bg-light px-3 shadow shadow-lg">

<a class="navbar-brand" href="javascript:void(0)" onclick="loadHome()">

<span class="fw-bold">

<img src="assets/images/logo.png" width="40" height="40" alt=""> Scorify

</span></a>

<a href="javascript:void(0)" class="h5 d-block d-lg-none" data-bs-toggle="collapse" data-bs-target="#nbcollapse">

<img src="assets/images/toggler.png" width="25" height="25" alt="">

</a>

<div class="collapse navbar-collapse" id="nbcollapse">

<ul class="navbar-nav my-2 my-lg-0">

<li class="nav-item mx-auto mx-lg-1 my-1 my-lg-0 shadow shadow-lg px-3 rounded-pill">

<a class="nav-link active rounded-pill text-primary" aria-current="page" href="javascript:void(0)" onclick="newMatch()">

<span>

<img src="assets/images/new-match.png" width="35" height="35" alt="">

</span>

<span> New match</span>

</a>

</li>

<li id="running-match-nav" class="d-none nav-item mx-auto mx-lg-1 my-1 my-lg-0 shadow shadow-lg px-3 rounded-pill">

<a class="nav-link active rounded-pill text-success" aria-current="page" href="javascript:void(0)"

onclick="runningMatch()">

<span>

<img src="assets/images/running-match.png" width="35" height="35" alt="">

</span>

<span> Running match</span>

</a>

</li>

<li id="score-nav" class="d-none nav-item mx-auto mx-lg-1 my-1 my-lg-0 shadow shadow-lg px-3 pt-lg-1 rounded-pill">

<a class="nav-link active rounded-pill text-success" aria-current="page" href="javascript:void(0)" onclick="teamFullCard(0)">

<span>

<img src="assets/images/scoreboard.png" width="25" height="25" alt="">

</span>

<span> &nbspFull scorecard</span>

</a>

</li>

</ul>

<ul class="navbar-nav ms-auto">

<li class="nav-item mx-1">

<a class="nav-link active"> Welcome <?php echo $login\_session; ?> </a>

</li>

<li class="nav-item mx-1">

<a class="nav-link active" href="../login.php">Sign Out</a>

</li>

</ul>

</div>

</nav>

<div class="container-fluid" id="main-container"></div>

<div class="modal fade" id="error-modal" tabindex="-1">

<div class="modal-dialog">

<div class="modal-content">

<div class="modal-header">

<h5 class="modal-title" id="exampleModalLabel">Error Message</h5>

<button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Close"></button>

</div>

<div class="modal-body">

<div id="error-msg"></div>

</div>

</div>

</div>

</div>

<div class="modal fade" id="new-match-modal" tabindex="-1">

<div class="modal-dialog modal-dialog-centered">

<div class="modal-content">

<div class="modal-header">

<h5 class="modal-title" id="exampleModalLabel">Confirmation Message</h5>

<button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Close"></button>

</div>

<div class="modal-body">

A match is still running, Are you sure want to discard this match and start a new one?

</div>

<div class="modal-footer">

<button class="btn btn-primary" id="new-match-btn" data-bs-dismiss="modal">Yes</button>

</div>

</div>

</div>

</div>

</div>

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

<script src="../js/route.js"></script>

<script src="../js/setup.js"></script>

<script src="../js/scoreboard.js"></script>

<script src="../js/run.js"></script>

<script src="../js/wicket.js"></script>

<script src="../js/run\_out.js"></script>

<script src="../js/wide\_ball.js"></script>

<script src="../js/no\_ball.js"></script>

<script data-section-id="navbar" src="https://unpkg.com/@teleporthq/teleport-custom-scripts"></script>

<script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.6.3/jquery.min.js" integrity="sha512-STof4xm1wgkfm7heWqFJVn58Hm3EtS31XFaagaa8VMReCXAkQnJZ+jEy8PCC/iT18dFy95WcExNHFTqLyp72eQ==" crossorigin="anonymous" referrerpolicy="no-referrer"></script>

</body>

</html>

**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) => {

let match = $("#match").val();

//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,

},

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,

},

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,

},

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,

},

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)**

//Hidden Fields Data Bridge

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 = [];

$("#savedata").click(() => {

let match = JSON.parse(localStorage.getItem("match"));

let realmatch = match;

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));

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];

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));

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>

<!--Team-1 Scorecard-->

<div class="mt-5 row d-flex justify-content-center mx-1 mx-lg-5 ">

<div class="col-12 my-lg-0 col-lg-5 text-center shadow shadow-lg fw-bold mx-2 rounded-pill" id="teamOneCard" style="font-size: 1.3rem;">

<span id="teamOneName" class="nav-link text-center active rounded-pill text-success" ></span>

</div>

<div class="my-5 mx-lg-5 px-2">

<table class="table table-hover">

<thead>

<tr class=" fw-bold">

<th>Batting</th>

<th></th>

<th>R</th>

<th>B</th>

<th>0s</th>

<th>4s</th>

<th>6s</th>

</tr>

</thead>

<tbody id="battingCard1"> </tbody>

</table>

<div id="extraRuns"></div>

<div class="my-3" id="scoreRateOver" style="font-size: 1.3rem;"> </div>

<table class="table table-hover mt-5" id="bowlingTable1">

<thead>

<tr class="fw-bold">

<th>Bowling</th>

<th>O</th>

<th>R</th>

<th>M</th>

<th>W</th>

<th>Wd</th>

<th>Nb</th>

</tr>

</thead>

<tbody id="bowlingCard1"></tbody>

</table>

</div>

<!--Team-2 Scorecard-->

<div class="col-11 mt-3 my-lg-0 col-lg-5 text-center shadow shadow-lg fw-bold mx-2 rounded-pill" id="teamTwoCard" style="font-size: 1.3rem;">

<p id="teamTwoName" class="nav-link text-center active rounded-pill text-success" aria-current="page"> </p>

</div>

<div class="my-5 mx-lg-5 px-2">

<table class="table table-hover">

<thead>

<tr class=" fw-bold">

<th>Batting</th>

<th></th>

<th>R</th>

<th>B</th>

<th>0s</th>

<th>4s</th>

<th>6s</th>

</tr>

</thead>

<tbody id="battingCard2"> </tbody>

</table>

<div id="extraRuns"></div>

<div class="my-3" id="scoreRateOver" style="font-size: 1.3rem;"> </div>

<table class="table table-hover mt-5" id="bowlingTable1">

<thead>

<tr class="fw-bold">

<th>Bowlling</th>

<th>O</th>

<th>R</th>

<th>M</th>

<th>W</th>

<th>Wd</th>

<th>Nb</th>

</tr>

</thead>

<tbody id="bowlingCard2"></tbody>

</table>

</div>

</div>

**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.

**5.3.1 Unit Testing**

These Tests are designed to ensure that each unit performs as expected and produces the correct outputs for various inputs or scenarios.

**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 5.2 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 team and +1 in extras. | Add 1 run in batting team and +1 in extras. | Pass |

***Table 5.3 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. | Adds 1 run in batting team and adds 1 ball | 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 5.4 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 |

**Following changes were done to solve the errors to fix the issues:**

//Test Case Fixes

**6.1 Test Report**

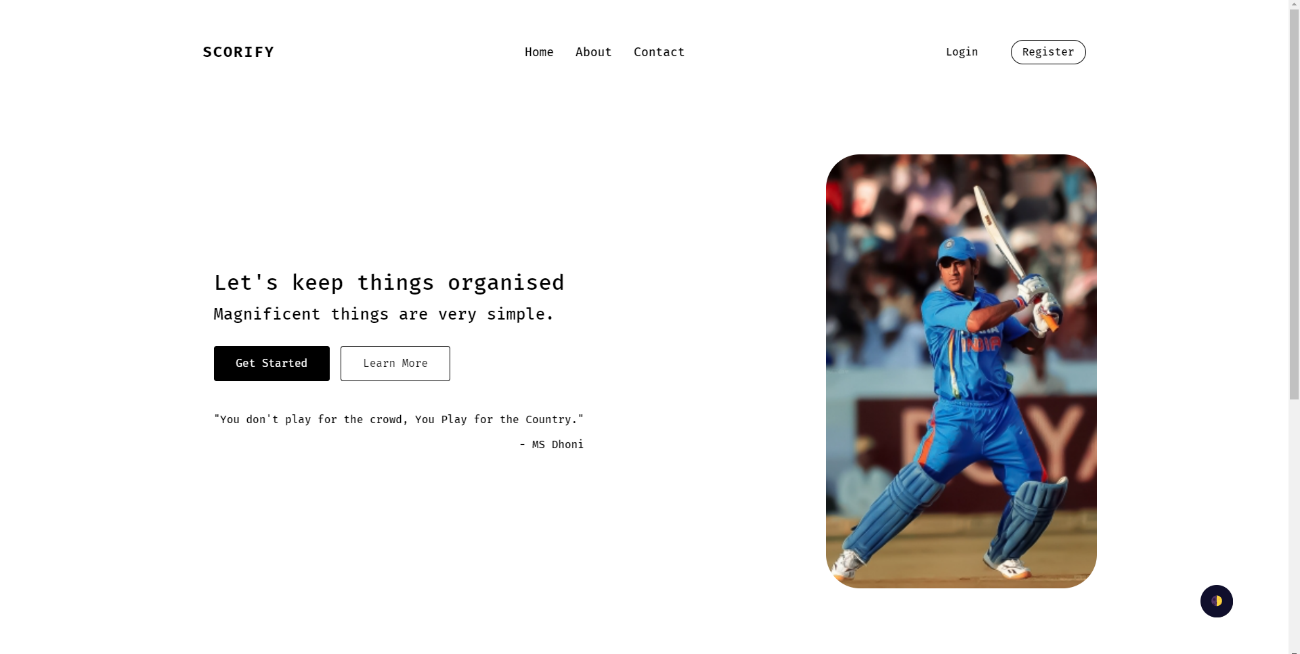
The testing phase of project development is critical. The testing step allows you to determine whether all of the capabilities are being executed correctly. The testing phase began with the creation of test cases for each module as well as the design of the modules themselves. The process for integrating 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

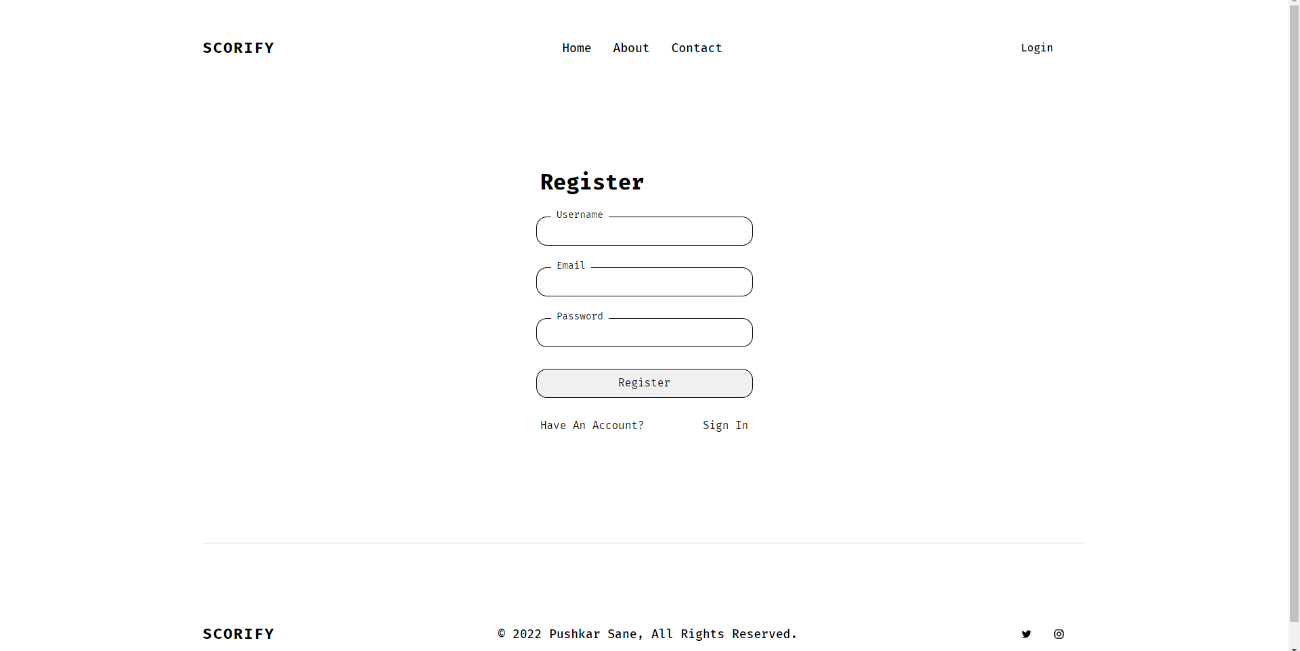
The First Page the user can see when they visit the website.



***Table 6.1 Home Page***

* Register

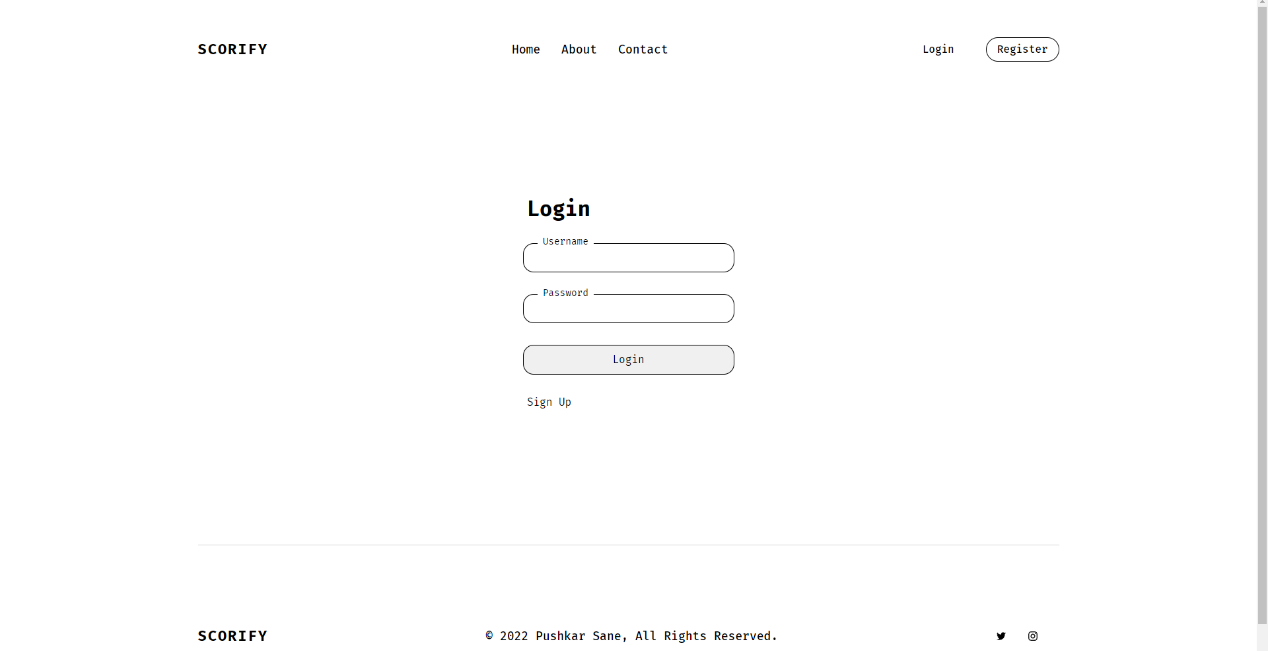
The user can create their account from here which will be used for logging in to the system.



***Table 6.2 Registration Page***

* Login Page

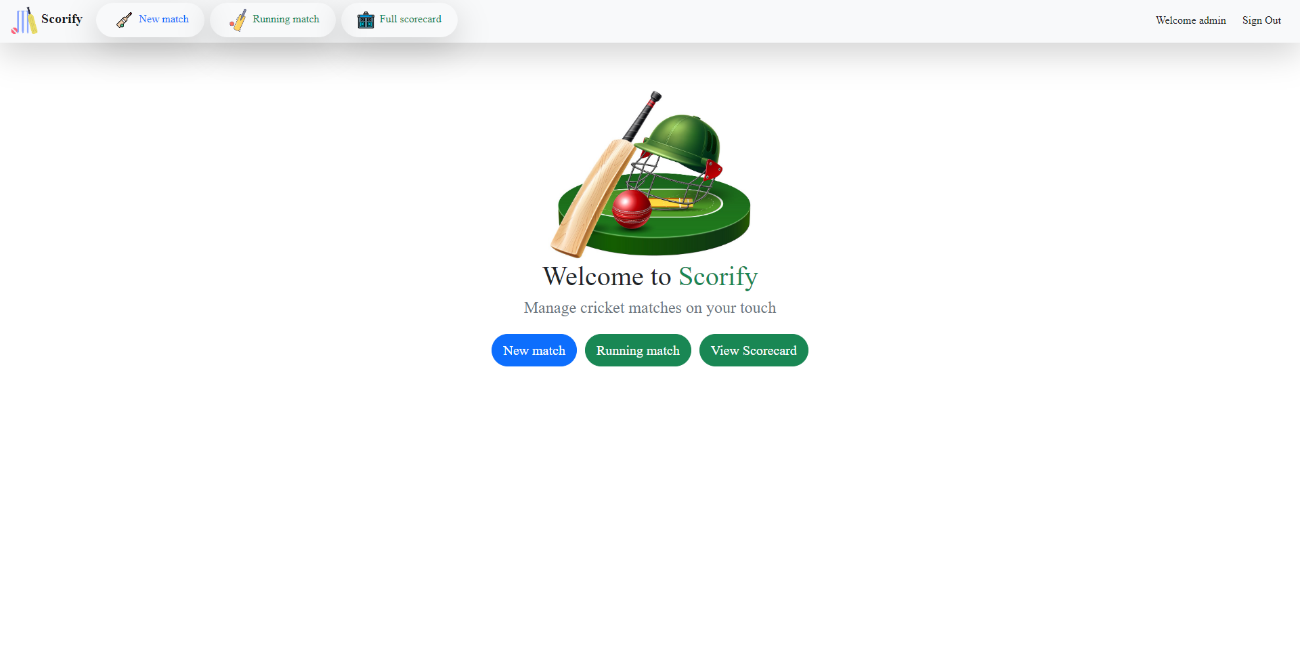
This is login page that will allow registered users to login themselves in to the system and use the system.



***Table 6.2 Login Page***

* Dashboard

This is the first page the user sees after logging in



**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 manage vaccine distribution and administration.

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 can’t be used to score Test / Multi-Day matches.
* Once the score is recorded it cannot be reversed.
* This Project is only capable to Handle Moderate Traffic as the hosting solution is based on n 1 CPU core and 2GB RAM. (Financial Issue)

**7.3 Future Scope of the System**

* Detailed, organized scorecard.
* Display scores of matches for general people (Cricbuzz).
* Fully responsive design.

**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](https://www.youtube.com/watch?v=WnMQ8HlmeXc)
* **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 Edition Robin Nixon, O’Reilly
  + Fundamentals of Creating a Great UI/UX Author: - Creative Tim as of 23/08/22