Table Of Contents:

List Of Figures:

List Of Tables:

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.

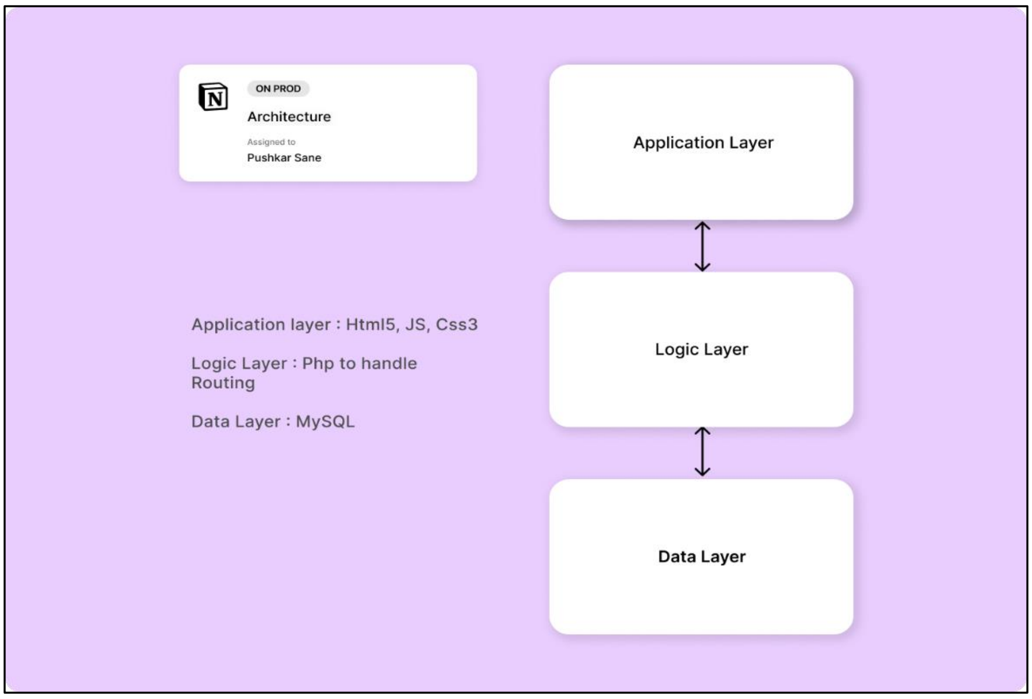
- To reduce multiple manual calculations during the match such as net run rate, batsman's analysis, bowler's analysis.
- To provide ball-by-ball update of the match.
- To provide detailed scorecard.

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



Application layer : Html5, JS, Css3

Logic Layer : Php to handle Routing

Data Layer : MySQL

Figure 1.1 Architecture

Requirements

Software Requirements

- Front-end: HTML5, CSS,JS, Tailwinds
- 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.

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.

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 afast, 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.

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 and download 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.

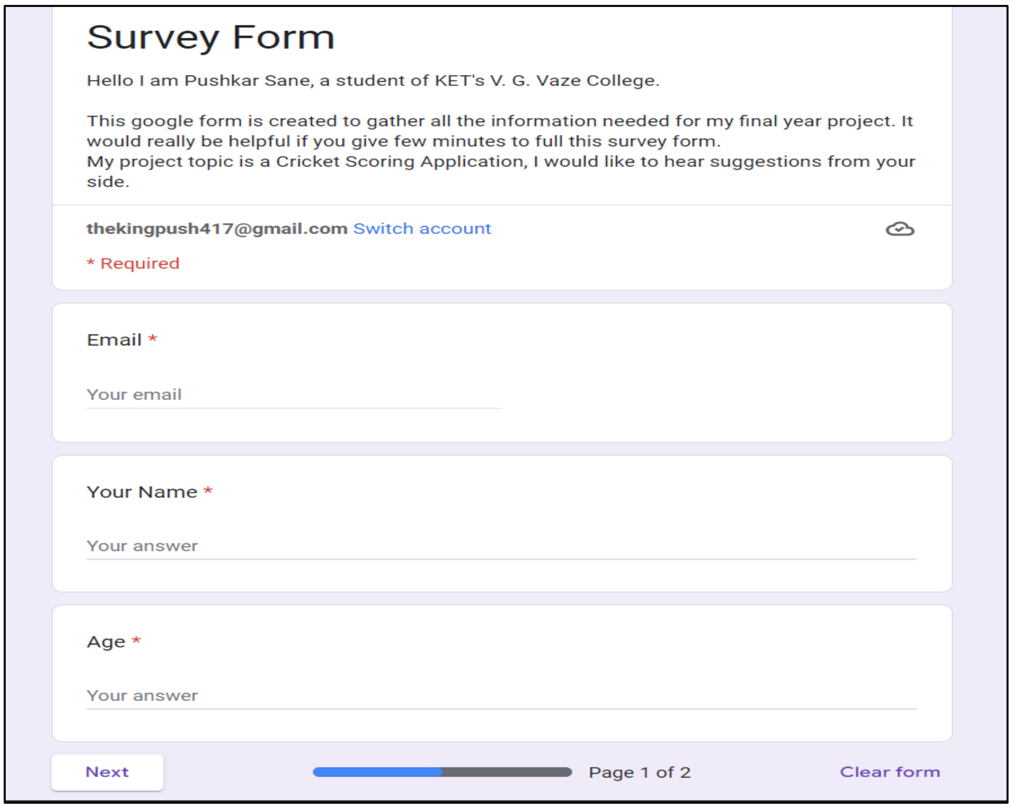
- Capture requirements
- Holding One-One interviews
- Conduct Group Workshops
- Get Feedbacks
- Build Small Prototypes

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

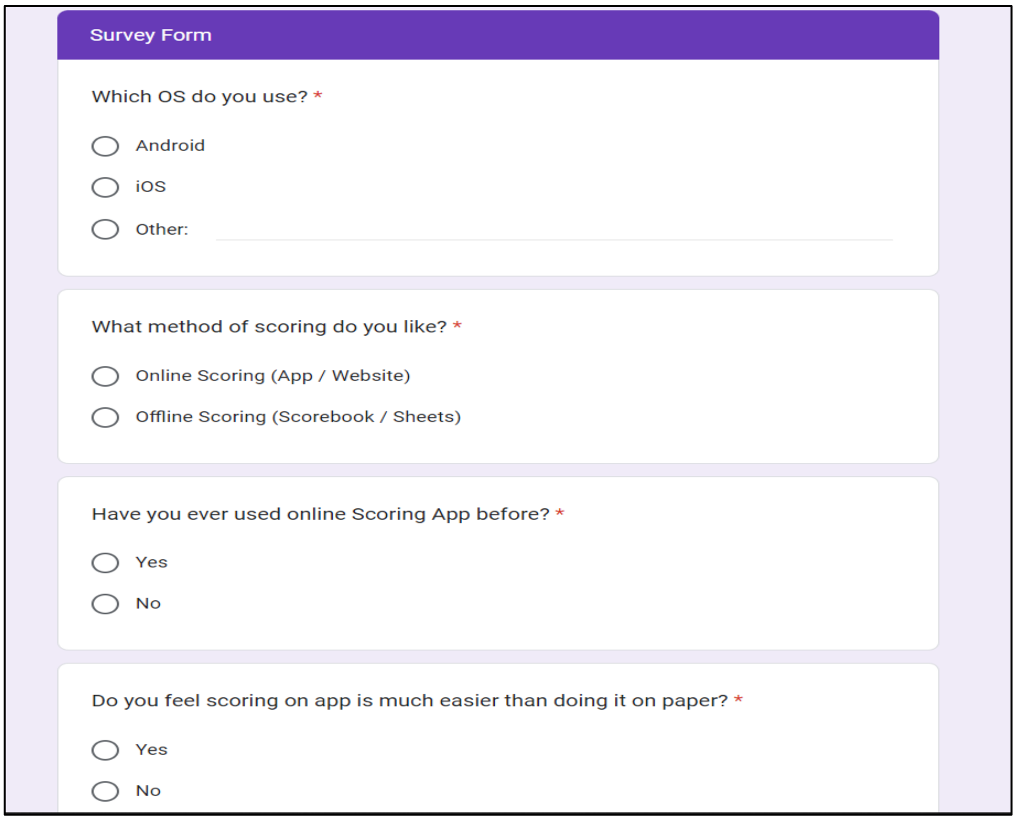
Your Name *

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? *

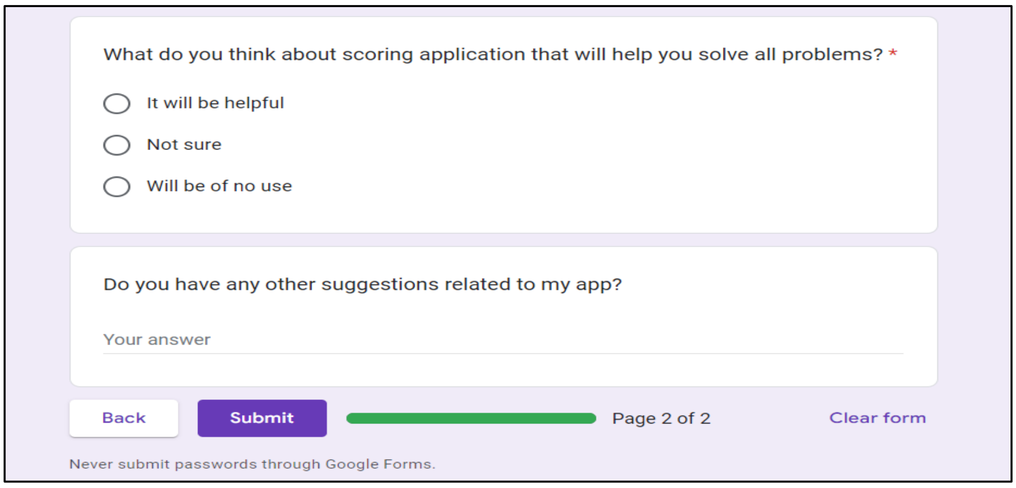
- Online Scoring (App / Website)
- Offline Scoring (Scorebook / Sheets)

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

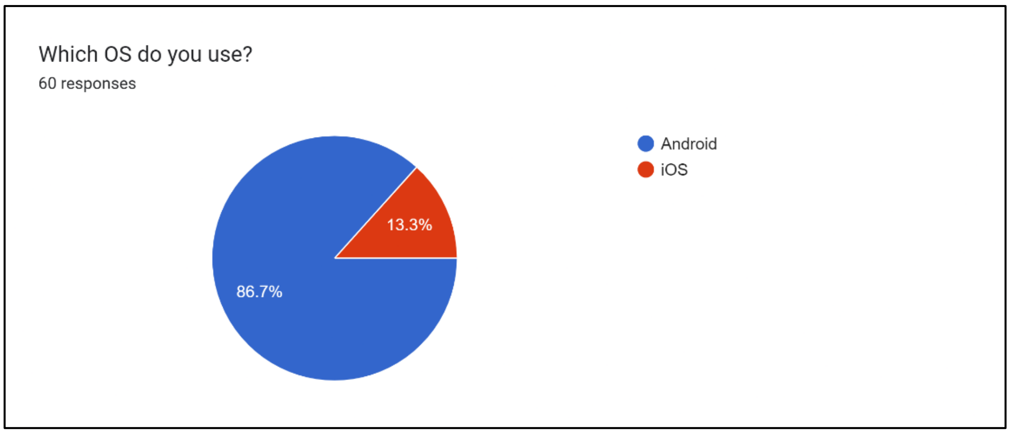
◯ Not sure

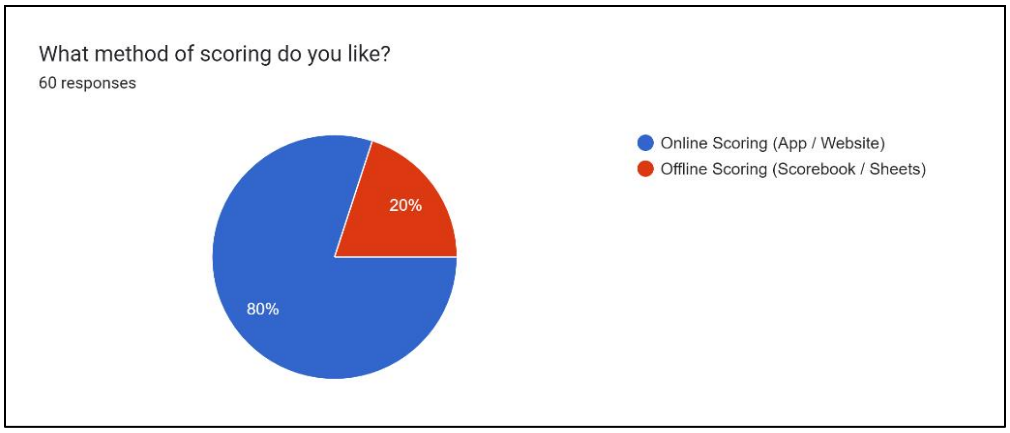
◯ Will be of no use

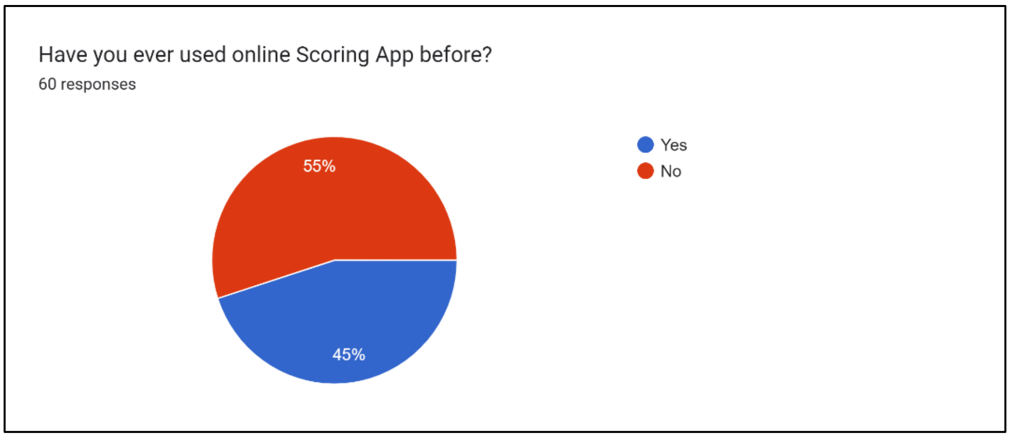
Do you have any other suggestions related to my app?

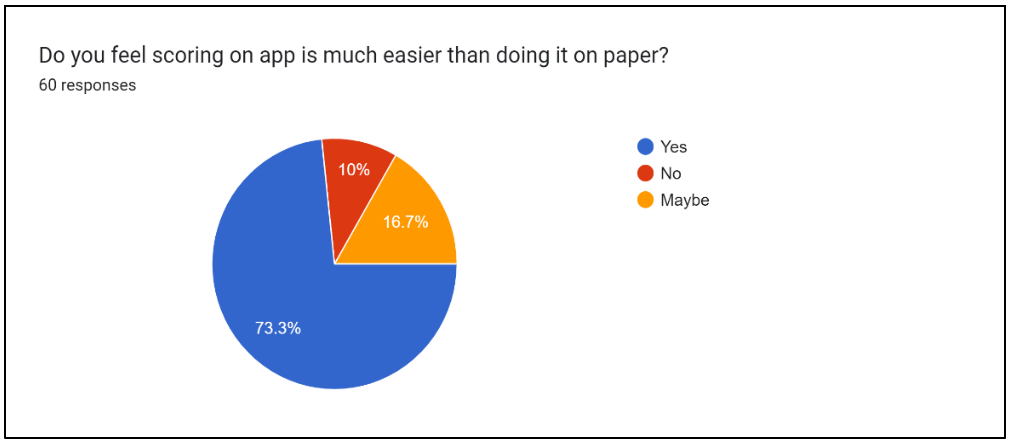
Your answer

Never submit passwords through Google Forms.









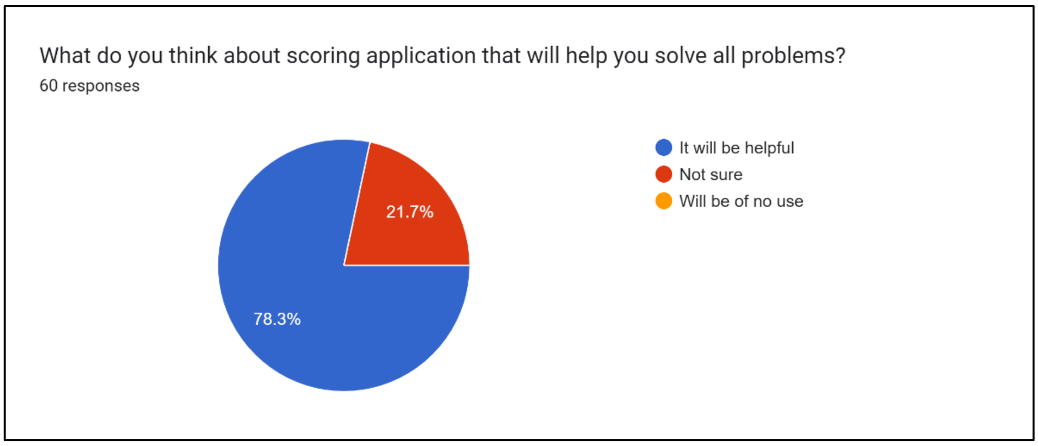


Fig. 3.1 Requirement Gathering

3.2.3 Functional Requirements

a. Login/sign up: 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-by-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. 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: -

5. Scorecard (Download):
    a) Description: Use will be able to download the scorecard of the completed match.
    b) Input: -
    c) Source: Database.
    d) Output: Downloads the scorecard.
    e) Destination: -
    f) Action: User gets to download the scorecard.
    g) Pre-condition: Match should be finished:
    h) Post-condition: -
6. 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: -

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

3.3.2 Gantt Chart

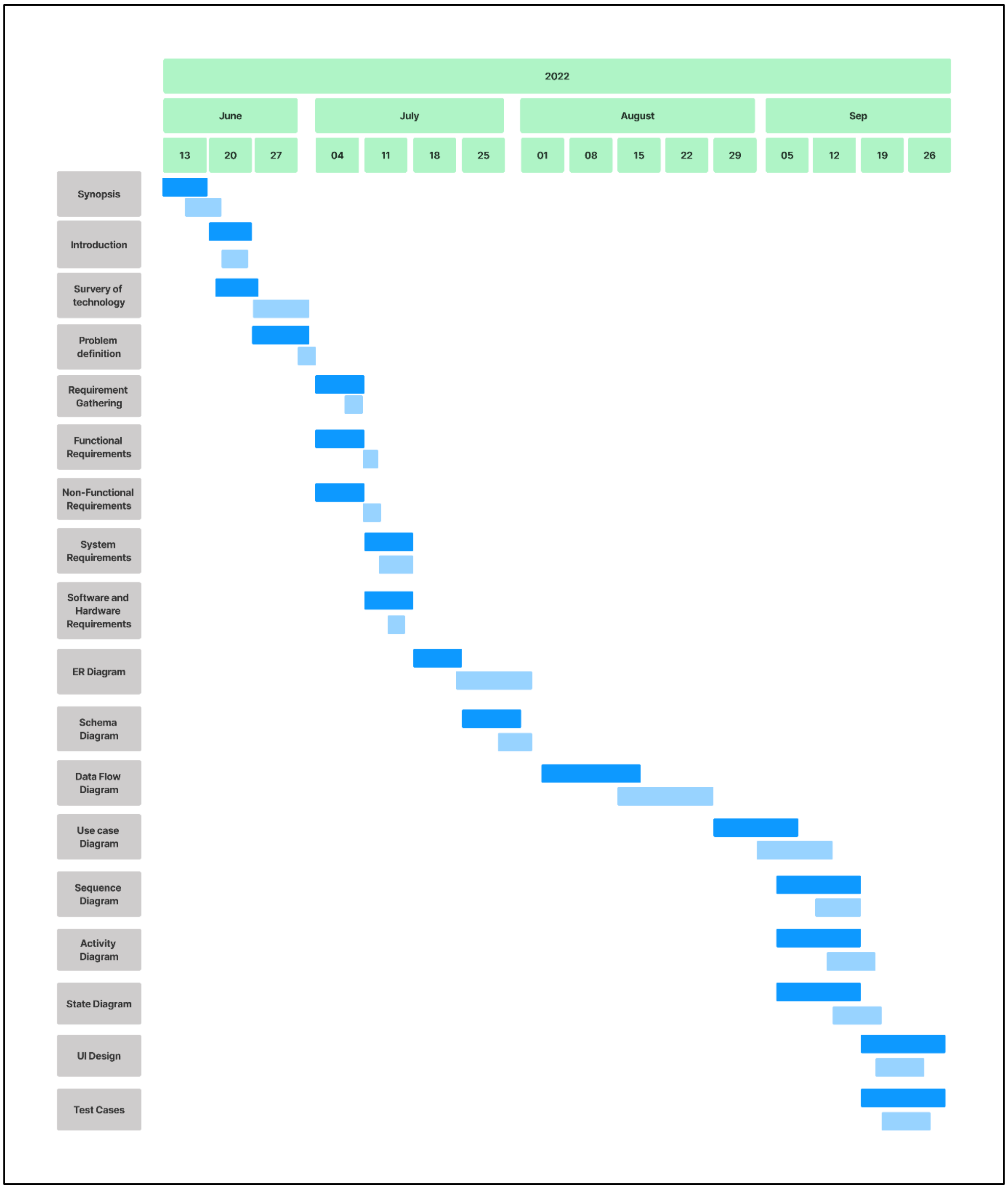


Fig. 3.2 Gantt Chart

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 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:

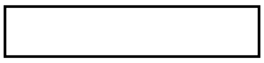
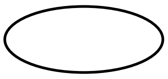
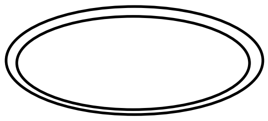
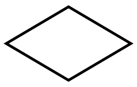
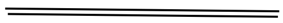
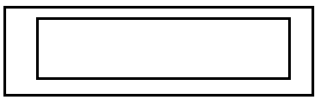
Name	Symbol	Description
Rectangle		Represents entity set
Ellipse		Represents attributes
Double Ellipse		Represent multivalued attributes
Diamond		Represents relationship set
Double Lines		Represents total participation
Double Rectangle		Represents weak entity

Table 3.2 ER-Diagram Notations

List of entity sets:

- User
- Match
- Player
- Team
- Score

List of relationship sets:

1. User crates a Match
2. Match has Teams
3. Match has Player
4. Match has Score
5. Player is a batsman or bowler

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, etc.

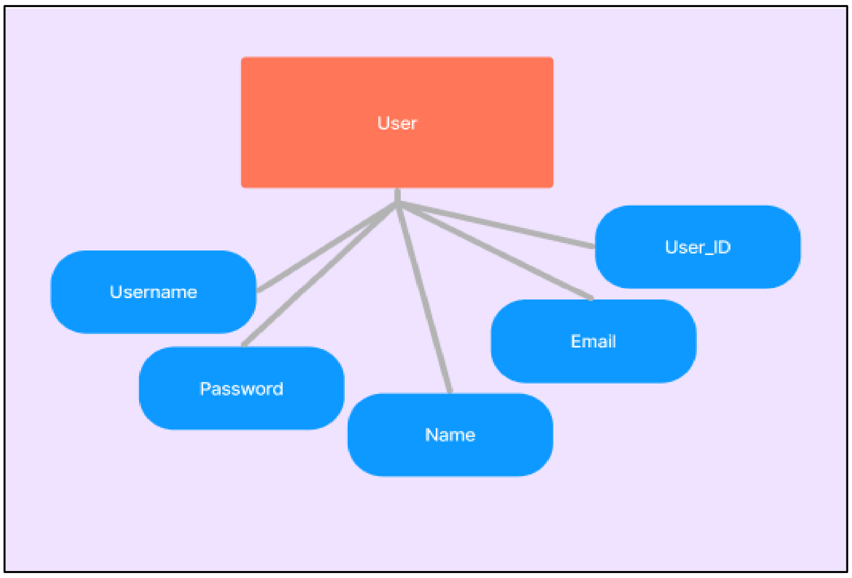


Fig. 3.3 User Entity Set

2. Match: 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.

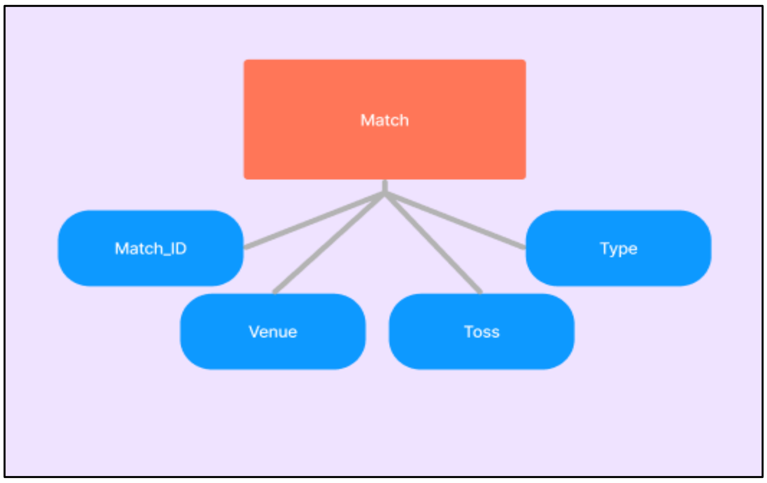


Fig. 3.4 Match Entity Set

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

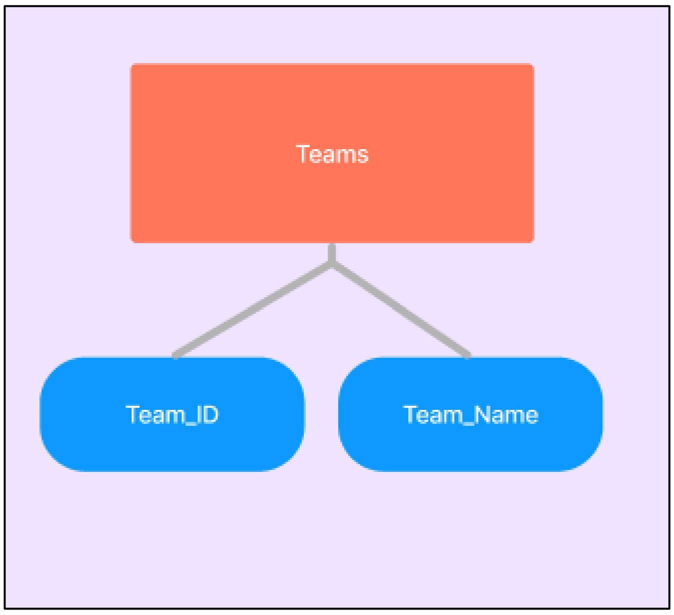


Fig. 3.5 Team Entity Set

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

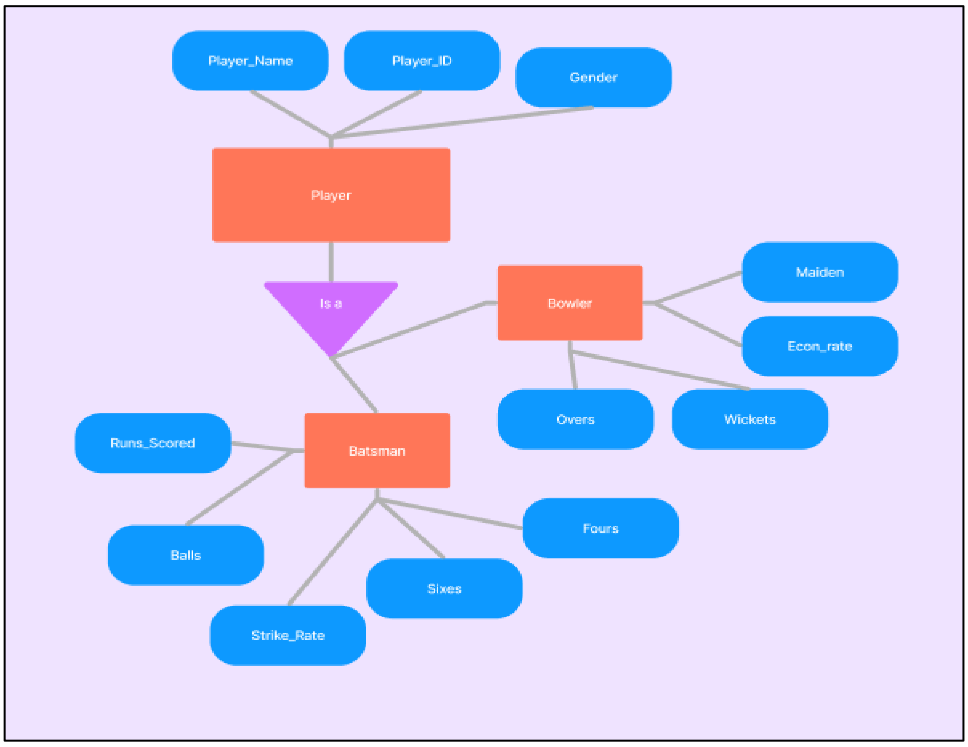


Fig. 3.6 Player Entity Set

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

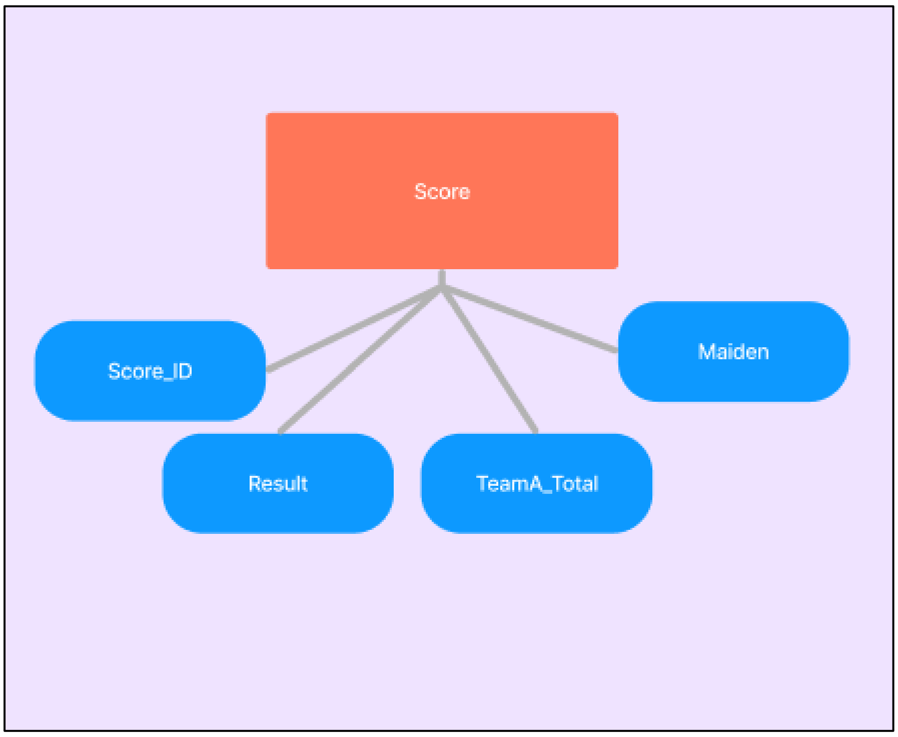


Fig. 3.7 Score Entity Set

3.5.2 Relationship Sets

1) User creates a Match
   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

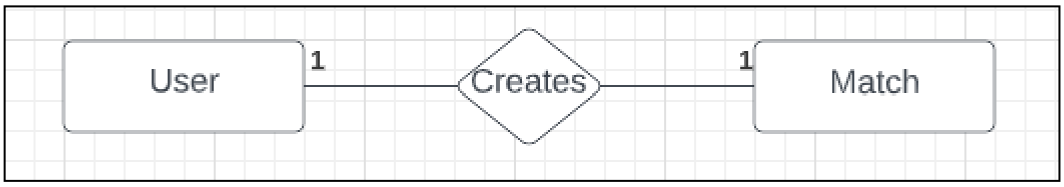


Fig. 3.8 Create Relationship Set

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

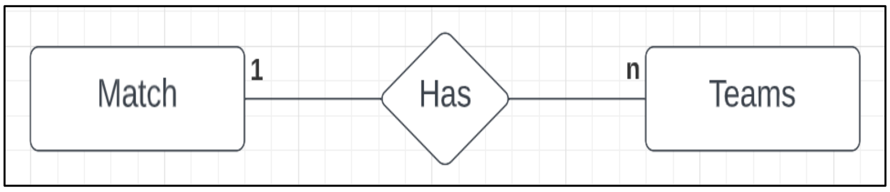
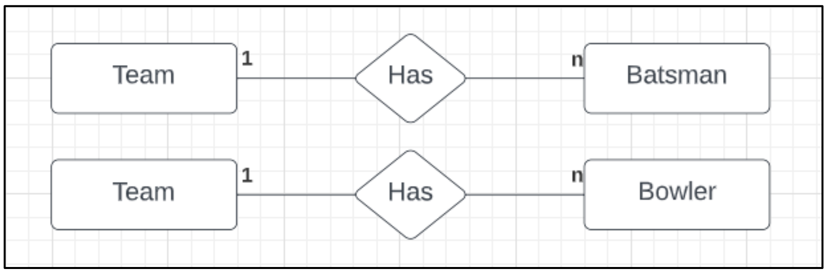


Fig. 3.9 Has Relationship Set

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



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

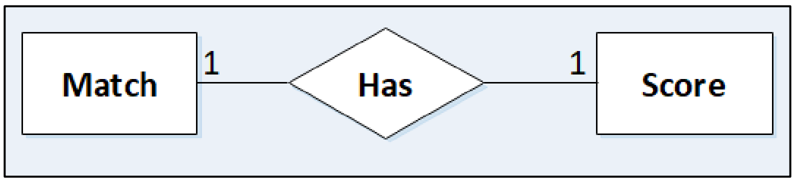


Fig. 3.10 Has relationship set

3.5.1.3 Entity-Relationship Diagram

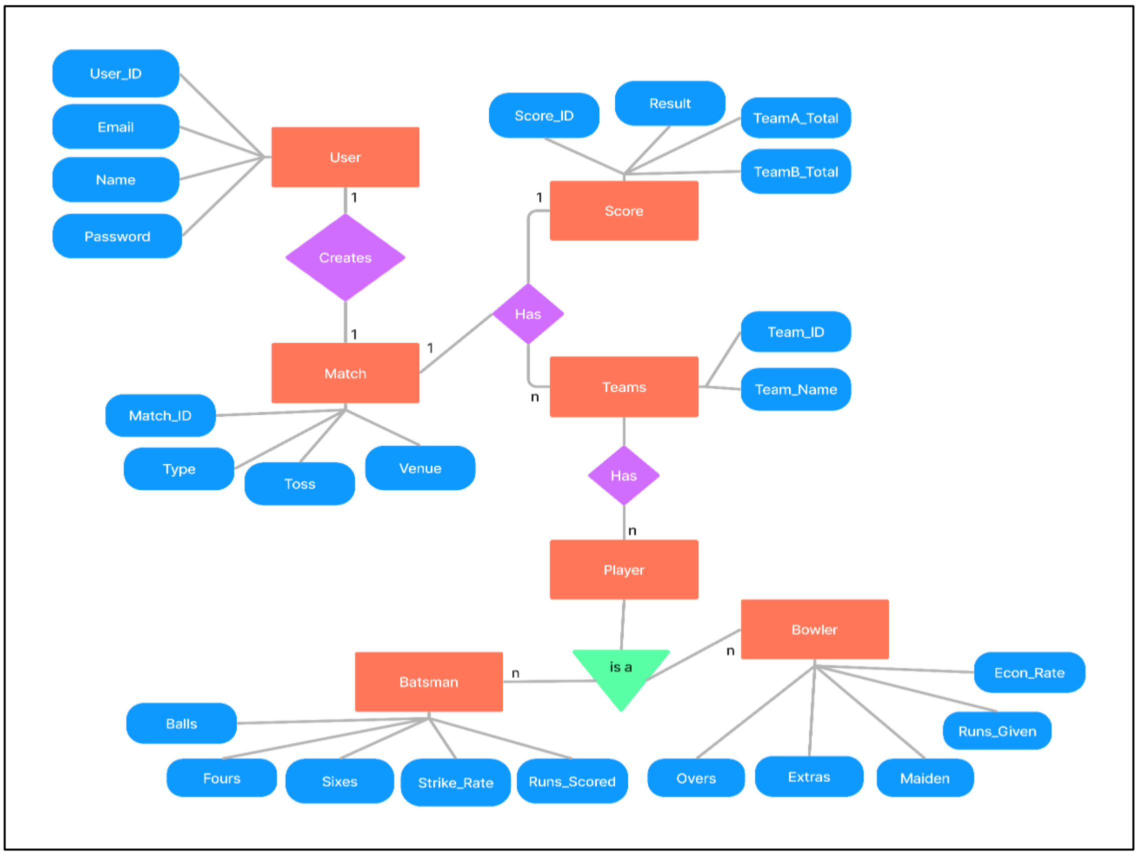


Fig. 3.10 ER Diagram

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 3.3 Schema Diagram Notations

Diagram:

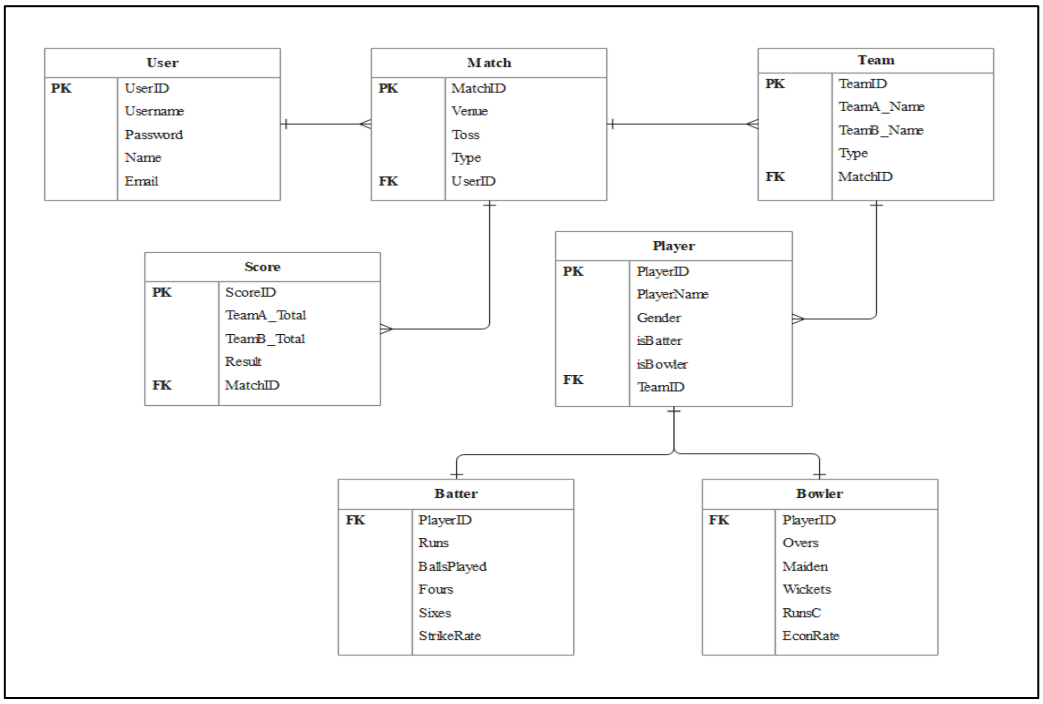


Fig. 3.11 Schema Diagram

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/

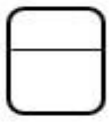
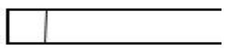
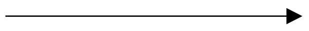
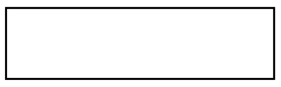
Name	Symbol	Description
Process		A process transforms incoming data flow into outgoing data flow.
Database		Data stores are repositories of data in the system.
Data Flow		Data flows are pipelines through which packets of information flow. Label the arrows with the name of the data that moves through 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):

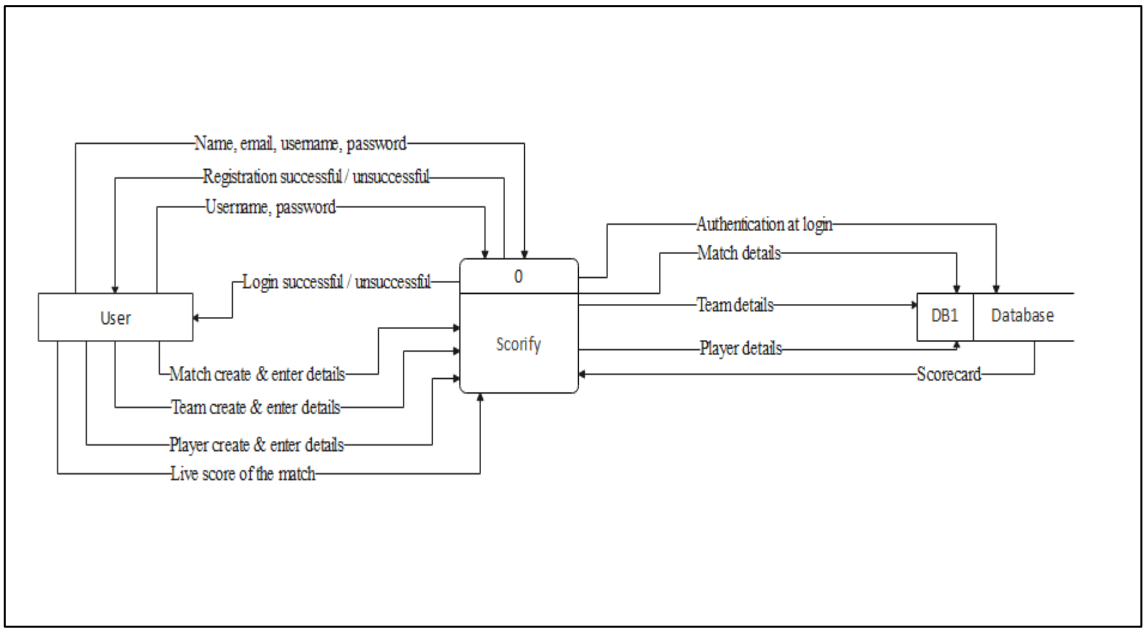

Name, email, username, password
Registration successful / unsuccessful
Username, password
Login successful / unsuccessful
0
User
Scorify
Match create & enter details
Team create & enter details
Player create & enter details
Live score of the match
Authentication at login
Match details
Team details
Player details
Scorecard
DB1
Database

Fig. 3.12 Level 0 DFD

Level 1 DFD:

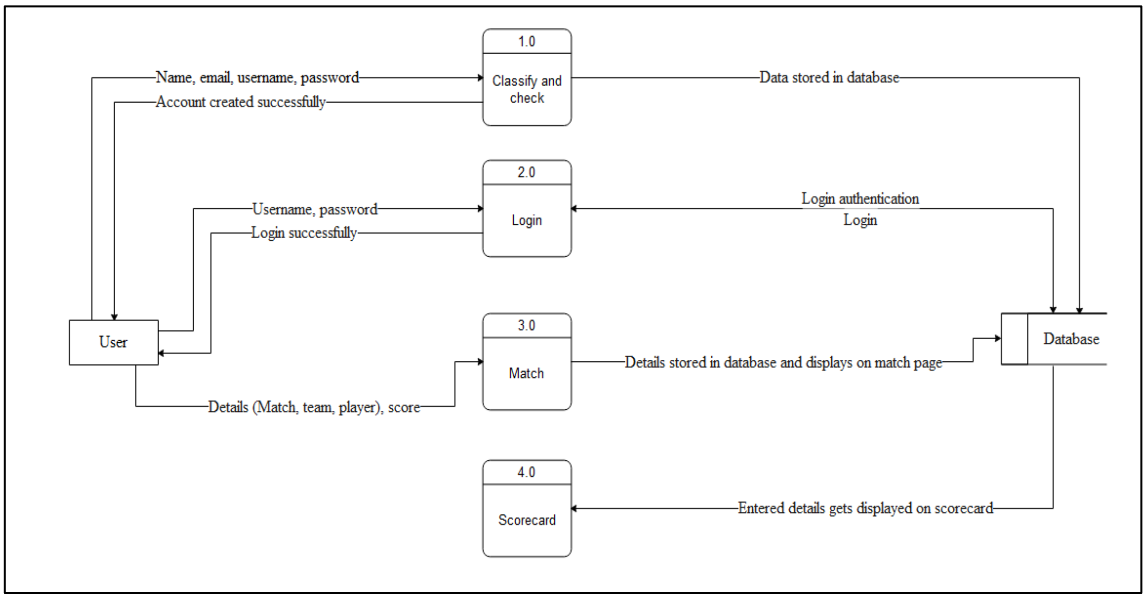

1.0
Classify and check
Name, email, username, password
Account created successfully
Data stored in database
2.0
Login
Username, password
Login successfully
Login authentication
Login
3.0
Match
User
Details (Match, team, player), score
Details stored in database and displays on match page
Database
4.0
Scorecard
Entered details gets displayed on scorecard

Fig. 3.13 Level 1 DFD

Level 2 DFD for Match:

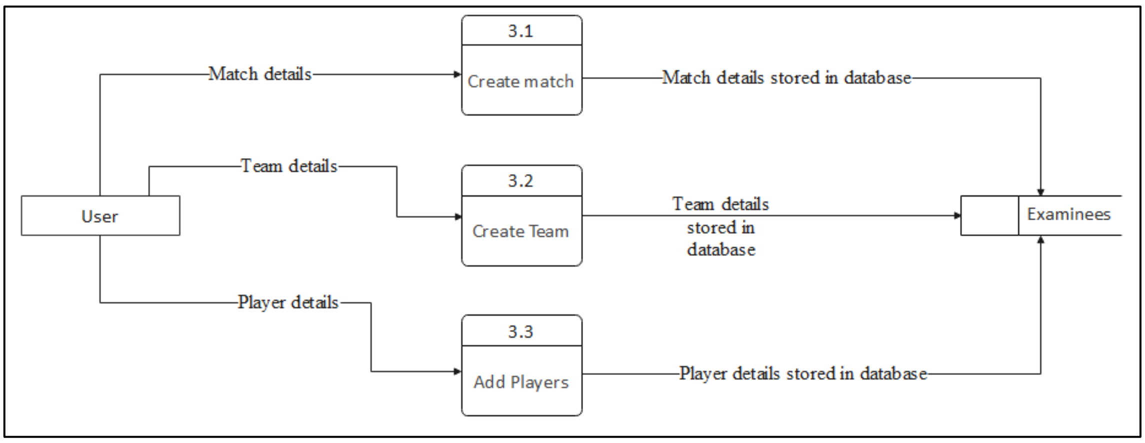

3.1
Create match
3.2
Create Team
3.3
Add Players
User
Examinees
Match details
Team details
Player details
Match details stored in database
Team details stored in database
Player details stored in database

Fig. 3.14 Level 2 DFD for match

Level 2 DFD for Scorecard:

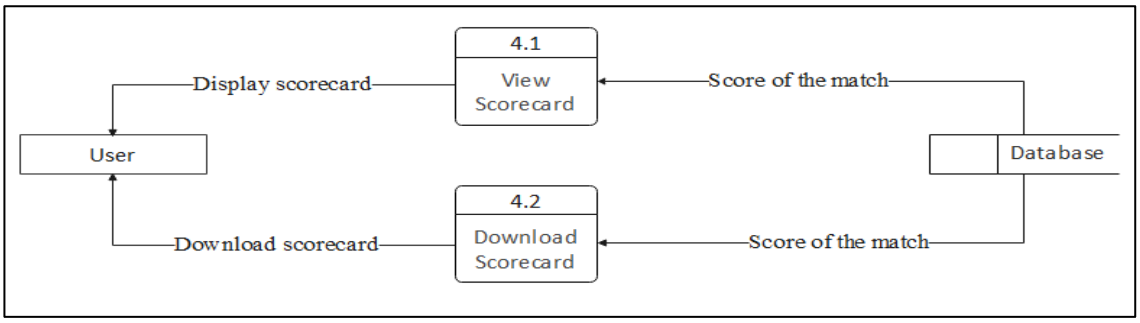


Fig. 3.15 Level 2 DFD for scorecard

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/

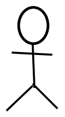
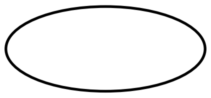
Name	**Symbol**	**Description**
Actor		Actor represents a user or another system that will interact with the system you are modelling.
Use Case		A use case is an external view of the system that represents some action the user might perform in order to complete a task.
Association	———————	Association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

Table 3.5 Use Case Notation

3.5.4.1 Diagram:

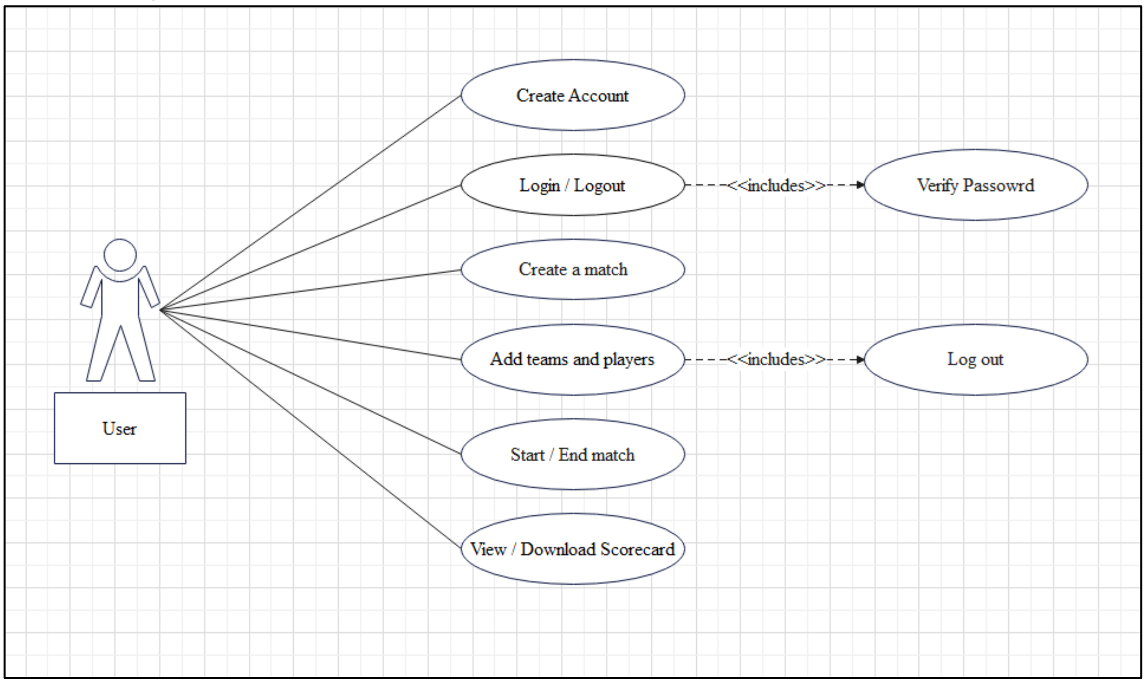


Fig. 3.16 Use Case Diagram

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 or download scorecard
    a. Description: The user will be able to view or download 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 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/

Name	Symbol	Description
Synchronous Message	⟶	Aninstantaneous 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:

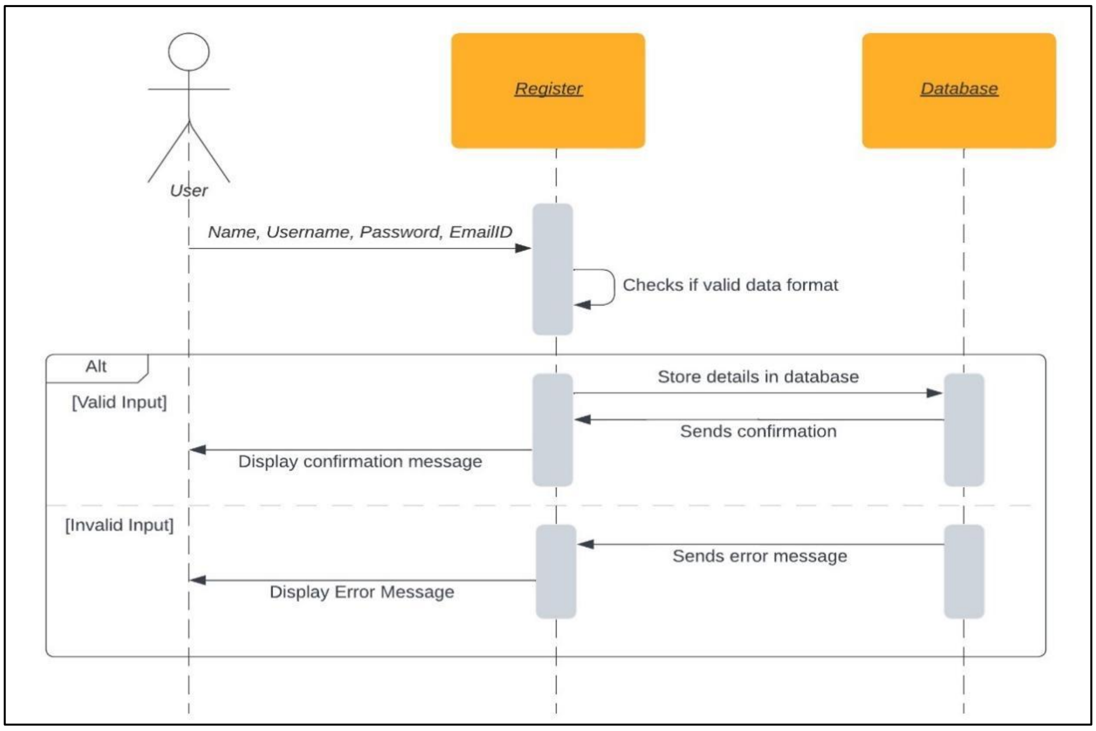

User
Register
Database
Name, Username, Password, EmailID
Checks if valid data format
Alt
[Valid Input]
Store details in database
Sends confirmation
Display confirmation message
[Invalid Input]
Sends error message
Display Error Message

Fig. 3.17 Sequence Diagram for Registration

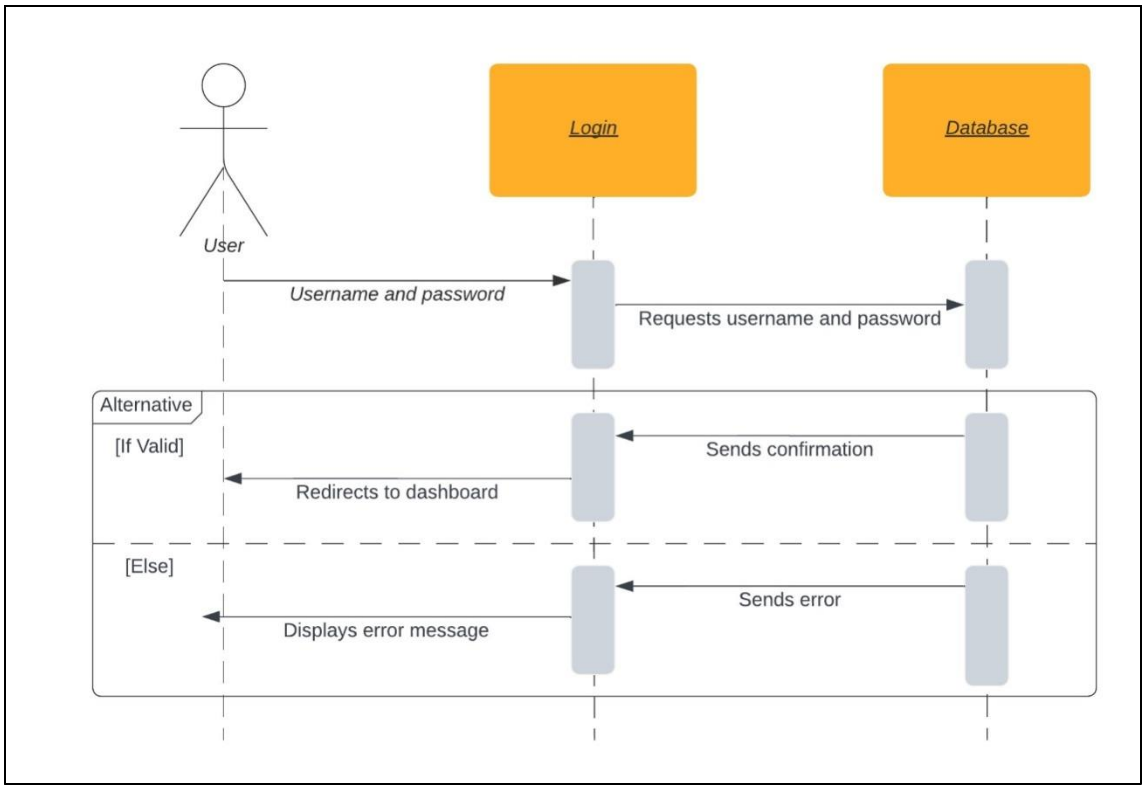


Fig. 3.18 Sequence Diagram for Login

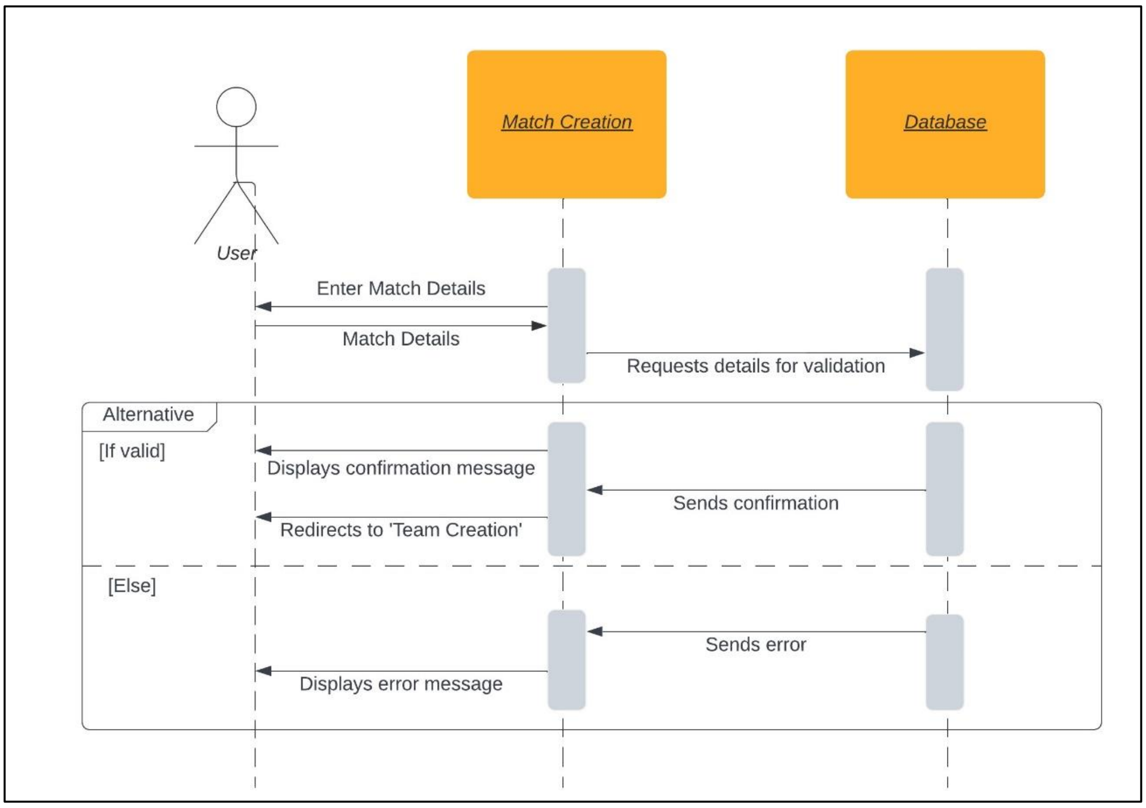


Fig. 3.19 Sequence Diagram for Match Creation

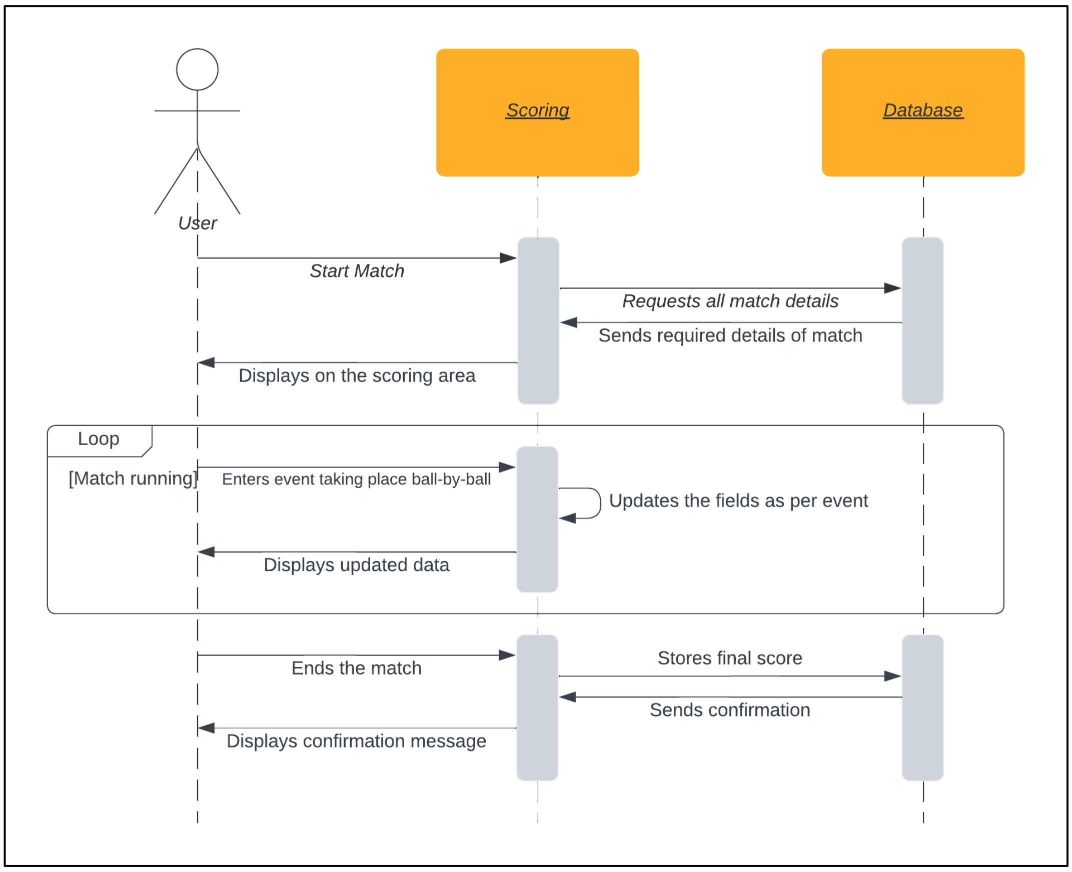

Scoring
Database
User
Start Match
Requests all match details
Sends required details of match
Displays on the scoring area
Loop
[Match running]
Enters event taking place ball-by-ball
Updates the fields as per event
Displays updated data
Ends the match
Stores final score
Sends confirmation
Displays confirmation message

Fig 3.20 Sequence diagram for Match Scoring

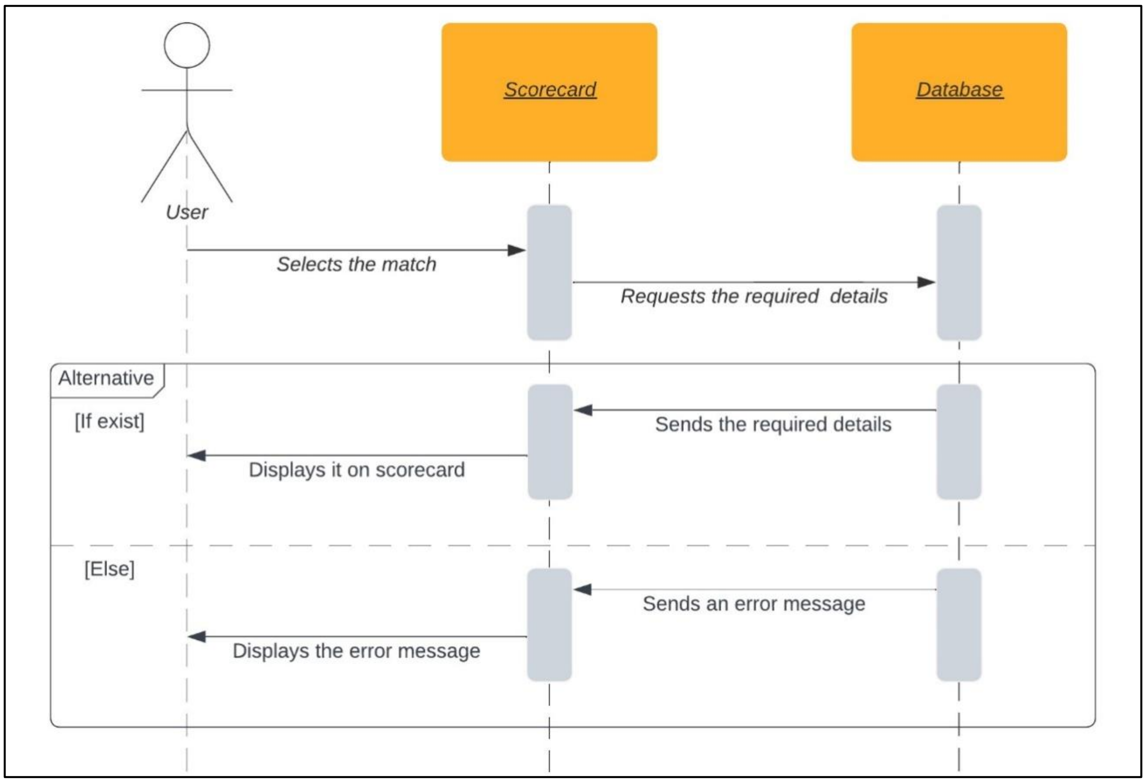


Fig. 3.21 Sequence diagram for scorecard

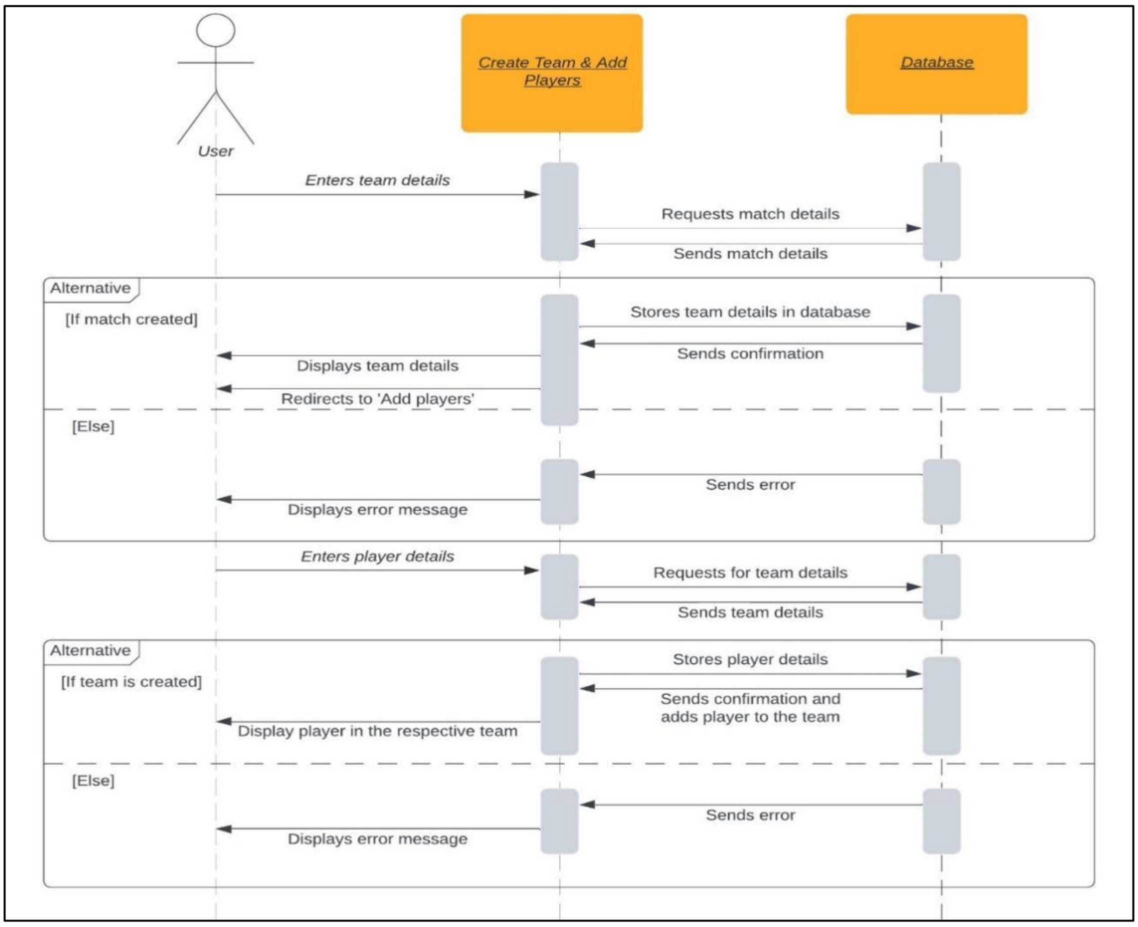


Fig. 3.22 Sequence diagram for Creation of teams and players

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 3.7 Activity Diagram Notations

Diagram:

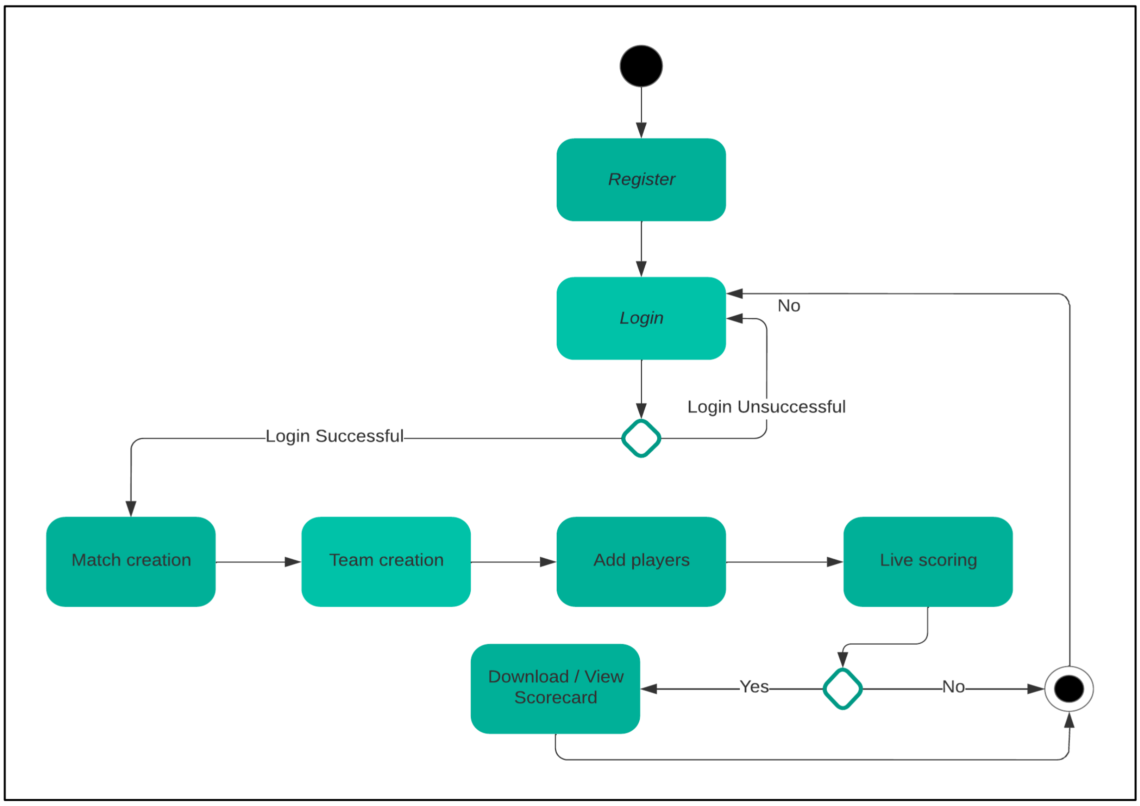


Fig 3.23 Activity Diagram

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:

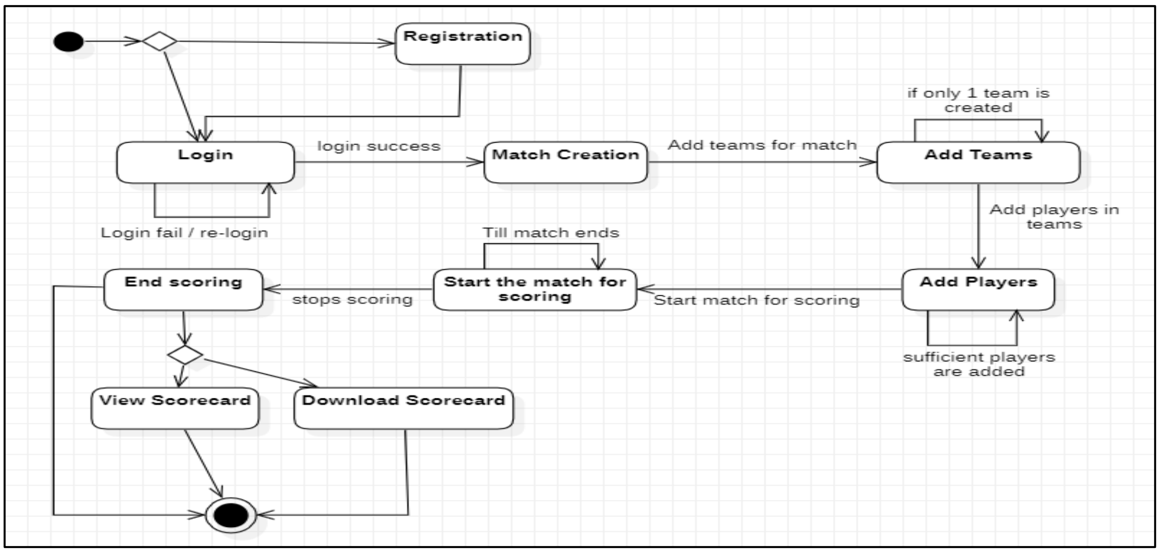


Fig. 3.24 State-Chart Diagram

4.1 User Interface

1. Home Page

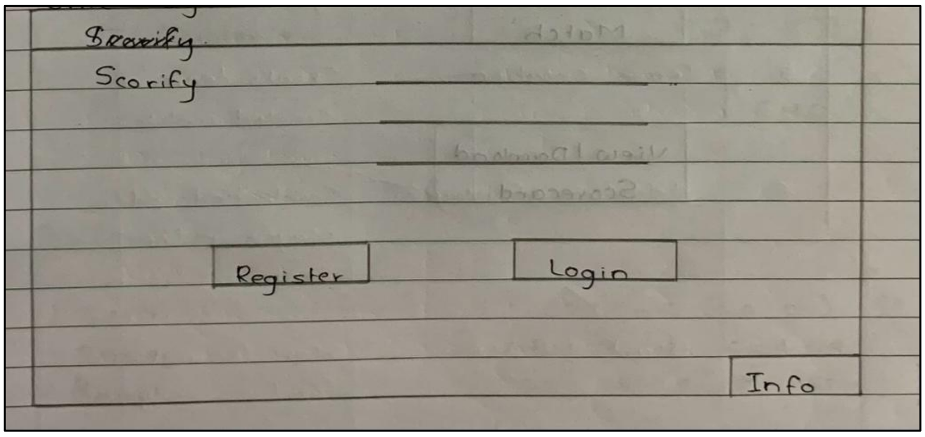


Fig. 4.1 UI for Home Page

2. Registration Page

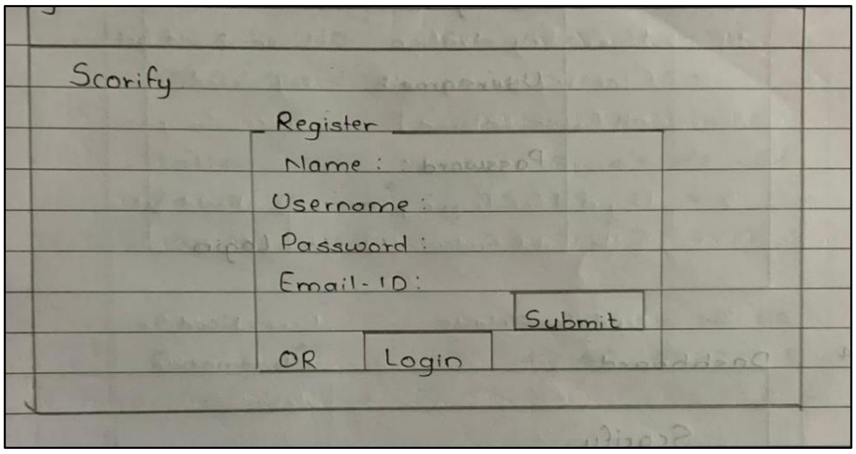


Fig. 4.2 UI for Registration Page

3. Login Page

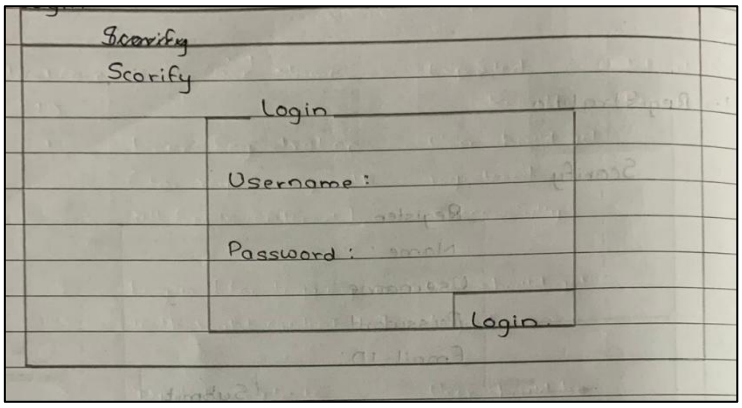


Fig. 4.3 UI for Login Page

4. Dashboard

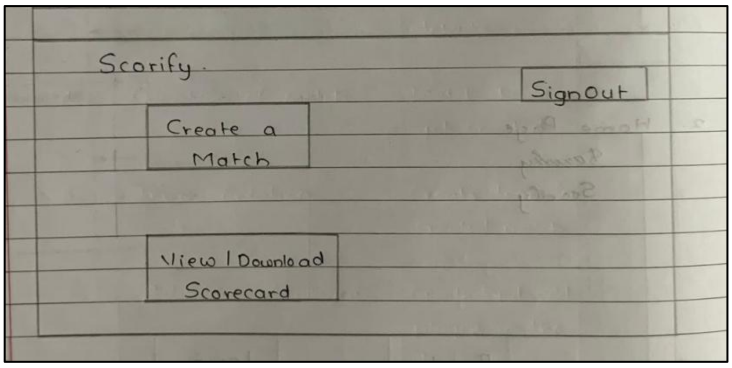


Fig. 4.4 UI for Dashboard

5. Live Scoring

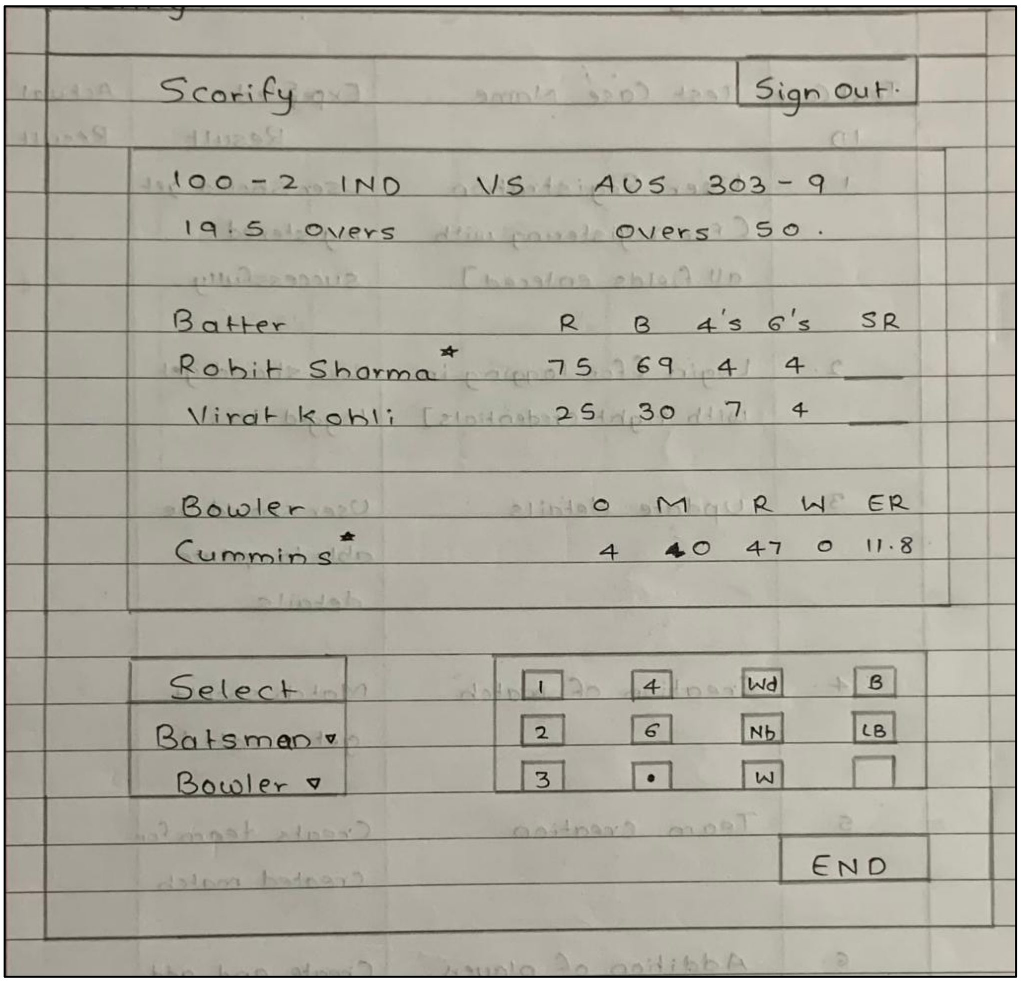


Fig. 4.5 UI for Live Scoring

6. Scorecard

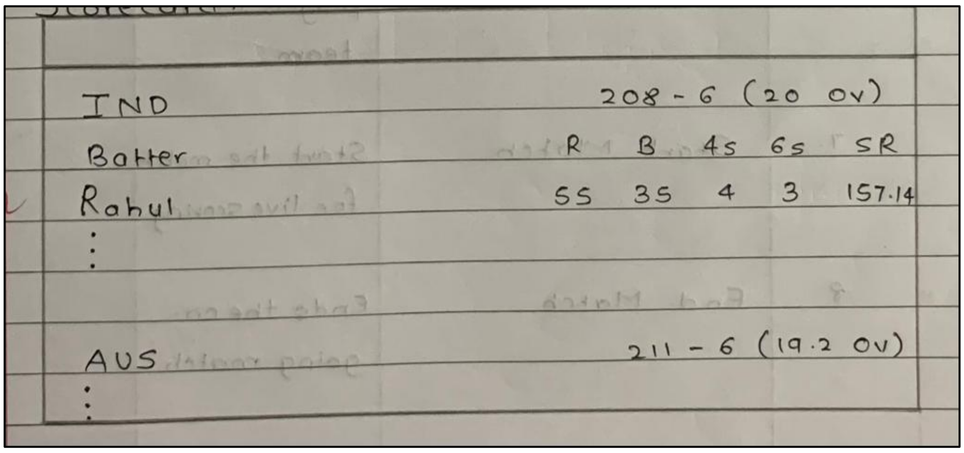


Fig. 4.6 UI for Scorecard

7. Creating match, team and players

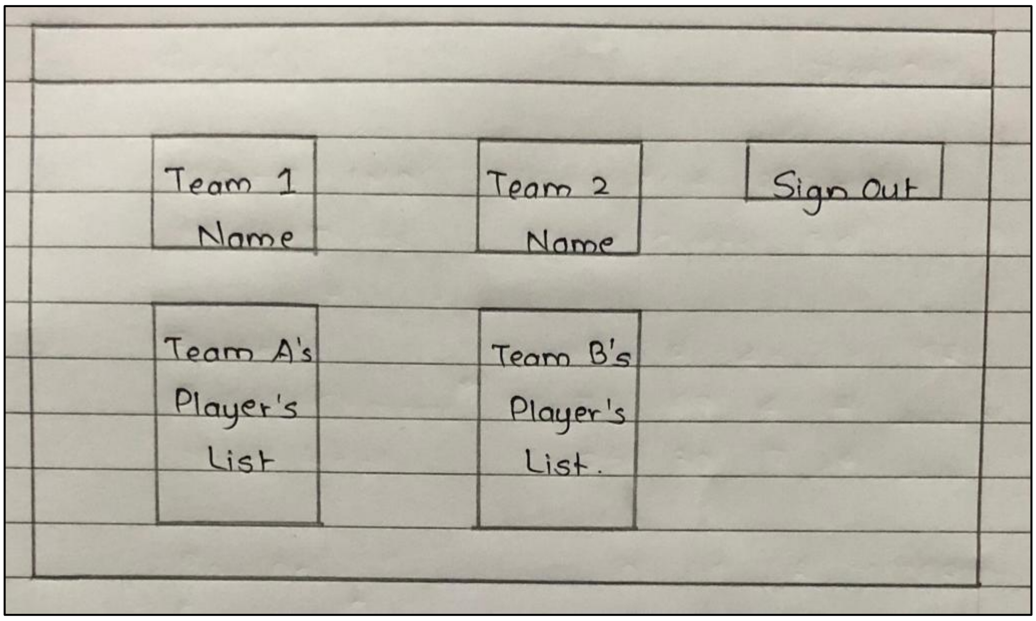


Fig. 4.7 UI For Creating match, team and players

4.2 Test Cases

Test Case No.	Test case Description	Test Case	Expected Output
1.	Register for user	Name: Pushkar Phone number: 835595447 Email id: pushkar@gmail.com Create password: pushkar@ Re-enter password: pushkar@	User has been registered successfully.
2.	Register for user	Name: Pushkar Phone number: 8355958 Email id: pushkar@gmail.com Create password: pushkar@ Re-enter password: pushkar@	Please enter valid phone number!
3.	Register for user	Name: Pushkar Phone number: 8355958447 Email id: pushkar@gmail.com Create password: pushkar@ Re-enter password: pushkar	Passwords don't match!
4	Login for user	Username: Pushkar Password: pushkar	Please enter correct password!
5.	Login for user	Username: pushkar Password: pushkar@	Please enter correct username!
6.	Login for user	Username: Pushkar Password: pushkar@	Redirects user to the dashboard.
7.	Create Match	Click on the button	User should get redirect to team creation page
8.	Team Creation	Team-A Name: India Team-B Name: Australia Venue: Hyderabad Type: T20I	User should get redirect to add player page
9.	Team Creation	Team-A Name: India Team-B Name: Australia Venue: Hyderabad Type: NULL	Enter match type!
10.	Team Creation	Team-A Name: India Team-B Name: India Venue: Hyderabad Type: T20I	Names of team cannot be same.

11	Start Match	Click on the button	User should get redirect to Scoring page and display entered details.
12	End Match	Click on the button	Stop the scoring of the match and display result box.
13.	Scoring	Click on wide	Should add 1 run in batting team and +1 in extras.
14.	Scoring	Click on no-ball	Should add 1 run in batting team and +1 in extras.
15.	Scoring	Click on 1-run	Add 1 run in striker runs and 1 run in bowler's run.
16.	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.
17	Scoring	Click on Ball	Add 1 ball in batsman and bowler.
18.	Scoring	Click on Wicket	End inning of batsman, add 1 wicket in batting team as well as bowler.
19.	Scoring	Selects new batsman	Display on scorecard.
20.	Scoring	Selects new bowler	Display on scorecard.
21.	Scoring	Clicks on end match	Should end the ongoing match.
22.	Scorecard	Clicks on view scorecard.	Display list of scorecards available to display to the user.
23.	Scorecard	Click on download scorecard	Display list of scorecards available to display and download.

24.	Home	Click on sign-out	User should get signed-out of their account.
25.	Player	Captain: abc Wicket-keeper: xyz	Display (c) & (wk) in front of their name.
26	Player	Captain: Wicket-keeper: xyz	You must select a captain of the team.
27	Player	Captain: abc Wicket-keeper:	You must select a wicket-keeper of the team.

Table 4.1 Test Cases

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/
- **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”