

## **CERTIFICATE**

Certified that **Mudit Kumar, Satyam Gubrele, Kuldeep Gupta, Gaurav Srivastav** have carried out the project work having “**Live Cricket Score**” for Master of Computer Applications from Dr. A.P.J. Abdul Kalam Technical University (AKTU) (formerly UPTU), Technical University, Lucknow under my supervision. The project report embodies original work, and studies are carried out by the student himself/herself and the contents of the project report do not form the basis for the award of any other degree to the candidate or to anybody else from this or any other University/Institution.

**Date:**

**Mudit Kumar (2100290140088)  
Satyam Gubrele (2100290140118)  
Kuldeep Gupta (2100290140079)  
Gaurav Srivastav (2100290140065)**

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

**Date:**

**Dr. Vipin Kumar  
Associate Professor**

**Department of Computer Applications  
KIET Group of Institutions, Ghaziabad**

**Signature of Internal Examiner  
Examiner**

**Signature of External**

**Dr. Arun Tripathi  
Head, Department of Computer Applications  
KIET Group of Institutions, Ghaziabad**

## **ABSTRACT**

The project entitled "**Live cricket score system**" which is utilized by the user with an update of the cricket even when the user is not watching the match. Each and every match details such as the description about the team and team members will be stored in the repository system in the form of database.

That database could be utilized by the mislaid by the users. Each and every match can be updated lively using this software. As soon as someone checks the scoreboard, details of a particular player can be viewed by a single click on his name any news other than cricket will also be updated. This software is error free anyone can use this software. You can download this software by clicking on download below. So use this software and get more benefit from this.

This project aims to make a live cricket score that will update the scores of a match along with commentary as it happens. Cricket being a special part of the lives of many people, there will be many takers for such a system and the ability to follow the match without seeing the video will make it interesting for many. A user, who is unable to watch the event like someone who is busy with their work, can easily check the commentary on a regular basis to get updates on what is happening.

The system will keep posting updated scores and the team line-up during the match. The admin will store upcoming match details and ensure that the team information posted the upcoming events; this will help the admin easily load information at the time of the match. This project will help the people who need to improve their performance of the event which will help to progress the betterment

## **ACKNOWLEDGEMENT**

Success in life is never attained single handedly. My deepest gratitude goes to my thesis supervisor, **Dr. Vipin Kumar** for his guidance, help and encouragement throughout my research work. Their enlightening ideas, comments, and suggestions. Words are not enough to express my gratitude to Dr. Arun K Tripathi Professor and Head, Department of Computer Applications, for his insightful comments and administrative help at various occasions.

Fortunately, I have many understanding friends, who have helped me a lot on many critical conditions. Finally, my sincere thanks go to my family members and all those who have directly and indirectly provided me moral support and other kind of help. Without their support, completion of this work would not have been possible in time. They keep my life filled with enjoyment and happiness.

**Mudit Kumar – 2100290140088 (B)**

**Satyam Gubrele – 2100290140118 (B)**

**Kuldeep Gupta – 2100290140079 (B)**

**Gaurav Srivastav – 2100290140065 (B)**

# TABLE OF CONTENTS

Certificate	i
Abstract	ii
Acknowledgements	iii
Table of Contents	iv
1 Chapter	5-6
1. Introduction	
1.1 Project description	5
1.2 Existing System	5
1.3 Project Scope	5
1.4 Module Design	5
1.5 Hardware/Software used in project	6
2 Chapter	7-8
2. Feasibility Study	
2.1 Operational Feasibility	7
2.2 Technical Feasibility	7
2.3 Economical Feasibility	8
3 Chapter	9-24
3. Design	
3.1 Architecture Diagrams	9
3.2 Workflow Diagram For Web Application	12
3.3 ER Diagram	13
3.4 Requirements Analysis	13
3.5 UML Diagrams	16
4 Chapter	26-28
4. UI Designs	
5 Chapter	29-69
5. Coding	
6 Chapter	70-75
6. Software Testing	
7 Chapter	69-79
7. Maintenance	
8 Chapter	80-84
8. Literature Review	
Bibliography	85

# **Chapter: 1**

## **Introduction**

### **1.1. Project description**

The project live cricket score developed in React is used to provide user with an update of the live cricket even when the user is not watching the mach. The user can use this website anytime; anywhere to see the teams, matches, player's squad, and runs scored by each player and can also view the reviews and commentary. This gives original experience of watching the match by the user.

Software or website developed must be built from user's point of view. It must be able to fulfill all the drawbacks that user's face in existing system. Our system fulfils and satisfies the user also it gives the experience of watching the match by adding the commentary in the website. The website is maintained by admin where he/she updates the score without any delay.

Adding teams, matches, players and score of each player is the responsibilities of admin. User can just register with their basic details and login to the website to view the team details and scores of individual. The system overcomes all the drawbacks and maintains the website up to date.

### **1.2. Existing System**

The existing Cricket Score Board system of watching cricket is generally on the television. Most matches are not scheduled on holidays and this will allow people access to the match regardless of their location. Some sites do exist that display text commentary but they are very impersonal.

### **1.3. Project Scope**

The proposed Cricket Score Board system will allow people to stream the video of the match anywhere and read text based commentary as well, while making cricket viewing a social experience by allowing friends to share text and audio commentary. The commentary will be available post the match as well for review.

## **1.4.Modules**

There are two main modules of this application one is the Admin and another one is the user, let us see the responsibility

### **1.4.1. Admin:**

- Create Players
- Create Match
- Create Team
- Toss Results
- Update Live Score
- Update Ball by ball runs
- Current player Update

### **1.4.2. Users:**

- View Batsman Score Board
- View Bowler Score Board

## **1.5.Hardware / Software used in Project**

### **1.5.1. Software:**

1. Operating System (windows)
2. Code Editor: VS Code
3. Front End: HTML, CSS, JavaScript, Bootstrap, React Js

### **1.5.2. Hardware:**

2. Intel i3 or Above
3. 4GB RAM or Above
4. Hard-Drive Capacity 64GB or Above

## CHAPTER 2

### 2. Feasibility Study:

This project will be developed on computer, so first check whether the technology is technically available or not. Now a day's computer interaction with any job becomes common for any kind of job or work. And because of increasing usage of Computer, Computer is also available with a variety of hardware. Users can fulfill any type of hardware requirement.

Preliminary investigation of a system examines the feasibility of a system that is useful to an organization. It is the first phase of system development.

The main objective of this phase is to identify the current deficiencies in the user's environment and to determine which existing problem are going to be solve in proposed system and also which new function needs to be added in proposed system.

An important outcome of such preliminary investigation is to determine whether the system that will meet all needed requirements.

Thus, three tests are carried out on the system namely operation, technical and economical.

#### 2.1. OPERATIONAL FEASIBILITY

Any project is beneficial if and only satisfies the organization's requirement. For any new system setup, it only meets to be communicated and work the other supporting system.

The new system meets all existing operations since it provides right information at a right time to the right user. A Leigh man can easily operate with the system.

#### 2.2. TECHNICAL FEASIBILITY

Technical Feasibility examines whether the technology needed is available and if it is available then it feasible to carry out all project activities.

The technical needs of a system include:

- The facility to produce outputs in a given time.
- Ability to process large number of transaction at a particular speed.
- Giving response to users under certain conditions.

The technology needed for our system is mainly:

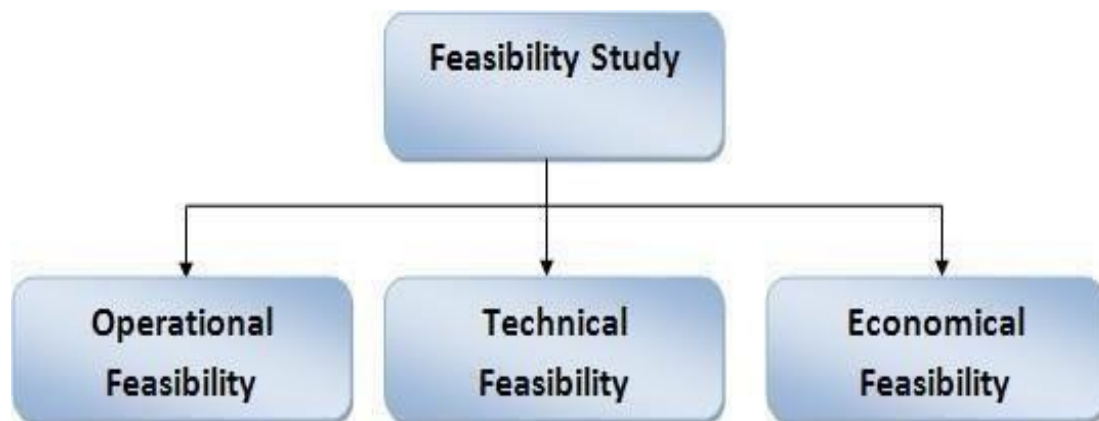
- Latest version of browsers.
- Any operating system.

These technologies are available which helps to carry out the system efficiently.

### **2.3. ECONOMICAL FEASIBILITY**

Economical feasibility of a system examines whether the finance is available for implementing the new system and whether the money spent is recoverable the satisfaction. The cost involves is in designing and developing a good investment for the organization. Thus, hardware requirements used for proposed system are very standard. Moreover, by making use of proposed system to carry out the work speedily will increase and also saves the valuable time of an organization.

In the proposed system the finance is highly required for the installation of the software's which can also be recovered by implementing a better system.

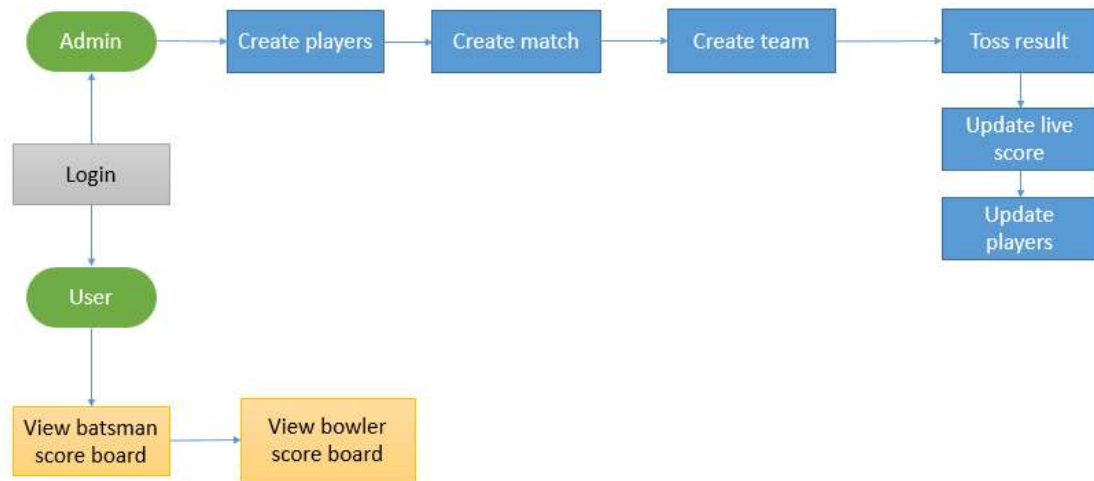




## CHAPTER 3

### DESIGN

#### 3.1. ARCHITECTURE DIAGRAMS

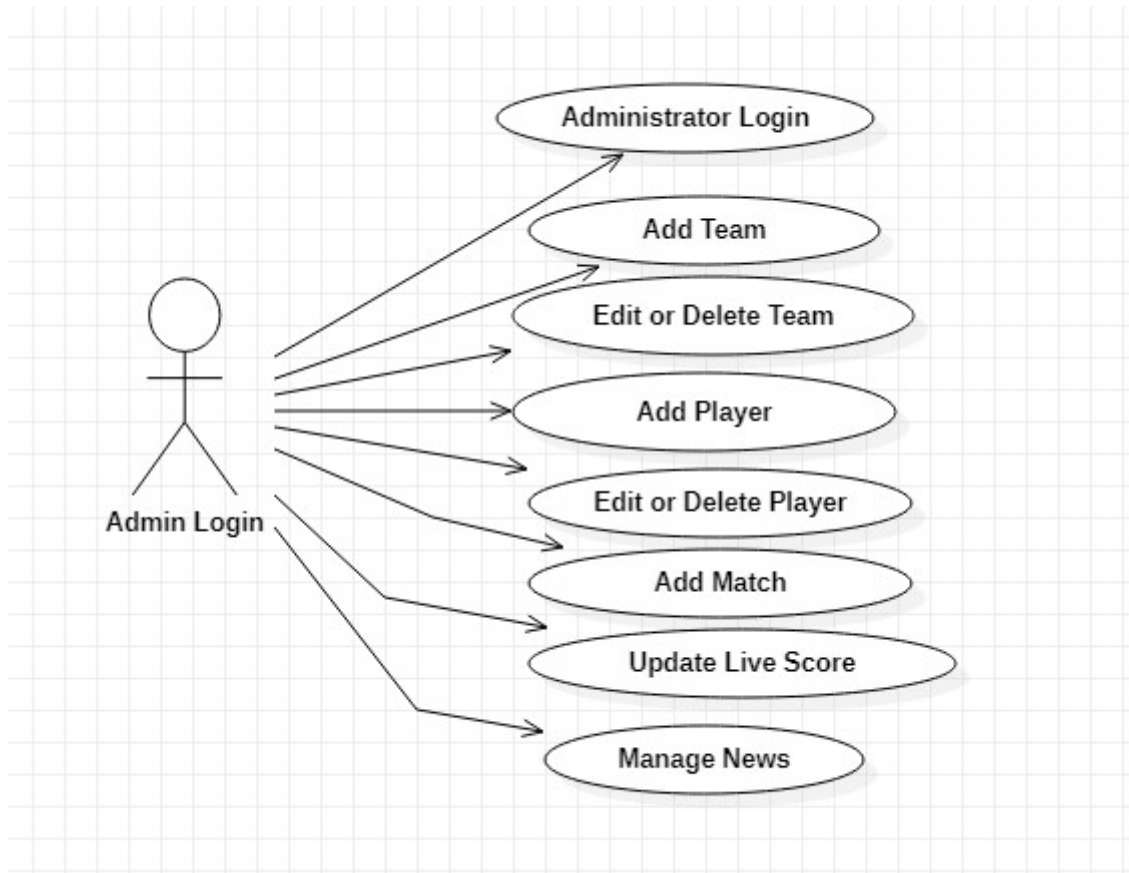


An architectural diagram is a visual representation that maps out the physical implementation for components of a software system.

In this diagram, these entities like Admin, users are doing their specific tasks. Admin can create players list, create matches, create team, update the toss result and update the live score of the current match.

Where User can check batsman Score board and view bowler score board But the common tasks of both of them is to login into the website.

## ADMIN

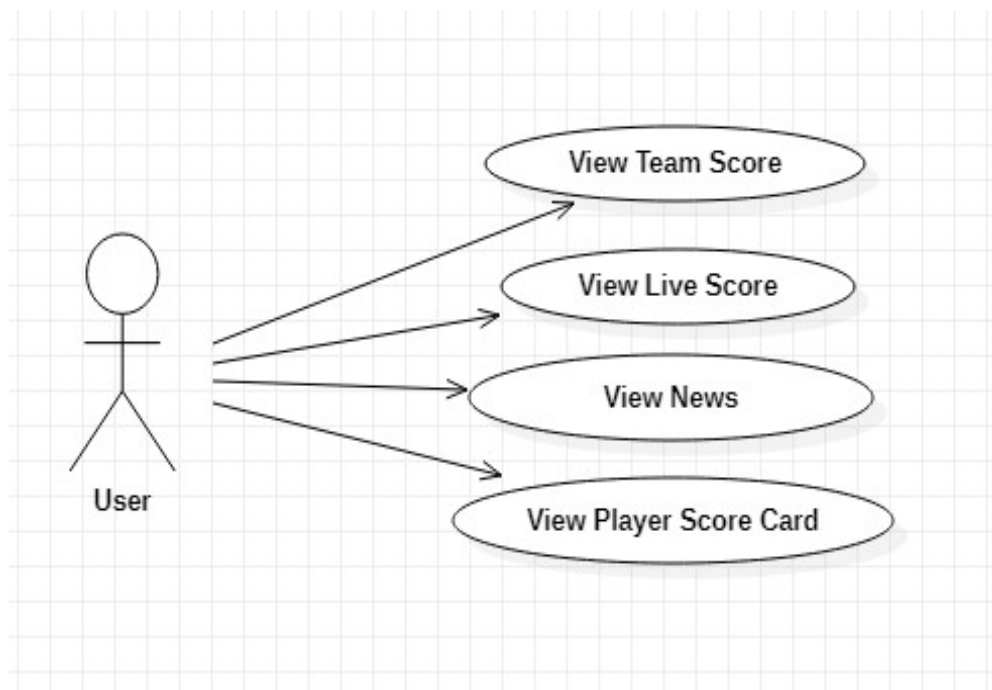


This use case diagram is describing the working of Admin side that will perform some special task.

In this module we provide the Admin login to login into the website after successfully login the admin can eligible to add the details of the teams to add details about the player who can play the match and add the details about current match.

Also provide the facility to update the live score of the match on the website.

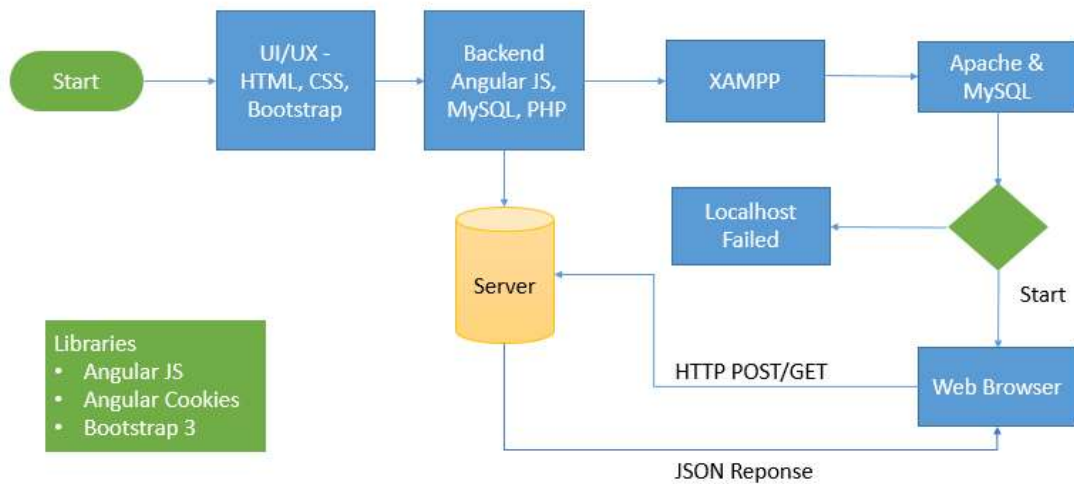
## USER



This use case diagram is describing the working of User side that will view the some component.

In this module user only can view the details of current and past matches. Like team score both side of Inning, live score card, view the details about the player score card and also able to view the news about the current, past and upcoming matches news on the website which helps to user to readable the news about the matches user are up to date about match.

### 3.2. WORKFLOW DIAGRAM FOR WEB APPLICATION



As we can see in the above diagram in react the dynamic html code is generated then it gets connected to the database and it starts working on a virtual server created with the help of XAMPP then it get started to work on the web browser

### 3.3. ER-Model / Diagrams

The entity-relationship (ER) data model allows us to describe the data involved in a real-world enterprise term of objects and their relationships and is widely used to develop an initial database.

The ER model is important primarily for its role in database design. It provides useful concepts that allow us to move from an informal description of what users want from their database to a more detailed and precise, description that can be implemented in a DBMS. We note that many variations of ER diagrams are in use, and no widely accepted standards prevail.

The database design process can be divided into six steps. The ER model is most relevant to the first three steps:

### 3.4. Requirements Analysis:

The very first step in designing a database application is to understand what data is to be stored in the database, what applications must be built on top of it, and what operations are most frequent and subject to performance.

#### 3.4.1. Conceptual Database Design:

The information gathered in the requirements analysis step is used to develop a high-level description of the data to be stored in the database, along with the constraints that are known to hold over this data. This step is often carried out using the ER model, or a similar high-level data model, and is discussed in the rest of this chapter.

#### 3.4.2. Logical Database Design:

We must choose a DBMS to implement our database design, and convert the conceptual database design into a database schema in the data model of the chosen DBMS. We will only consider relational DBMS, and therefore, the task in the logical design step is to convert an ER schema into a relational database schema. The result is a conceptual schema, sometimes called the **logical schema**, in the relational data model.

#### 3.4.3. Beyond the ER model:

ER model is sometimes regarded as a complete approach to designing a logical database schema. This is incorrect because the ER diagram is just an approximate description of the data, constructed through a very subjective evaluation of the information collected during requirements analysis.

#### 3.4.4. Schema Refinement:

The fourth step in database design is to analyse the collection of relations in our relational database schema to identify potential problems, and to refine it.

#### 3.4.5. Physical Database Design:

In this step we must consider typical expected workloads that our database must support and further refine the database design to ensure that it meets desired.

#### 3.4.6. Security Design

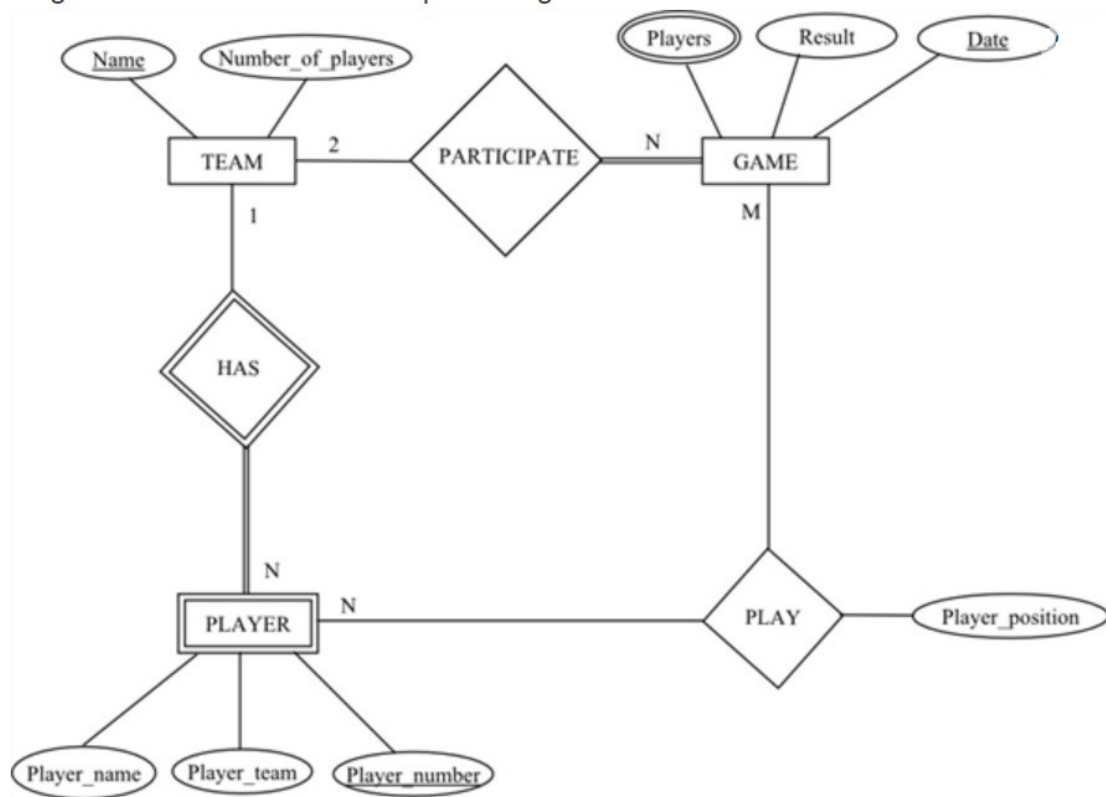
In this step, we identify different user groups and different **roles** played by various users (e.g., the development for a product, the customer support representatives, and the product manager).

#### 3.4.2. Entities, Attributes and Entity sets:

An **entity** is an object in the real world that is distinguishable from other objects manager of the toy department, the home address of the manager of the toy department. It is often useful to identify a collection of similar entities. Such a collection is called an **entity set**. Examples include the following: the Green Dragonwort toy, the toy department, and the performance criteria.

An entity is described by set of **attributes**. All entities in a given entity set have the same attribute; this is essentially what u has seen by similar. For each attribute associated with an entity set, we must identify a **domain** of possible values. A **key** is a minimal set of attributes whose values uniquely identify an entity in the set. There could be more than one **candidate** key.

## ER Diagram



An Entity Relationship (ER) Diagram is a type of flowchart that illustrates how “entities” such as people, objects or concepts relate to each other within a system.

ER Diagrams are most often used to design or debug relational databases in the fields of software engineering, business information systems, education and research.

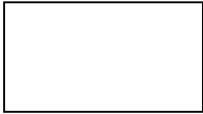

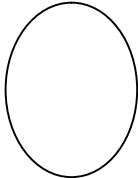

Also known as ERDs or ER Models, they use a defined set of symbols such as rectangles, diamonds, ovals and connecting lines to depict the interconnectedness of entities relationships and their attributes. They mirror grammatical structure, with entities as nouns and relationships as verbs.

Team, Game, Player are entities who associate with each other like Team can participate in game.

Team has their own attributes like Name, Number of players where Name is id attribute that make it unique. Game has their own attribute like Players, Results and date but here Players are multivalve attribute. These all are connected with each other with different cardinalities like one to one and many to one.

## Data Flow Diagram

Data flow diagrams (DFDs) use a number of symbols to represent systems. Most data flow modelling methods use four kinds of symbols to represent four kinds of system components: Processes, data stores, data flows and external entities (source or destination of data). The symbols that are used to represent the DFD are as follows: -

Symbol	Meaning
	Source or Destination of Data
	Data Flow
	Process that transform data flow
	Data Store

## 3.5. UML Diagrams

A diagram is a graphical representation of a set of elements. The various diagrams in UML are as follows:

### i. Class Diagram

A class diagram shows a set of classes, interfaces, and collaborations and their relationships. Class diagrams address the static design view of a system. Class diagrams that include active classes address the static process view of a system. A class is a description of a set of objects that share the same attributes, operations, relationships, and semantics. A class implements one or more interfaces.



## **ii. Use case Diagram**

A use case diagram shows a set of use cases and Actors (a special kind of class) and their relationships. Use case diagrams address the static use case view of a system. These diagrams are especially important in organizing and modeling the behaviors of a system.

## **iii. Sequence Diagram**

A sequence diagram is a visual representation of a scenario. A sequence diagram shows the various actors in the scenario, and the way they interact with all the subsystems.

## **iv. Collaboration Diagram**

A collaboration diagram is an interaction diagram that emphasizes the structural organization of the objects that send and receive messages. Collaboration diagram address the dynamic view of a system.

### **3.5.1. Unified Modelling Language:**

UML is a method for describing the system architecture in details using the blueprint. UML represents a collection of best engineering practices that have proven successful in the modelling of large and complex systems. The UML is a very important part of developing objects-oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects. Using the UML helps project teams communicate, explore potential designs, and validate the architectural design of the software

### **3.5.2. Definition**

UML is a general-purpose visual modelling language that is used to specify, visualize, construct, and document the artifacts of the software systems.

### **3.5.3. UML Specifying:**

Specifying means building models that are precise, unambiguous and complete. In particular, the UML address the specification of all the important analysis, design and implementation decisions that must be made in developing and displaying a software intensive system.

### **3.5.4. UML Visualization**

The UML includes both graphical and textual representation. It makes easy to visualize the system and for better understanding.

### **3.5.5. UML Constructing**

UML models can be directly connected to a variety of programming languages and it is sufficiently expressive and free from any ambiguity to permit the direct execution of models.

### **3.5.6. UML Documentation**

UML provides variety of documents in addition raw executable codes. A UML system is represented using five different views that describe the system from distinctly different perspective. Each view is defined by a set of diagrams, which is as follows.

### **3.5.7. User Model View**

- This view represents the system from the user's perspective.
- The analysis representation describes the usage scenario from the end-user's perspective.

### **3.5.8. Structural model view**

- In this model the data and functionality are arrived from inside the system.
- This model view models the static structures.

### **3.5.9. Behavioral model view**

- It represents the dynamic of behavioral as parts of the system, depicting the interactions of collection between various structural elements described in the user model and structural model view.

### **3.5.10. Implementation model view**

- In this the structural and behavioral as parts of the system are represented as they are to be built.

### **3.5.11. Environmental model view**

- In this structural and behavioral aspects of the environment in which the system is to be implemented are represented.

### **3.5.12. Goal of UML:**

1. The primary goals in the design of the UML were:
2. Provide users with a ready-to-use, expressive visual modelling language so they can develop and exchange meaningful models.
3. Provide extensibility and specialization mechanism to extend the core concepts.

4. Be independent of particular programming language and development processes.
5. Provide a formal basis for understanding the modelling language.
6. Encourage the growth of the OO tools market.
7. Support higher-level development concepts such as collaborations, frame works, patterns and components.
8. Integrate best practices

### **Uses of UML:**

The UML is intended primarily for software intensive systems. It has been used effectively for such domains as

1. Enterprise Information System
2. Telecommunications
3. Transportation
4. Defense /Aerospace
5. Retails
6. Medical Electronics
7. Scientific Fields
8. Distributed Web

### **Rules of UML:**

The UML has semantic rules for

1. NAMES: It will call things, relationships and diagrams.
2. SCOPE: The content that gives specific meaning to a name.
3. VISIBILITY: How those names can be seen and used by others.
4. INTEGRITY: How things properly and consistently relate to another.
5. EXECUTION: What it means is to run or simulate a dynamic model.

### **Building blocks of UML:**

The vocabulary of the UML encompasses 3 kinds of building blocks

1. Things
2. Relationships
3. Diagrams

### **Things:**

- Things are the data abstractions that are first class citizens in a model. Things are of four types
- Structural Things

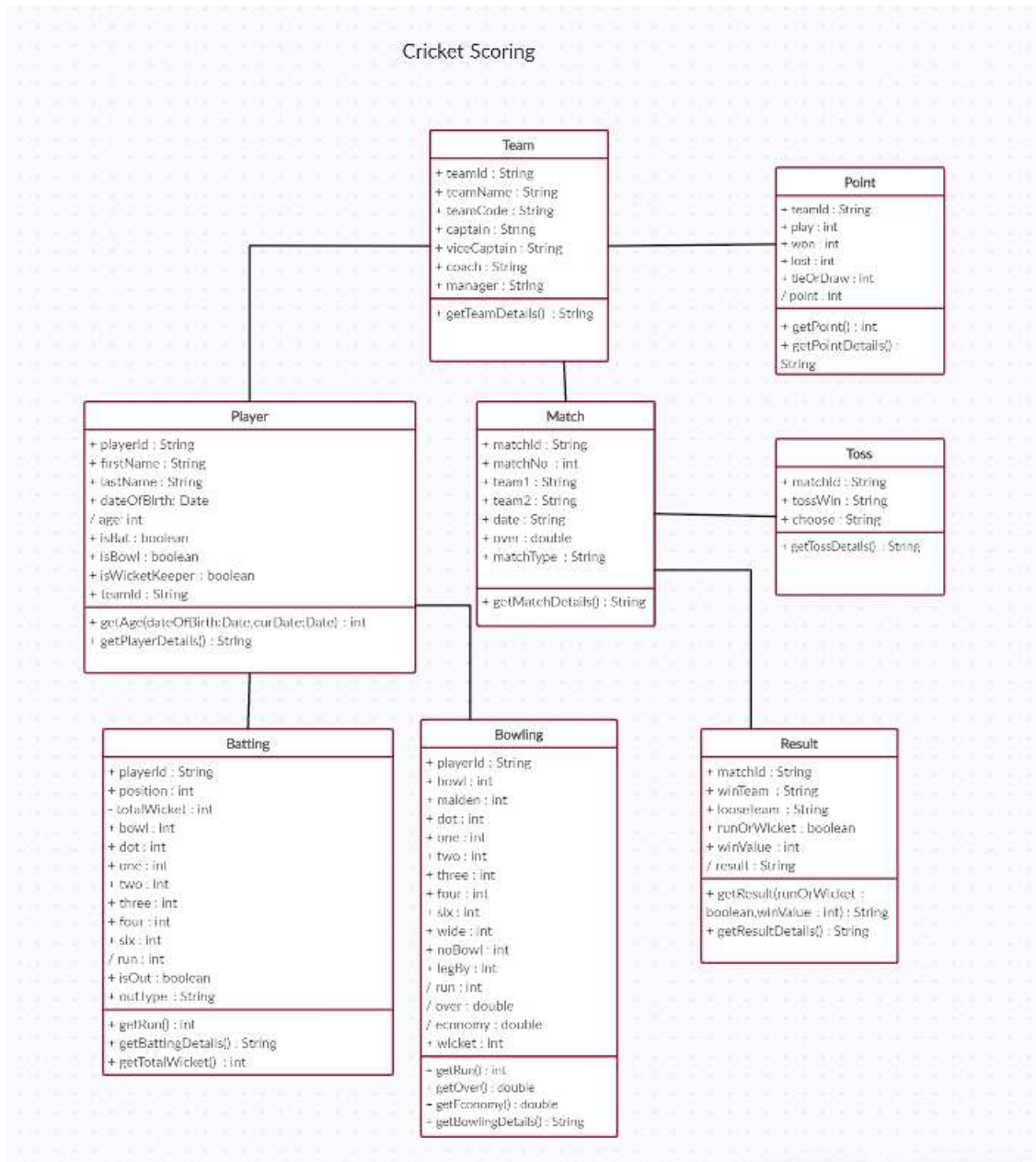
- Behavioural Things
- Grouping Thing
- A notational Things

### **Relationships:**

- Relationships tie the things together. Relationships in the UML are
- Dependency
- Association
- Generalization
- Specialization

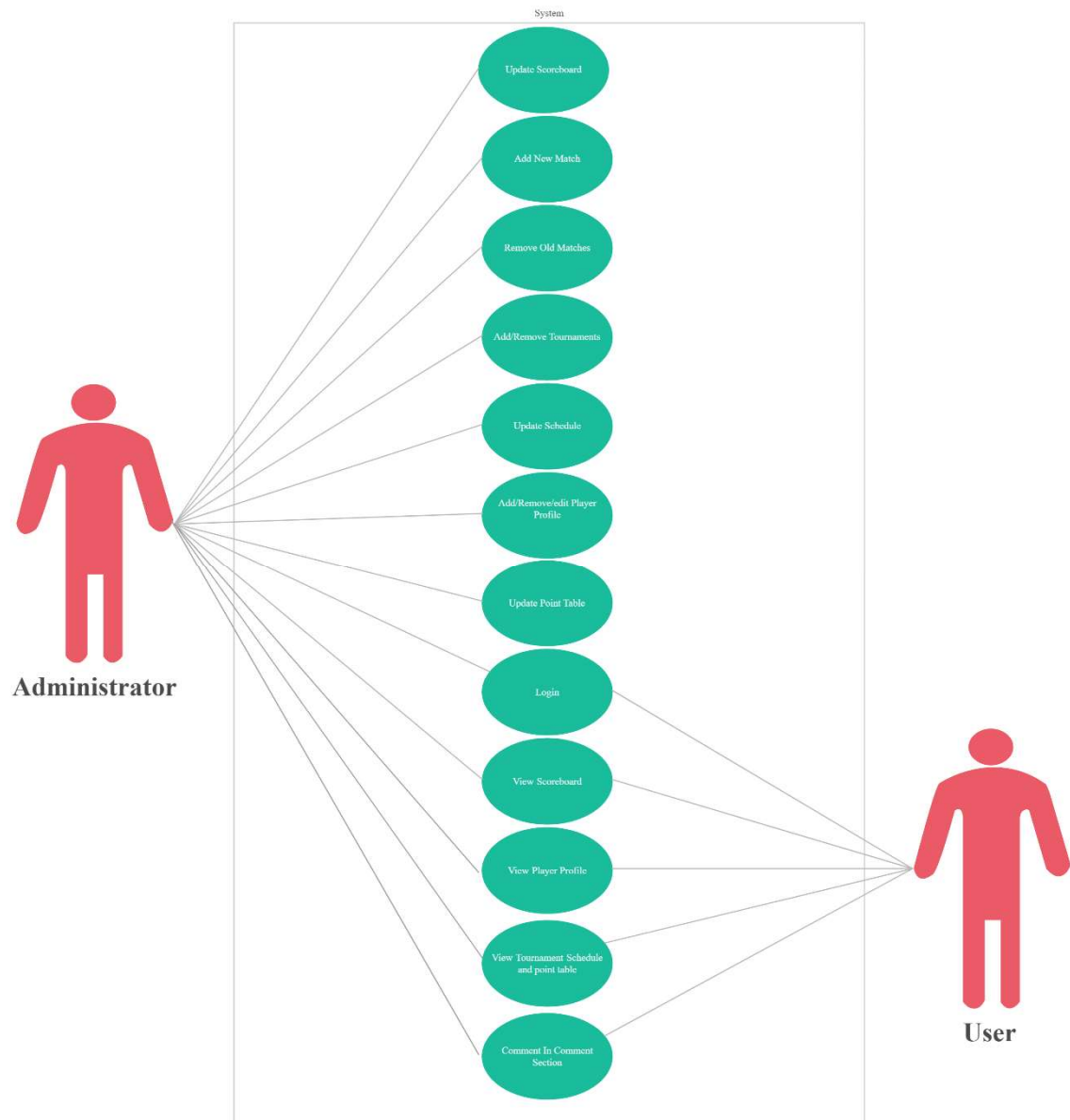
### 3.5.13. Class Diagram:

A class diagram shows a set of classes, interfaces, and collaborations and their relationships. Class diagrams address the static design view of a system. Class diagrams that include active classes address the static process view of a system. A class is a description of a set of objects that share the same attributes, operations, relationships, and semantics. A class implements on or more interfaces.



### 3.5.14. Use Case Diagram:

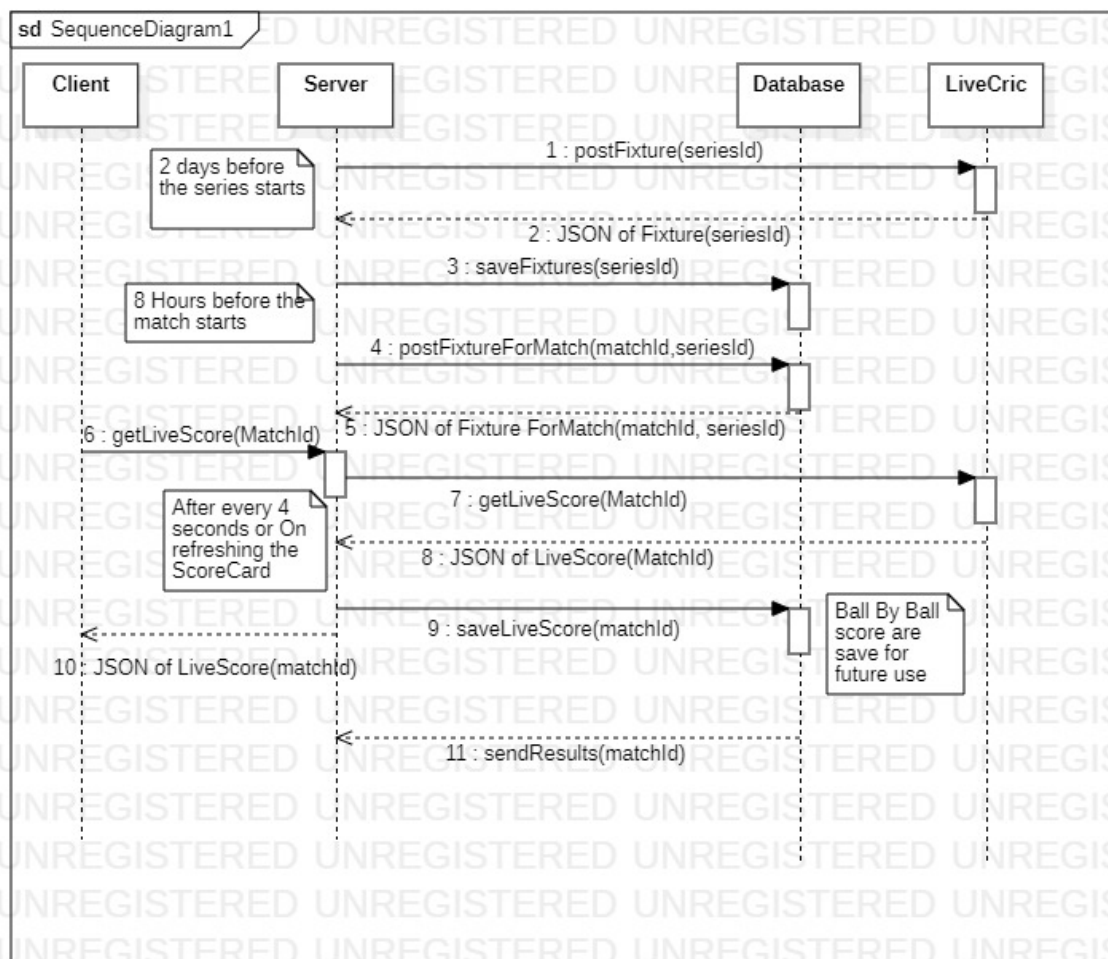
A Use Case diagram shows a set of use cases and actors and their relationships. Use Case diagrams address the static view of a system. These diagrams are especially important in organizing and modelling the behaviours of a system. Use Case diagram consists of use case, actors, and their relationships between them.



### 3.5.15. Sequence Diagram

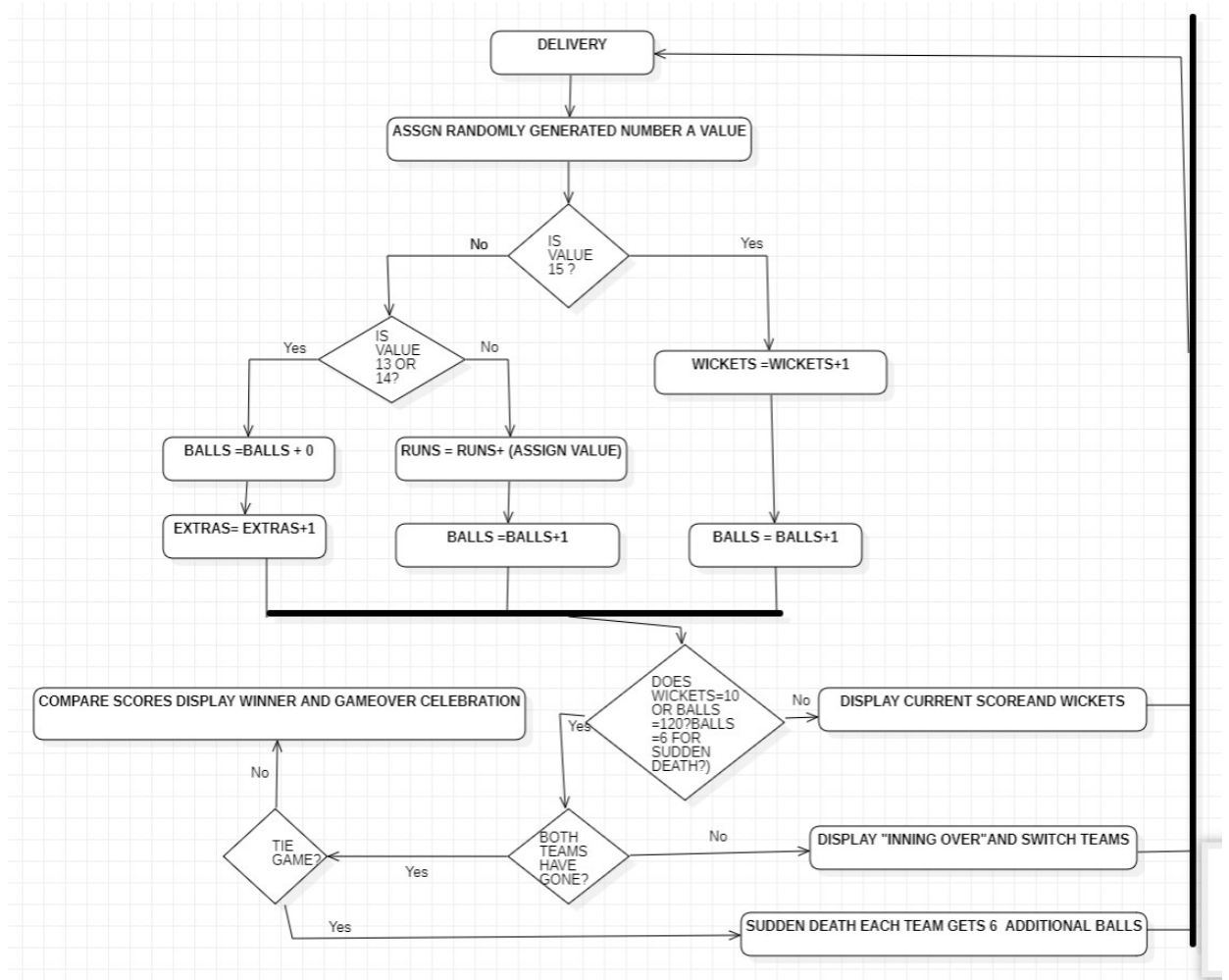
The sequence diagram is an interaction diagram that emphasizes the time ordering of messages for modelling a real time system. Graphically, a sequence diagram is a table that shows objects arranged along the X-axis and messages, ordered in increasing time, along the Y-axis. Sequence Diagram consists of objects, links, lifeline, focus of control, and messages.

- It has two features they are:
- This is the object life time
- There is the focus of control



### 3.5.16. Activity Diagram:

An activity diagram is a special type of state chart diagram. It usually depicts the flow of events within an object. An activity diagram addresses the dynamic view of a system. They are especially important in modeling the function of a system and emphasize the flow of control among objects.





## **Modules**

The project entitled with "Cricket Score Card System" is divided into numerous modules. The detail description about the whole modules will be explained in below. Fig 1 depicts the detail functionality of the Cricket Score Card System with the connection of the modules like Admin, User, Commentary and Review.

**Admin** The admin will update details of upcoming cricket matches, create and block user accounts. The admin module is the major module as it is responsible for carrying out the major operations regarding site updates, score updates etc., It maintains information regarding other modules. The various software components in administrator module update the information about match details, player details. Censoring of comments can be done during or after the match by the admin.

**User Registration** is the main module in this application. The new user has to do the registration process to access the application in online. The registration process includes username, password, address, phone etc.

Once the registration process is completed successfully, then the user can login with the username and password and then search can be easily performed. This module will allow access to all the consumption features of the system, such as listening to official or unofficial commentary, posting comments, viewing scores and cricket statistics.

**Commentary** this module can be used by any user, admin update commentary to particular match for public use. The module will adjust a time itself accurately depending on the score commentary.

Viewers of the commentary may experience a lag which will depend on internet speed for the user and for the commentator.

**Review** In this review module, the user can give the review for the matches. Every user can give review of the match or players through web site. Reports The report is the final stage of the project.

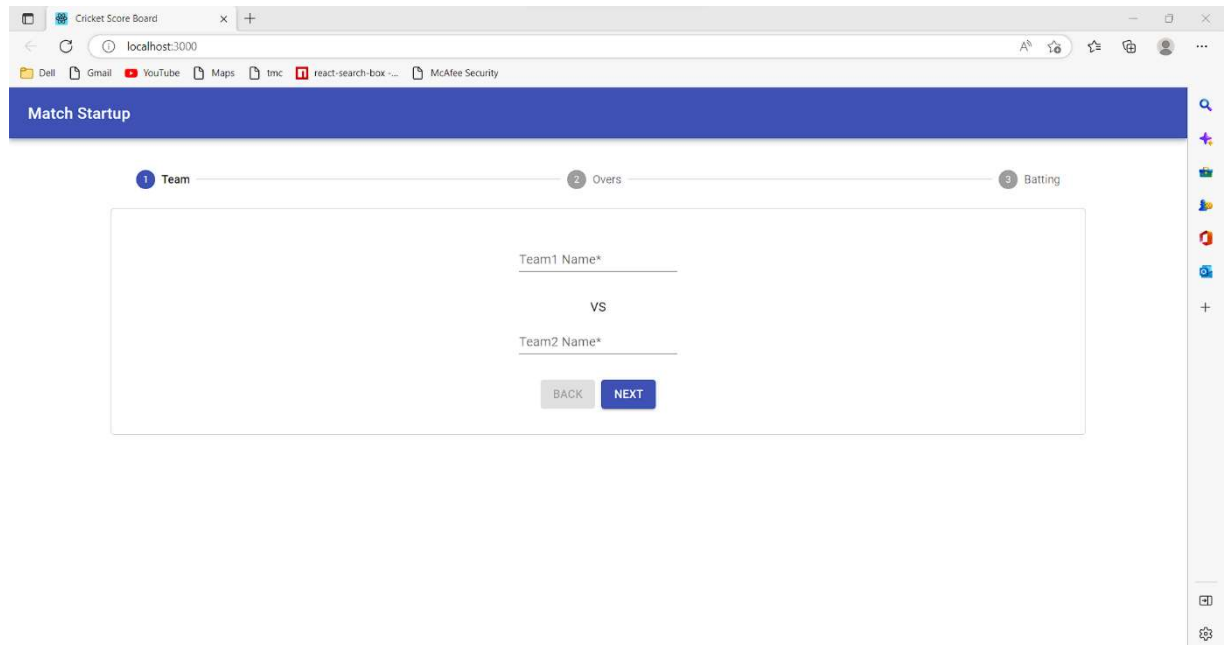
The report can be generated for user details, score details. The admin can view the report accordingly. Through this module the user of the project can view and make report of the entire cricket score card system.

## Chapter – 4

# UI DESIGNS

### Match Startup Page:

On the opening of the web application first of all we reach to the match startup page in we have to enter the team names and then press next to move on to the next page



The screenshot shows a web browser window titled "Cricket Score Board" with the address bar displaying "localhost:3000". The page has a blue header bar with the text "Match Startup". Below the header, there is a progress indicator with three steps: "1 Team", "2 Overs", and "3 Batting". The "1 Team" step is currently active. The main content area is a white box with a light gray border. Inside this box, there are two text input fields labeled "Team1 Name\*" and "Team2 Name\*", separated by "VS". Below the input fields are two buttons: "BACK" (disabled, light gray) and "NEXT" (active, blue). The browser's taskbar at the bottom shows various icons including Dell, Gmail, YouTube, Maps, tmc, react-search-box, and McAfee Security.

## Over Selection Page:

After entering the team name we have to select number of over for the match after that we move on to the toss and we have to select which team is batting first then we move on to the actual scoring User Interface

Cricket Score Board

localhost:3000

Match Startup

✓ Team — ✓ Overs — 3 Batting

Who is Batting?

☐ KKR

☐ MI

BACK START GAME

## Inning Details Page:

Here we can see all of options available for scoring the match we have to enter the 2 batsman Names and 1 Bowler name then we can simply start our scoring by pressing the buttons on the application

If any Batsman gets out a popup window will open and we have few options to select the way the batsman got dismissed

The screenshot shows a web application titled "Cricket Score Board" running on localhost:3000/score. The interface is for the 1st Inning of a match between KKR and MI. The current score is KKR: 0/0 (0) and CRR: 0. The batting team is KKR, and the bowling team is MI. The interface includes a "Batting" section with two batsmen, a "Bowler" section, and a "Recent Overs" section. The "Recent Overs" section shows the current over as 0-0 (0 Ov). The "Score Board" section at the bottom shows the current score and the list of batsmen and bowlers.

Batting	R(B)	4s	6s	SR
<input type="text"/>	0(0)	0	0	0
<input type="text"/>	0(0)	0	0	0

Bowler:

0	1	2	nb	W
3	4	6	wd	

Extras: 0 Wd: 0 NB: 0

Recent Overs

Score Board							
KKR Innings							
Batter	R(B)	4s	6s	SR			
Extras:	0	wd:0	nb:0				
Bowler	0	M	R	W	NB	WD	ECO

## Chapter – 5

### CODING

#### Module Wise

##### 1. HorizontalStepper.js

This Module show the timeline to start a match first we have to entre team names then total match over and who's batting first and which team blowing.

After the toss result decision taken by the captain of the teams. So these code is for create that component on the behalf of the Admin side.

```
import { TextField } from '@material-ui/core'
import Button from '@material-ui/core/Button'
import Step from '@material-ui/core/Step'
import StepLabel from '@material-ui/core/StepLabel'
import Stepper from '@material-ui/core/Stepper'
import { makeStyles } from '@material-ui/core/styles'
import Typography from '@material-ui/core/Typography'
import FormControl from '@mui/material/FormControl'
import FormControlLabel from '@mui/material/FormControlLabel'
import FormLabel from '@mui/material/FormLabel'
import Radio from '@mui/material/Radio'
import RadioGroup from '@mui/material/RadioGroup'
import { Formik } from 'formik'
import React from 'react'
import { useHistory } from 'react-router-dom'
import * as Yup from 'yup'

const useStyles = makeStyles((theme) => ({
  mainContainer: {
    margin: 'auto',
    border: '1px solid #ccc',
    borderRadius: '4px',
  },
  formContainer: {
    margin: '2rem 0 2rem',
    display: 'flex',
    flexDirection: 'column',
    alignItems: 'center',
  },
  formGroup: {
    marginBottom: '2rem',
  },
  resetContainer: {
    display: 'flex',
    flexDirection: 'column',
    justifyContent: 'center',
  },
}))
```

```

    },
    backButton: {
      marginRight: theme.spacing(1),
    },
    instructions: {
      marginTop: theme.spacing(1),
      marginBottom: theme.spacing(1),
    },
    textField: {
      '&> *': {
        margin: theme.spacing(1),
        width: '25ch',
      },
    },
    textFieldWidth: {
      width: 200,
    },
    teamNameHeading: {
      fontWeight: 'bold',
      // padding: '16px',
      // paddingBottom: '0px',
    },
    center: {
      textAlign: 'center',
    },
  )))
const HorizontalStepper = () => {
  const history = useHistory()
  const classes = useStyles()
  const [activeStep, setActiveStep] = React.useState(0)
  const [isSubmitting, setSubmitting] = React.useState(false)

  const steps = ['Team', 'Overs', 'Batting']
  const handleNext = () => {
    setActiveStep((prevActiveStep) => prevActiveStep + 1)
  }
  const handleBack = () => {
    setActiveStep((prevActiveStep) => prevActiveStep - 1)
  }
  const initialValues = {
    team1: "",
    team2: "",
    maxOver: "",
    batting: "",
  }
  const validationSchema = [
    Yup.object().shape({
      team1: Yup.string().required('Team Name is required'),
      team2: Yup.string().required('Team Name is required'),
    }),
    Yup.object().shape({
      maxOver: Yup.string().required('Over is required'),
    }),
  ]

```

```

    Yup.object().shape({
      batting: Yup.string().required('Please choose who is Batting'),
    }),
  ],
  const currentValidationSchema = validationSchema[activeStep]
  function isLastStep() {
    return activeStep === steps.length - 1
  }
  return (
    <div>
    <Stepper activeStep={activeStep} orientation='horizontal'>
      {steps.map((label) => (
        <Step key={label}>
        <StepLabel>{label}</StepLabel>
        </Step>
      ))}
    </Stepper>
    <div className={classes.mainContainer}>
    <Formik
      enableReinitialize
      validationSchema={currentValidationSchema}
      initialValues={initialValues}
      onSubmit={(values, actions) => {
        handleNext()
        actions.setTouched({})
        actions.setSubmitting(false)
        if (isLastStep()) {
          setSubmitting(true)
          const data = JSON.stringify(values)
          localStorage.setItem('data', data)
          history.push('/score')
          setSubmitting(false)
        }
      }}
    >
      {(prp) => {
        const { values, touched, errors, handleChange, handleBlur, handleSubmit, setFieldValue } = prp
        return (
          <form onSubmit={handleSubmit}>
          <div className={classes.formContainer}>
            {activeStep === 0 && (
              <div>
              <div className={classes.formGroup}>
              <TextField
                id='team1'
                name='team1'
                label='Team1 Name*'
                value={values.team1}
                onChange={handleChange}
                onBlur={handleBlur}
                helperText={errors.team1 && touched.team1 && errors.team1}
                error={errors.team1 && touched.team1}
                className={classes.textfieldWidth}

```

```

        />
    </div>
    <div>
    <Typography className={classes.center}>VS</Typography>
    </div>
    <div className={classes.formGroup}>
    <TextField
        id='team2'
        name='team2'
        label='Team2 Name*'
        value={values.team2}
        onChange={handleChange}
        onBlur={handleBlur}
        helperText={errors.team2 && touched.team2 && errors.team2}
        error={errors.team2 && touched.team2}
        className={classes.textfieldWidth}
    />
    </div>
    </div>
    )}
    {activeStep === 1 && (
    <div>
    <div className={classes.formGroup} id='team1-players'>
    <Typography className={classes.teamNameHeading}>How many overs?</Typography>
    <div className={classes.formGroup}>
    <TextField
        id='maxOver'
        name='maxOver'
        label='Overs*'
        value={values.maxOver}
        onChange={handleChange}
        onBlur={handleBlur}
        helperText={errors.maxOver && touched.maxOver && errors.maxOver}
        error={errors.maxOver && touched.maxOver}
        className={classes.textfieldWidth}
    />
    </div>
    </div>
    </div>
    )}
    {activeStep === 2 && (
    <div>
    <div className={classes.formGroup}>
    <FormControl component='fieldset'>
    <FormLabel component='legend'>Who is Batting?</FormLabel>
    <RadioGroup
        name='batting'
        value={values.batting.toString()}
        onChange={(event) => {
            setFieldValue('batting', event.currentTarget.value)
        }}
    >
    <FormControlLabel value={values.team1} control={}<Radio /> label={values.team1} />

```



```

<FormControlLabel value={values.team2} control={<Radio />} label={values.team2} />
</RadioGroup>
</FormControl>
</div>
</div>
    )}
<div>
<Button variant='contained' disabled={activeStep === 0} onClick={handleBack}
className={classes.backButton}>
    Back
</Button>
<Button id='submit' disabled={isSubmitting} variant='contained' color='primary' type='submit'>
    {isLastStep() ? 'Start Game' : 'Next'}
</Button>
</div>
</div>
</form>
    )
  }}
</Formik>
</div>
</div>
)
}

```

```
export default HorizontalStepper
```

## 2. ScoreBoard.js

This Modules shows the score board of a cricket match which consists wicket runs overs batsman name bowler name economy strike rate etc. and contain the details about the current match score how many boundary, run, single, with balls, no ball, lbw are coming in the Inning of both side team.

```

import DeleteIcon from '@mui/icons-material/Delete'
import EditIcon from '@mui/icons-material/Edit'
import { IconButton } from '@mui/material'
import Box from '@mui/material/Box'
import { pink } from '@mui/material/colors'
import FormControl from '@mui/material/FormControl'
import FormControlLabel from '@mui/material/FormControlLabel'
import Modal from '@mui/material/Modal'
import Radio from '@mui/material/Radio'
import RadioGroup from '@mui/material/RadioGroup'
import React, { useEffect, useState } from 'react'
import { useHistory } from 'react-router-dom'
import Autosuggest from 'react-autosuggest'
import { BATTING, OUT } from '../constants/BattingStatus'
import { BOLD, CATCH, HIT_WICKET, RUN_OUT, STUMP } from '../constants/OutType'

```

```

import MathUtil from '../util/MathUtil'
import './ScoreBoard.css'
import { radioGroupBoxStyle } from './ui/RadioGroupBoxStyle'

const ScoreBoard = () => {
  const [inningNo, setInningNo] = useState(1)
  const [match, setMatch] = useState({ inning1: { batters: [], bowlers: [] }, inning2: { batters: [], bowlers: [] } })
  const [currentRunStack, setCurrentRunStack] = useState([])
  const [totalRuns, setTotalRuns] = useState(0)
  const [extras, setExtras] = useState({ total: 0, wide: 0, noBall: 0 })
  const [runsByOver, setRunsByOver] = useState(0)
  const [wicketCount, setWicketCount] = useState(0)
  const [totalOvers, setTotalOvers] = useState(0)
  const [batters, setBatters] = useState([])
  const [ballCount, setBallCount] = useState(0)
  const [overCount, setOverCount] = useState(0)
  const [recentOvers, setRecentOvers] = useState([])
  const [batter1, setBatter1] = useState({})
  const [batter2, setBatter2] = useState({})
  const [battingOrder, setBattingOrder] = useState(0)
  const [isBatter1Edited, setBatter1Edited] = useState(false)
  const [isBatter2Edited, setBatter2Edited] = useState(false)
  const [isBowlerEdited, setBowlerEdited] = useState(false)
  const [bowler, setBowler] = useState({})
  const [bowlers, setBowlers] = useState([])
  const [inputBowler, setInputBowler] = useState('')
  const [isModalOpen, setModalOpen] = React.useState(false)
  const [outType, setOutType] = React.useState('')
  const [runOutPlayerId, setRunOutPlayerId] = React.useState('')
  const [remainingBalls, setRemainingBalls] = useState(0)
  const [remainingRuns, setRemainingRuns] = useState(0)
  const [strikeValue, setStrikeValue] = React.useState('strike')
  const [isNoBall, setNoBall] = useState(false)
  const [suggestions, setSuggestions] = useState([])
  const [hasNameSuggested, setNameSuggested] = useState(false)
  const [hasMatchEnded, setMatchEnded] = useState(false)

  let data = JSON.parse(localStorage.getItem('data'))
  const { batting, team1, team2 } = data
  const maxOver = parseInt(data.maxOver)
  const history = useHistory()

  useEffect(() => {
    const endInningButton = document.getElementById('end-inning')
    endInningButton.disabled = true
  }, [])

  const handleEndInning = (e) => {
    const endInningButton = document.getElementById('end-inning')

```

```

if (endInningButton.textContent === 'Reset') {
  history.push('/')
} else {
  if (batter1.id !== undefined) {
    const { id, name, run, ball, four, six, strikeRate, onStrike } = batter1
    batters.push({
      id,
      name,
      run,
      ball,
      four,
      six,
      strikeRate,
      onStrike,
      battingOrder: batter1.battingOrder,
      battingStatus: BATTING,
    })
  }
  if (batter2.id !== undefined) {
    batters.push({
      id: batter2.id,
      name: batter2.name,
      run: batter2.run,
      ball: batter2.ball,
      four: batter2.four,
      six: batter2.six,
      strikeRate: batter2.strikeRate,
      onStrike: batter2.onStrike,
      battingOrder: batter2.battingOrder,
      battingStatus: BATTING,
    })
  }
  if (bowler.id !== undefined) {
    const currentDisplayOver = Math.round((ballCount === 6 ? 1 : ballCount * 0.1) * 10) / 10
    let isMaidenOver = true
    let countWicket = 0
    let countNoBall = 0
    let countWide = 0
    const deliveries = ['1', '2', '3', '4', '6', 'wd']
    for (let delivery of currentRunStack) {
      delivery = delivery.toString()
      if (deliveries.includes(delivery) || delivery.includes('nb')) {
        isMaidenOver = false
      }
      if (delivery === 'W') {
        countWicket++
      }
      if (delivery.includes('nb')) {
        countNoBall++
      }
    }
  }
}

```

```

    if (delivery.includes('wd')) {
      countWide++
    }
  }
  if (ballCount !== 6) {
    isMaidenOver = false
  }
  const index = bowlers.findIndex((blr) => {
    return blr.id === bowler.id
  })
  if (index !== -1) {
    const existingBowler = bowlers[index]
    const { maiden, wicket, noBall, wide, over } = existingBowler
    const bowlerTotalOver = over + ballCount / 6
    existingBowler.over = existingBowler.over + currentDisplayOver
    existingBowler.maiden = isMaidenOver ? maiden + 1 : maiden
    existingBowler.run = existingBowler.run + runsByOver
    existingBowler.wicket = wicket + countWicket
    existingBowler.noBall = noBall + countNoBall
    existingBowler.wide = wide + countWide
    existingBowler.economy = Math.round((existingBowler.run / bowlerTotalOver) * 100) / 100
    bowlers[index] = existingBowler
    setBowlers(bowlers)
  } else {
    if (ballCount !== 6) {
      bowlers.push({
        id: bowler.id,
        name: bowler.name,
        over: currentDisplayOver,
        maiden: isMaidenOver ? 1 : 0,
        run: runsByOver,
        wicket: countWicket,
        noBall: countNoBall,
        wide: countWide,
        economy: Math.round((runsByOver / (ballCount / 6)) * 100) / 100,
      })
      setBowlers(bowlers)
    }
  }
}

if (inningNo === 1) {
  setMatch((state) => {
    const totalFours = batters.map((batter) => batter.four).reduce((prev, next) => prev + next)
    const totalSixes = batters.map((batter) => batter.four).reduce((prev, next) => prev + next)
    return {
      ...state,
      inning1: {
        runs: totalRuns,
        wickets: wicketCount,
        runRate: crr,
      },
    }
  })
}

```

```

        overs: totalOvers,
        four: totalFours,
        six: totalSixes,
        extra: extras,
        batters,
        bowlers,
      },
    }
  })
  setInningNo(2)
  setCurrentRunStack([])
  setTotalRuns(0)
  setExtras({ total: 0, wide: 0, noBall: 0 })
  setRunsByOver(0)
  setWicketCount(0)
  setTotalOvers(0)
  setBallCount(0)
  setOverCount(0)
  setRecentOvers([])
  setBatter1({})
  setBatter2({})
  setBatters([])
  setBowlers([])
  setBattingOrder(0)
  setInputBowler('')
  setBowler({})
  setRemainingBalls(maxOver * 6)
  setRemainingRuns(totalRuns + 1)
  const bowlerNameElement = document.querySelector('.react-autosuggest__input')
  bowlerNameElement.disabled = false
  const batter1NameElement = document.getElementById('batter1Name')
  batter1NameElement.value = ''
  batter1NameElement.disabled = false
  const batter2NameElement = document.getElementById('batter2Name')
  batter2NameElement.value = ''
  batter2NameElement.disabled = false
  setStrikeValue('strike')
  endInningButton.disabled = true
} else {
  setMatch((state) => {
    const totalFours = batters.map((batter) => batter.four).reduce((prev, next) => prev + next)
    const totalSixes = batters.map((batter) => batter.four).reduce((prev, next) => prev + next)
    return {
      ...state,
      inning2: {
        runs: totalRuns,
        wickets: wicketCount,
        runRate: crr,
        overs: totalOvers,
        four: totalFours,

```

```

        six: totalSixes,
        extra: extras,
        batters,
        bowlers,
      },
    }
  })
  endInningButton.textContent = 'Reset'
  setMatchEnded(true)
}
}
}
const handleBatter1Blur = (e) => {
  let name = e.target.value
  name = name.charAt(0).toUpperCase() + name.slice(1)
  e.target.value = name
  e.target.disabled = true
  if (isBatter1Edited) {
    setBatter1((state) => ({
      ...state,
      name: name,
    })))
    setBatter1Edited(false)
  } else {
    const randomNo = MathUtil.getRandomNo()
    setBatter1({
      id: name + randomNo,
      name: name,
      run: 0,
      ball: 0,
      four: 0,
      six: 0,
      strikeRate: 0,
      onStrike: strikeValue === 'strike' ? true : false,
      battingOrder: battingOrder + 1,
      battingStatus: BATTING,
    })
    setBattingOrder(battingOrder + 1)
  }
}
const handleBatter2Blur = (e) => {
  let name = e.target.value
  name = name.charAt(0).toUpperCase() + name.slice(1)
  e.target.value = name
  e.target.disabled = true
  if (isBatter2Edited) {
    setBatter2((state) => ({
      ...state,
      name: name,
    })))
  }
}

```

```

    setBatter2Edited(false)
  } else {
    const randomNo = MathUtil.getRandomNo()
    setBatter2({
      id: name + randomNo,
      name: name,
      run: 0,
      ball: 0,
      four: 0,
      six: 0,
      strikeRate: 0,
      onStrike: strikeValue === 'non-strike' ? true : false,
      battingOrder: battingOrder + 1,
      battingStatus: BATTING,
    })
    setBattingOrder(battingOrder + 1)
  }
}
const handleBowlerBlur = (e) => {
  let name = e.target.value
  if (name !== '') {
    name = name.charAt(0).toUpperCase() + name.slice(1)
    setInputBowler(name)
    e.target.value = name
    e.target.disabled = true
    if (isBowlerEdited) {
      setBowler((state) => ({
        ...state,
        name: name,
      }))
      setBowlerEdited(false)
    } else {
      if (hasNameSuggested) {
        setNameSuggested(false)
      } else {
        const randomNo = MathUtil.getRandomNo()
        const id = name + randomNo
        setBowler({
          id,
          name,
        })
      }
    }
  }
}
const onSuggestionsFetchRequested = (param) => {
  const inputValue = param.value.trim().toLowerCase()
  const suggestionArr = inputValue.length === 0 ? [] : bowlers.filter((bowlerObj) =>
    bowlerObj.name.toLowerCase().includes(inputValue))
  setSuggestions(suggestionArr)
}

```

```

    }
    const getSuggestionValue = (suggestion) => {
      setBowler({
        id: suggestion.id,
        name: suggestion.name,
      })
      setNameSuggested(true)
      return suggestion.name
    }
    const inputProps = {
      value: inputBowler,
      onChange: (e, { newValue }) => {
        setInputBowler(newValue)
      },
      onBlur: handleBowlerBlur,
    }
    const overCompleted = (runsByOverParam, currentRunStackParam) => {
      const bowlerNameElement = document.querySelector('.react-autosuggest__input')
      if (overCount + 1 === maxOver) {
        const endInningButton = document.getElementById('end-inning')
        endInningButton.disabled = false
      } else {
        bowlerNameElement.disabled = false
      }
      disableAllScoreButtons()
      setRecentOvers((state) => [
        ...state,
        { overNo: overCount + 1, bowler: bowler.name, runs: runsByOverParam, stack:
currentRunStackParam },
      ])
      setInputBowler('')
      setBowler({})
      setCurrentRunStack([])
      setRunsByOver(0)
      setBallCount(0)
      setOverCount(overCount + 1)
      const index = bowlers.findIndex((blr) => blr.id === bowler.id)
      let isMaidenOver = true
      let countWicket = 0
      let countNoBall = 0
      let countWide = 0
      const deliveries = ['1', '2', '3', '4', '6', 'wd']
      for (let delivery of currentRunStackParam) {
        delivery = delivery.toString()
        if (deliveries.includes(delivery) || delivery.includes('nb')) {
          isMaidenOver = false
        }
        if (delivery === 'W') {
          countWicket++
        }
      }
    }
  }

```



```

    if (delivery.includes('nb')) {
      countNoBall++
    }
    if (delivery.includes('wd')) {
      countWide++
    }
  }
  if (index !== -1) {
    const existingBowler = bowlers[index]
    const { over, maiden, run, wicket, noBall, wide } = existingBowler
    existingBowler.over = over + 1
    existingBowler.maiden = isMaidenOver ? maiden + 1 : maiden
    existingBowler.run = run + runsByOverParam
    existingBowler.wicket = wicket + countWicket
    existingBowler.noBall = noBall + countNoBall
    existingBowler.wide = wide + countWide
    existingBowler.economy = Math.round((existingBowler.run / existingBowler.over) * 100) / 100
    bowlers[index] = existingBowler
    setBrowsers(browsers)
  } else {
    setBrowsers((state) => [
      ...state,
      {
        id: bowler.id,
        name: bowler.name,
        over: 1,
        maiden: isMaidenOver ? 1 : 0,
        run: runsByOverParam,
        wicket: countWicket,
        noBall: countNoBall,
        wide: countWide,
        economy: runsByOverParam,
      },
    ])
  }
}

const newBatter1 = () => {
  const batter1NameElement = document.getElementById('batter1Name')
  batter1NameElement.value = ''
  batter1NameElement.disabled = false
  const { id, name, run, ball, four, six, strikeRate, onStrike } = batter1
  setBatters((state) => [
    ...state,
    {
      id,
      name,
      run,
      ball,
      four,
      six,

```

```

        strikeRate,
        onStrike,
        battingOrder: batter1.battingOrder,
        battingStatus: OUT,
      },
    ])
    setBatter1({})
  }
  const newBatter2 = () => {
    const batter2NameElement = document.getElementById('batter2Name')
    batter2NameElement.value = ''
    batter2NameElement.disabled = false
    const { id, name, run, ball, four, six, strikeRate, onStrike } = batter2
    setBatters((state) => [
      ...state,
      {
        id,
        name,
        run,
        ball,
        four,
        six,
        strikeRate,
        onStrike,
        battingOrder: batter2.battingOrder,
        battingStatus: OUT,
      },
    ])
    setBatter2({})
  }
  const editBatter1Name = () => {
    if (overCount !== maxOver && wicketCount !== 10 && !hasMatchEnded) {
      const batter1NameElement = document.getElementById('batter1Name')
      batter1NameElement.disabled = false
      setBatter1Edited(true)
    }
  }
  const editBatter2Name = () => {
    if (overCount !== maxOver && wicketCount !== 10 && !hasMatchEnded) {
      const batter2NameElement = document.getElementById('batter2Name')
      batter2NameElement.disabled = false
      setBatter2Edited(true)
    }
  }
  const editBowlerName = () => {
    if (overCount !== maxOver && wicketCount !== 10 && !hasMatchEnded) {
      const bowlerNameElement = document.querySelector('.react-autosuggest__input')
      bowlerNameElement.disabled = false
      setBowlerEdited(true)
    }
  }

```

```

}
const undoWicket = (isNoBallParam) => {
  if (!isNoBallParam) {
    setBallCount(ballCount - 1)
    setTotalOvers(Math.round((totalOvers - 0.1) * 10) / 10)
  }
  setWicketCount(wicketCount - 1)
  const batter = batters[batters.length - 1]
  const { id, name, run, ball, four, six, strikeRate, onStrike } = batter
  if (batter1.name === undefined || batter1.onStrike) {
    const batter1NameElement = document.getElementById('batter1Name')
    batter1NameElement.value = batter.name
    batter1NameElement.disabled = true
    setBatter1({
      id,
      name,
      run,
      ball,
      four,
      six,
      strikeRate,
      onStrike,
      battingOrder: batter.battingOrder,
      battingStatus: BATTING,
    })
    if (!batter.onStrike) {
      changeStrikeRadio()
      setBatter2((state) => ({
        ...state,
        onStrike: true,
      })))
    }
  } else if (batter2.name === undefined || batter2.onStrike) {
    const batter2NameElement = document.getElementById('batter2Name')
    batter2NameElement.value = batter.name
    batter2NameElement.disabled = true
    setBatter2({
      id,
      name,
      run,
      ball,
      four,
      six,
      strikeRate,
      onStrike,
      battingOrder: batter.battingOrder,
      battingStatus: BATTING,
    })
    if (!batter.onStrike) {
      changeStrikeRadio()
    }
  }
}

```

```

    setBatter1((state) => ({
      ...state,
      onStrike: true,
    })))
  }
}
batters.pop()
setBatters(batters)
}
const undoRun = (run, isNoBallParam) => {
  if (isNoBallParam) {
    setTotalRuns(totalRuns - (run + 1))
    setRunsByOver(runsByOver - (run + 1))
  } else {
    setTotalRuns(totalRuns - run)
    setRunsByOver(runsByOver - run)
    setBallCount(ballCount - 1)
    setTotalOvers(Math.round((totalOvers - 0.1) * 10) / 10)
    currentRunStack.pop()
    setCurrentRunStack(currentRunStack)
  }
  if (batter1.onStrike) {
    if (run % 2 === 0) {
      setBatter1((state) => {
        const updatedRun = state.run - run
        const updatedBall = state.ball - 1
        const updatedSr = updatedRun / updatedBall
        const sr = Math.round(isNaN(updatedSr) ? 0 : updatedSr * 100 * 100) / 100
        let four = state.four
        if (run === 4) {
          four = four - 1
        }
        let six = state.six
        if (run === 6) {
          six = six - 1
        }
        return {
          ...state,
          run: updatedRun,
          ball: updatedBall,
          four: four,
          six: six,
          strikeRate: sr,
        }
      })
    } else {
      changeStrikeRadio()
      switchBatterStrike()
      setBatter2((state) => {
        const updatedRun = state.run - run

```

```

const updatedBall = state.ball - 1
const updatedSr = updatedRun / updatedBall
const sr = Math.round(isNaN(updatedSr) ? 0 : updatedSr * 100 * 100) / 100
let four = state.four
if (run === 4) {
  four = four - 1
}
let six = state.six
if (run === 6) {
  six = six - 1
}
return {
  ...state,
  run: updatedRun,
  ball: updatedBall,
  four: four,
  six: six,
  strikeRate: sr,
}
})
}
} else if (batter2.onStrike) {
  if (run % 2 === 0) {
    setBatter2((state) => {
      const updatedRun = state.run - run
      const updatedBall = state.ball - 1
      const updatedSr = updatedRun / updatedBall
      const sr = Math.round(isNaN(updatedSr) ? 0 : updatedSr * 100 * 100) / 100
      let four = state.four
      if (run === 4) {
        four = four - 1
      }
      let six = state.six
      if (run === 6) {
        six = six - 1
      }
      return {
        ...state,
        run: updatedRun,
        ball: updatedBall,
        four: four,
        six: six,
        strikeRate: sr,
      }
    })
  } else {
    changeStrikeRadio()
    switchBatterStrike()
    setBatter1((state) => {
      const updatedRun = state.run - run

```

```

const updatedBall = state.ball - 1
const updatedSr = updatedRun / updatedBall
const sr = Math.round(isNaN(updatedSr) ? 0 : updatedSr * 100 * 100) / 100
let four = state.four
if (run === 4) {
  four = four - 1
}
let six = state.six
if (run === 6) {
  six = six - 1
}
return {
  ...state,
  run: updatedRun,
  ball: updatedBall,
  four: four,
  six: six,
  strikeRate: sr,
}
})
}
}
}
const undoDelivery = () => {
  if (currentRunStack.length > 0) {
    const last = currentRunStack[currentRunStack.length - 1]
    if (typeof last === 'number') {
      const run = parseInt(last)
      undoRun(run, false)
    } else {
      currentRunStack.pop()
      setCurrentRunStack(currentRunStack)
      if (last === 'wd') {
        setTotalRuns(totalRuns - 1)
        setExtras((state) => ({
          ...state,
          total: state.total - 1,
          wide: state.wide - 1,
        })))
      } else if (last === 'W') {
        undoWicket(false)
      } else {
        const lastChar = last.substr(last.length - 1)
        const run = parseInt(lastChar)
        if (isNaN(run)) {
          setTotalRuns(totalRuns - 1)
          setRunsByOver(runsByOver - 1)
          if (last !== 'nb') {
            undoWicket(true)
          }
        }
      }
    }
  }
}

```

```

    } else {
      undoRun(run, true)
    }
  }
}
}
}

const handleStrikeChange = (e) => {
  changeStrikeRadio(e.target.value)
  if (e.target.value === 'strike') {
    switchBatterStrike('batter1')
  } else {
    switchBatterStrike('batter2')
  }
}

const changeStrikeRadio = (strikeParam) => {
  if (strikeParam === undefined) {
    setStrikeValue(strikeValue === 'strike' ? 'non-strike' : 'strike')
  } else {
    setStrikeValue(strikeParam)
  }
}

const switchBatterStrike = (strikeParam) => {
  if (strikeParam === undefined) {
    setBatter1((state) => ({
      ...state,
      onStrike: !state.onStrike,
    }))
    setBatter2((state) => ({
      ...state,
      onStrike: !state.onStrike,
    }))
  } else {
    if (strikeParam === 'batter1') {
      setBatter1((state) => ({
        ...state,
        onStrike: true,
      }))
      setBatter2((state) => ({
        ...state,
        onStrike: false,
      }))
    } else if (strikeParam === 'batter2') {
      setBatter1((state) => ({
        ...state,
        onStrike: false,
      }))
      setBatter2((state) => ({
        ...state,
        onStrike: true,
      }))
    }
  }
}

```

```

    )))
  }
}
}
const handleRun = (run) => {
  if (isNoBall) {
    setCurrentRunStack((state) => [...state, 'nb' + run])
    removeNoBallEffect()
  } else {
    setBallCount(ballCount + 1)
    setCurrentRunStack((state) => [...state, run])
  }
  setTotalRuns(totalRuns + run)
  setRunsByOver(runsByOver + run)
  if (inningNo === 2) {
    if (!isNoBall) {
      setRemainingBalls(remainingBalls - 1)
    }
    setRemainingRuns(remainingRuns - run)
  }
  if (ballCount === 5) {
    if (isNoBall) {
      if (run % 2 !== 0) {
        changeStrikeRadio()
      }
    } else {
      setTotalOvers(overCount + 1)
      const arr = [...currentRunStack]
      arr.push(run)
      overCompleted(runsByOver + run, arr)
      if (run % 2 === 0) {
        changeStrikeRadio()
      }
    }
  } else {
    if (!isNoBall) {
      setTotalOvers(Math.round((totalOvers + 0.1) * 10) / 10)
    }
    if (run % 2 !== 0) {
      changeStrikeRadio()
    }
  }
  if (batter1.onStrike) {
    setBatter1((state) => {
      const updatedRun = state.run + run
      const updatedBall = state.ball + 1
      const sr = Math.round((updatedRun / updatedBall) * 100 * 100) / 100
      let four = state.four
      if (run === 4) {
        four = four + 1
      }
    })
  }
}

```



```

    }
    let six = state.six
    if (run === 6) {
      six = six + 1
    }
    return {
      ...state,
      run: updatedRun,
      ball: updatedBall,
      four: four,
      six: six,
      strikeRate: sr,
    }
  })
  if (isNoBall) {
    if (run % 2 !== 0) {
      switchBatterStrike()
    }
  } else {
    if ((ballCount === 5 && run % 2 === 0) || (ballCount !== 5 && run % 2 !== 0)) {
      switchBatterStrike()
    }
  }
} else {
  setBatter2((state) => {
    const updatedRun = state.run + run
    const updatedBall = state.ball + 1
    const sr = Math.round((updatedRun / updatedBall) * 100 * 100) / 100
    let four = state.four
    if (run === 4) {
      four = four + 1
    }
    let six = state.six
    if (run === 6) {
      six = six + 1
    }
    return {
      ...state,
      run: updatedRun,
      ball: updatedBall,
      four: four,
      six: six,
      strikeRate: sr,
    }
  })
  if ((ballCount === 5 && run % 2 === 0) || (ballCount !== 5 && run % 2 !== 0)) {
    switchBatterStrike()
  }
}
}
}

```

```

const handleNoBall = () => {
  if (inningNo === 2) {
    setRemainingRuns(remainingRuns - 1)
  }
  setTotalRuns(totalRuns + 1)
  setRunsByOver(runsByOver + 1)
  setExtras((state) => ({
    ...state,
    total: state.total + 1,
    noBall: state.noBall + 1,
  })))
  addNoBallEffect()
}

const addNoBallEffect = () => {
  const scoreTypesButtons = document.querySelectorAll('.score-types-button')
  for (let i = 0; i < scoreTypesButtons.length; i++) {
    scoreTypesButtons[i].classList.add('score-types-button-noball')
  }
  setNoBall(true)
}

const removeNoBallEffect = () => {
  const scoreTypesButtons = document.querySelectorAll('.score-types-button')
  for (let i = 0; i < scoreTypesButtons.length; i++) {
    scoreTypesButtons[i].classList.remove('score-types-button-noball')
  }
  setNoBall(false)
}

const handleWide = () => {
  if (isNoBall) {
    setCurrentRunStack((state) => [...state, 'nb'])
    removeNoBallEffect()
  } else {
    if (inningNo === 2) {
      setRemainingRuns(remainingRuns - 1)
    }
    setCurrentRunStack((state) => [...state, 'wd'])
    setTotalRuns(totalRuns + 1)
    setRunsByOver(runsByOver + 1)
    setExtras((state) => ({
      ...state,
      total: state.total + 1,
      wide: state.wide + 1,
    })))
  }
}

const handleWicket = (isRunOut, playerId) => {
  setRunOutPlayerId("")
  if (ballCount === 5) {
    if (isNoBall) {
      removeNoBallEffect()
    }
  }
}

```

```

    if (isRunOut) {
      setCurrentRunStack((state) => [...state, 'nbW'])
      setWicketCount(wicketCount + 1)
      disableAllScoreButtons()
    } else {
      setCurrentRunStack((state) => [...state, 'nb'])
    }
  } else {
    setTotalOvers(overCount + 1)
    const arr = [...currentRunStack]
    arr.push('W')
    overCompleted(runsByOver, arr)
    setWicketCount(wicketCount + 1)
    disableAllScoreButtons()
  }
} else {
  if (isNoBall) {
    removeNoBallEffect()
    if (isRunOut) {
      setCurrentRunStack((state) => [...state, 'nbW'])
      setWicketCount(wicketCount + 1)
      disableAllScoreButtons()
    } else {
      setCurrentRunStack((state) => [...state, 'nb'])
    }
  } else {
    setBallCount(ballCount + 1)
    setCurrentRunStack((state) => [...state, 'W'])
    setTotalOvers(Math.round((totalOvers + 0.1) * 10) / 10)
    setWicketCount(wicketCount + 1)
    disableAllScoreButtons()
  }
}
if (isRunOut) {
  if (batter1.id === playerId) {
    newBatter1()
    changeStrikeRadio('strike')
    switchBatterStrike('batter1')
  } else {
    newBatter2()
    changeStrikeRadio('non-strike')
    switchBatterStrike('batter2')
  }
} else {
  if (!isNoBall) {
    if (batter1.onStrike) {
      newBatter1()
    } else {
      newBatter2()
    }
  }
}

```

```

    }
  }
  if (isNoBall) {
    if (isRunOut && wicketCount + 1 === 10) {
      const endInningButton = document.getElementById('end-inning')
      endInningButton.disabled = false
      const bowlerNameElement = document.querySelector('.react-autosuggest__input')
      bowlerNameElement.disabled = true
      const batter1NameElement = document.getElementById('batter1Name')
      batter1NameElement.disabled = true
      const batter2NameElement = document.getElementById('batter2Name')
      batter2NameElement.disabled = true
      setInputBowler('')
    }
  } else {
    if (wicketCount + 1 === 10) {
      const endInningButton = document.getElementById('end-inning')
      endInningButton.disabled = false
      const bowlerNameElement = document.querySelector('.react-autosuggest__input')
      bowlerNameElement.disabled = true
      const batter1NameElement = document.getElementById('batter1Name')
      batter1NameElement.disabled = true
      const batter2NameElement = document.getElementById('batter2Name')
      batter2NameElement.disabled = true
      setInputBowler('')
    }
  }
}
const handleCloseModal = () => {
  if (outType !== '') {
    if (outType === RUN_OUT) {
      if (runOutPlayerId !== '') {
        handleWicket(true, runOutPlayerId)
      }
    } else {
      handleWicket(false, '')
    }
  }
}
setModalOpen(false)
setOutType('')
setRunOutPlayerId('')
}
const handleOutTypeChange = (e) => {
  const outTypeValue = e.target.value
  setOutType(outTypeValue)
  if (outTypeValue === RUN_OUT) {
    const runOutPlayerElement = document.getElementById('run-out-player')
    runOutPlayerElement.classList.remove('hide')
    const runOutPlayerErrorElement = document.getElementById('run-out-player-error')
    runOutPlayerErrorElement.classList.remove('hide')
  }
}

```

```

    }
  }
  const handleRunOutPlayerChange = (e) => {
    const playerId = e.target.value
    const runOutPlayerErrorElement = document.getElementById('run-out-player-error')
    runOutPlayerErrorElement.classList.add('hide')
    setRunOutPlayerId(playerId)
  }
  const endMatch = () => {
    disableAllScoreButtons()
    const endInningButton = document.getElementById('end-inning')
    if (endInningButton.textContent === 'Score Board') {
      endInningButton.disabled = false
    }
  }
  const disableAllScoreButtons = () => {
    const scoreTypesButtons = document.querySelectorAll('.score-types-button')
    for (let i = 0; i < scoreTypesButtons.length; i++) {
      scoreTypesButtons[i].disabled = true
    }
  }
  const enableAllScoreButtons = () => {
    const scoreTypesButtons = document.querySelectorAll('.score-types-button')
    for (let i = 0; i < scoreTypesButtons.length; i++) {
      scoreTypesButtons[i].disabled = false
    }
  }
  if (batter1.name !== undefined && batter2.name !== undefined && inputBowler !== '') {
    enableAllScoreButtons()
  }
  let rrr = (remainingRuns / (remainingBalls / 6)).toFixed(2)
  rrr = isFinite(rrr) ? rrr : 0
  const overs = overCount + ballCount / 6
  let crr = (totalRuns / overs).toFixed(2)
  crr = isFinite(crr) ? crr : 0
  const inning1 = match.inning1
  const inning2 = match.inning2
  const scoringTeam = batting === team1 ? team1 : team2
  const chassingTeam = scoringTeam === team1 ? team2 : team1
  let winningMessage = `${inningNo === 1 ? scoringTeam : chassingTeam} needs ${remainingRuns} ${
    remainingRuns <= 1 ? 'run' : 'runs'
  } in ${remainingBalls} ${remainingBalls <= 1 ? 'ball' : 'balls'} to win`
  if (inningNo === 2) {
    var target = inning1.runs + 1
    if (wicketCount < 10 && overCount <= maxOver && totalRuns >= target) {
      winningMessage = `${chassingTeam} won by ${10 - wicketCount} wickets`
      endMatch()
    }
  }
  if ((wicketCount >= 10 || overCount >= maxOver) && totalRuns < target - 1) {
    winningMessage = `${scoringTeam} won by ${target - totalRuns - 1} runs`
  }

```

```

        endMatch()
    }
    if (wicketCount < 10 && overCount === maxOver && totalRuns === target - 1) {
        winningMessage = 'Match Tied'
        endMatch()
    }
}
const welcomeContent = (
<>
<div></div>
<div>Welcome to MyLive Cricket Score</div>
<div></div>
</>
)
const firstInningCompletedContent = (
<>
    {overCount === maxOver &&<div>1st inning completed</div>}
    {wicketCount === 10 &&<div>All Out</div>}
<div>Please click "End Inning" button</div>
</>
)
const remainingRunsContent = (
<>
<div>Target: {target}</div>
<div>{winningMessage}</div>
<div>RRR: {isNaN(rrr) ? 0 : rrr}</div>
</>
)
return (
<div className='container'>
<div className='inning'>
<div>
            {team1} vs {team2}, {inningNo === 1 ? '1st' : '2nd'} Inning
        </div>
<div>
<button id='end-inning' onClick={handleEndInning}>
            {inningNo === 1 ? 'End Inning' : 'Score Board'}
        </button>
</div>
</div>
<div id='badge' className='badge badge-flex'>
            {inningNo === 2 ? remainingRunsContent : overCount === maxOver || wicketCount === 10 ?
firstInningCompletedContent : welcomeContent}
        </div>
<div className='score-container'>
<div>
<Modal
            open={isModalOpen}
            onClose={handleCloseModal}
            aria-labelledby='modal-modal-title'

```

```

        aria-describedby='modal-modal-description'
    >
    <Box sx={radioGroupBoxstyle}>
    <FormControl component='fieldset'>
    <RadioGroup
        row
        aria-label='wicket'
        name='row-radio-buttons-group'
        onChange={handleOutTypeChange}
        sx={{ alignItems: 'center' }}
    >
    <FormControlLabel
        value={CATCH}
        control={
    <Radio
        sx={{
            '&.Mui-checked': {
                color: pink[600],
            },
        }}
    />
        label={CATCH}
    />
    <FormControlLabel
        value={STUMP}
        control={
    <Radio
        sx={{
            '&.Mui-checked': {
                color: pink[600],
            },
        }}
    />
        label={STUMP}
    />
    <FormControlLabel
        value={HIT_WICKET}
        control={
    <Radio
        sx={{
            '&.Mui-checked': {
                color: pink[600],
            },
        }}
    />
        label={HIT_WICKET}
    />

```

```

<FormControlLabel
  value={BOLD}
  control={
    <Radio
      sx={{
        '&.Mui-checked': {
          color: pink[600],
        },
      }}
    />
  }
  label={BOLD}
/>
<FormControlLabel
  value={RUN_OUT}
  control={
    <Radio
      sx={{
        '&.Mui-checked': {
          color: pink[600],
        },
      }}
    />
  }
  label={RUN_OUT}
/>
<select defaultValue="" id='run-out-player' className='run-out-player hide'
onChange={handleRunOutPlayerChange}>
  <option value="" disabled>
    select option
  </option>
  <option value={batter1.id}>{batter1.name}</option>
  <option value={batter2.id}>{batter2.name}</option>
</select>
</RadioGroup>
<div id='run-out-player-error' className='run-out-player-error hide'>
  Please select run out player name
</div>
</FormControl>
</Box>
</Modal>
</div>
<div className='score'>
  <div>
    {inningNo === 1 ? scoringTeam : chessingTeam} : {totalRuns}/{wicketCount} {totalOvers}
  </div>
  <div>CRR : {isNaN(crr) ? 0 : crr}</div>
</div>
<div className='batting-container'>
  <table>

```



```

<thead>
<tr>
<td className='score-types padding-left'>Batting</td>
<td className='score-types'>R(B)</td>
<td className='score-types text-center'>4s</td>
<td className='score-types text-center'>6s</td>
<td className='score-types text-center'>SR</td>
</tr>
</thead>
<tbody>
<tr>
<td className='score-types'>
<span id='strike'>
<Radio
      size='small'
      checked={strikeValue === 'strike'}
      onChange={handleStrikeChange}
      value='strike'
      name='radio-buttons'
      inputProps={{ 'aria-label': 'strike' }}
      style={{ padding: '0 4px 0 2px' }}
    />
</span>
<input type='text' id='batter1Name' className='batter-name' onBlur={handleBatter1Blur} />
<IconButton color='primary' className='icon-button' onClick={editBatter1Name}>
<EditIcon className='icon-size' />
</IconButton>
</td>
<td className='score-types'>{batter1.run === undefined ? `0(0)` : `${batter1.run}(${batter1.ball})`}</td>
<td className='score-types'>{batter1.four === undefined ? 0 : batter1.four}</td>
<td className='score-types'>{batter1.six === undefined ? 0 : batter1.six}</td>
<td className='score-types'>{batter1.strikeRate === undefined ? 0 : batter1.strikeRate}</td>
</tr>
<tr>
<td className='score-types'>
<span id='non-strike'>
<Radio
      size='small'
      checked={strikeValue === 'non-strike'}
      onChange={handleStrikeChange}
      value='non-strike'
      name='radio-buttons'
      inputProps={{ 'aria-label': 'non-strike' }}
      style={{ padding: '0 4px 0 2px' }}
    />
</span>
<input type='text' id='batter2Name' className='batter-name' onBlur={handleBatter2Blur} />
<IconButton color='primary' className='icon-button' onClick={editBatter2Name}>
<EditIcon className='icon-size' />
</IconButton>

```

```

</td>
<td className='score-types'>{batter2.run === undefined ? `0(0)` : `${batter2.run}(${batter2.ball})`}</td>
<td className='score-types'>{batter2.four === undefined ? 0 : batter2.four}</td>
<td className='score-types'>{batter2.six === undefined ? 0 : batter2.six}</td>
<td className='score-types'>{batter2.strikeRate === undefined ? 0 : batter2.strikeRate}</td>
</tr>
</tbody>
</table>
</div>
<div className='bowler-container'>
<div className='bowler'>
    Bowler:
    <Autosuggest
        suggestions={suggestions}
        onSuggestionsFetchRequested={onSuggestionsFetchRequested}
        onSuggestionsClearRequested={() => {
            setSuggestions([])
        }}
        getSuggestionValue={getSuggestionValue}
        renderSuggestion={(suggestion) => <div>{suggestion.name}</div>}
        inputProps={inputProps}
    />
    <IconButton color='primary' className='icon-button' onClick={editBowlerName}>
    <EditIcon className='icon-size' />
    </IconButton>
</div>
<div className='bowler-runs'>
    {currentRunStack.map((run, i) => (
    <div key={i}>{run}</div>
    )))}
    <IconButton color='warning' className='icon-button' onClick={undoDelivery}>
    <DeleteIcon className='delete-icon-size' />
    </IconButton>
</div>
</div>
<div className='score-types-container'>
<table>
<tbody>
<tr>
<td className='score-types' onClick={() => handleRun(0)}>
<button className='score-types-button' disabled>
    0
</button>
</td>
<td className='score-types' onClick={() => handleRun(1)}>
<button className='score-types-button' disabled>
    1
</button>
</td>
<td className='score-types' onClick={() => handleRun(2)}>

```

```

<button className='score-types-button' disabled>
    2
</button>
</td>
<td className='score-types' onClick={handleNoBall}>
<button className='score-types-button' disabled>
    nb
</button>
</td>
<td rowspan='2' className='score-types' onClick={() => setModalOpen(true)}>
<button className='score-types-button' disabled>
    W
</button>
</td>
</tr>
<tr>
<td className='score-types' onClick={() => handleRun(3)}>
<button className='score-types-button' disabled>
    3
</button>
</td>
<td className='score-types' onClick={() => handleRun(4)}>
<button className='score-types-button' disabled>
    4
</button>
</td>
<td className='score-types' onClick={() => handleRun(6)}>
<button className='score-types-button' disabled>
    6
</button>
</td>
<td className='score-types' onClick={handleWide}>
<button className='score-types-button' disabled>
    wd
</button>
</td>
</tr>
</tbody>
</table>
</div>
<div className='extras-container'>
<div>Extras: {extras.total}</div>
<div>Wd: {extras.wide}</div>
<div>NB: {extras.noBall}</div>
</div>
<div className='recent-over-container'>
<div className='recent-over-text'>Recent Overs</div>
<div className='recent-over-details'>
<table>
<tbody>

```

```

        {recentOvers.map((recentOver, i) => (
<tr key={i}>
<td>{recentOver.overNo}</td>
<td>{recentOver.bowler}</td>
<td>
<div className='recent-over-runs'>
        {recentOver.stack.map((run, index) => (
<div key={index}>{run}</div>
        ))}
</div>
</td>
<td className='recent-over-total-run'>
<div>{recentOver.runs}</div>
</td>
</tr>
        ))}
</tbody>
</table>
</div>
</div>
<div className='score-board-container'>
<div className='score-board-text text-center'>Score Board</div>
    { /* Inning1 Starts here */ }
<div>
<div className='score-board-innings'>
<div>{scoringTeam} Innings</div>
<div>RR:{inningNo === 1 ? crr : inning1.runRate}</div>
<div>
        {inningNo === 1 ? totalRuns : inning1.runs}-{inningNo === 1 ? wicketCount : inning1.wickets} (
        {inningNo === 1 ? totalOvers : inning1.overs} Ov)
</div>
</div>
<div className='sb-batting'>
<table>
<thead>
<tr>
<td className='score-types padding-left'>Batter</td>
<td className='score-types'>R(B)</td>
<td className='score-types text-center'>4s</td>
<td className='score-types text-center'>6s</td>
<td className='score-types text-center'>SR</td>
</tr>
</thead>
<tbody>
        {inning1.batters.map((batter, i) => {
            return (
<tr key={i}>
<td className='score-types padding-left'>{batter.name}</td>
<td className='score-types'>
                {batter.run}{batter.ball}

```

```

</td>
<td className='score-types text-center'>{batter.four}</td>
<td className='score-types text-center'>{batter.six}</td>
<td className='score-types text-center'>{batter.strikeRate}</td>
</tr>
    )
  }}}
<tr>
<td className='score-types padding-left'>Extras:</td>
<td className='score-types'>{inningNo === 1 ? extras.total : inning1.extra.total}</td>
<td className='score-types text-center'>wd:{inningNo === 1 ? extras.wide : inning1.extra.wide}</td>
<td className='score-types text-center'>nb:{inningNo === 1 ? extras.noBall : inning1.extra.noBall}</td>
<td className='score-types text-center'></td>
</tr>
</tbody>
</table>
</div>
<div className='sb-bowling'>
<table>
<thead>
<tr>
<td className='score-types padding-left'>Bowler</td>
<td className='score-types'>O</td>
<td className='score-types text-center'>M</td>
<td className='score-types text-center'>R</td>
<td className='score-types text-center'>W</td>
<td className='score-types text-center'>NB</td>
<td className='score-types text-center'>WD</td>
<td className='score-types text-center'>ECO</td>
</tr>
</thead>
<tbody>
    {inning1.bowlers.map((blr, i) => {
      const { name, over, maiden, run, wicket, noBall, wide, economy } = blr
      return (
<tr key={i}>
<td className='score-types padding-left'>{name}</td>
<td className='score-types'>{over}</td>
<td className='score-types text-center'>{maiden}</td>
<td className='score-types text-center'>{run}</td>
<td className='score-types text-center'>{wicket}</td>
<td className='score-types text-center'>{noBall}</td>
<td className='score-types text-center'>{wide}</td>
<td className='score-types text-center'>{economy}</td>
</tr>
      )
    })}
</tbody>
</table>
</div>

```

```

</div>
    { /* Inning2 Starts here */ }
    { inningNo === 2 && (
<div>
<div className='score-board-innings'>
<div>{chessingTeam} Innings</div>
<div>RR:{inningNo === 2 ? crr : inning2.runRate}</div>
<div>
        {hasMatchEnded ? inning2.runs : totalRuns}-{hasMatchEnded ? inning2.wickets : wicketCount}
    (
        {hasMatchEnded ? inning2.overs : totalOvers} Ov)
</div>
</div>
<div className='sb-batting'>
<table>
<thead>
<tr>
<td className='score-types padding-left'>Batter</td>
<td className='score-types'>R(B)</td>
<td className='score-types text-center'>4s</td>
<td className='score-types text-center'>6s</td>
<td className='score-types text-center'>SR</td>
</tr>
</thead>
<tbody>
        {inning2.batters.map((batter, i) => {
            return (
<tr key={i}>
<td className='score-types padding-left'>{batter.name}</td>
<td className='score-types'>
                {batter.run}{batter.ball}
            </td>
<td className='score-types text-center'>{batter.four}</td>
<td className='score-types text-center'>{batter.six}</td>
<td className='score-types text-center'>{batter.strikeRate}</td>
</tr>
                )
            )}}
<tr>
<td className='score-types padding-left'>Extras:</td>
<td className='score-types'>{hasMatchEnded ? inning2.extra.total : extras.total}</td>
<td className='score-types text-center'>wd:{hasMatchEnded ? inning2.extra.wide : extras.wide}</td>
<td className='score-types text-center'>nb:{hasMatchEnded ? inning2.extra.noBall :
extras.noBall}</td>
<td className='score-types text-center'></td>
</tr>
</tbody>
</table>
</div>
<div className='sb-bowling'>

```

```

<table>
<thead>
<tr>
<td className='score-types padding-left'>Bowler</td>
<td className='score-types'>O</td>
<td className='score-types text-center'>M</td>
<td className='score-types text-center'>R</td>
<td className='score-types text-center'>W</td>
<td className='score-types text-center'>NB</td>
<td className='score-types text-center'>WD</td>
<td className='score-types text-center'>ECO</td>
</tr>
</thead>
<tbody>
    {inning2.bowlers.map((blr, i) => {
      const { name, over, maiden, run, wicket, noBall, wide, economy } = blr
      return (
        <tr key={i}>
          <td className='score-types padding-left'>{name}</td>
          <td className='score-types'>{over}</td>
          <td className='score-types text-center'>{maiden}</td>
          <td className='score-types text-center'>{run}</td>
          <td className='score-types text-center'>{wicket}</td>
          <td className='score-types text-center'>{noBall}</td>
          <td className='score-types text-center'>{wide}</td>
          <td className='score-types text-center'>{economy}</td>
        </tr>
      )
    })}
</tbody>
</table>
</div>
</div>
    )}
</div>
</div>
</div>
  )
}

export default ScoreBoard

```

### 3. StepperContainer.js

This shows the first page after the application starts working. It is show the GUI of the first page of the Application

```
import { AppBar, Box, Container, Toolbar, Typography } from '@material-ui/core'
import React from 'react'
import HorizontalStepper from './HorizontalStepper'

const StepperContainer = () => {
  return (
    <div>
      <AppBar position='fixed'>
        <Toolbar>
          <Typography variant='h6'>Match Startup</Typography>
        </Toolbar>
      </AppBar>
      <Container>
        <Box marginTop={10}>
          <HorizontalStepper />
        </Box>
      </Container>
    </div>
  )
}

export default StepperContainer
```

## 4. ScoreBoard.css

This module consists all the CSS used to maintain scoreboard. CSS-in-JS libraries provide us with a new approach for writing CSS. They aim to tackle the limitations of CSS, such as the lack of dynamic functionality, scoping, and portability.

In spite of its advantages, CSS-in-JS is a controversial technology, as many developers ask if it's worth further complicating front-end development.

```
.inning {
  background: #3f51b5;
  color: white;
  padding: 0.5rem;
  font-size: 1.2rem;
  display: flex;
  justify-content: space-between;
  align-items: center;
}
```



```

#end-inning {
  height: 100%;
  padding: 4px;
  outline: none;
  border: 1px solid #fff;
  border-radius: 4px;
}
#end-inning:active {
  color: #fff;
  background-color: #3f51b5;
}
.score-container {
  margin: 0.5rem;
}
.run-out-player {
  height: 2rem;
  max-width: 9.3rem;
}
.run-out-player-error {
  color: #ff0000;
  font-size: 0.7rem;
  text-align: right;
  padding-right: 0rem;
}
.hide {
  display: none;
}
.badge {
  background: #ffff4d;
  color: #000000;
  text-align: center;
  font-size: 0.75rem;
  font-weight: 500;
  padding: 8px;
  margin-bottom: 8px;
}
.badge-flex {
  display: flex;
  align-items: center;
  justify-content: space-between;
}
.score {
  display: flex;
  align-items: center;
  justify-content: space-between;
  font-size: 1.05rem;
  font-weight: bold;
  padding: 4px;
}
.batting-container {
  display: flex;

```

```

    align-items: center;
    justify-content: flex-start;
    margin: 0.8rem 0;
  }
  .batting-container table {
    width: 100%;
    border: none;
    border-spacing: unset;
  }
  table thead {
    padding: 8px;
    background-color: #3f51b517;
  }
  .batting-container table td {
    padding: 8px 4px;
  }
  .bowler {
    display: flex;
    align-items: center;
    background-color: #3f51b517;
    padding: 8px;
  }
  .react-autosuggest__container {
    padding-left: 4px;
    position: relative;
  }
  .react-autosuggest__input {
    max-width: 6.5rem;
  }
  .react-autosuggest__input--focused {
    /* outline: none; */
  }
  .react-autosuggest__input--open {
    border-bottom-left-radius: 0;
    border-bottom-right-radius: 0;
  }
  .react-autosuggest__suggestions-container {
    display: none;
  }
  .react-autosuggest__suggestions-container--open {
    display: block;
    position: absolute;
    top: 1.3rem;
    width: 6.85rem;
    border: 1px solid #808080;
    background-color: #fff;
    font-weight: 400;
    font-size: 0.8rem;
    z-index: 2;
  }
  .react-autosuggest__suggestions-list {
    margin: 0;
    padding: 0;
  }

```

```

    list-style-type: none;
  }
  .react-autosuggest__suggestion {
    cursor: pointer;
    padding: 4px;
  }
  .react-autosuggest__suggestion--highlighted {
    background-color: #ddd;
  }
  .bowler-runs {
    display: flex;
    align-items: center;
    justify-content: space-between;
    padding: 8px;
  }
  .score-types-container table {
    width: 100%;
    height: 100%;
    border-spacing: 8px;
  }
  .score-types-container table .score-types {
    background: #3f51b517;
  }
  .score-types-container table td {
    border: 1px solid #3f51b5;
  }
  .score-types-button {
    display: block;
    width: 100%;
    height: 100%;
    padding: 8px;
    outline: none;
    border: none;
  }
  .score-types-button:active {
    color: white;
    background-color: #3f51b5;
  }
  .score-types-button-noball {
    color: #000;
    background-color: #ffff4d;
  }
  .icon-button {
    padding: 0 4px !important;
  }
  .icon-size {
    font-size: 1rem !important;
  }
  .delete-icon-size {
    font-size: 1.3rem !important;
  }
  .strike-icon-button {
    padding: 0 !important;
  }

```

```

}
.batter-name {
  max-width: 6.5rem;
}
.extras-container {
  margin: 0.8rem 0;
  display: flex;
  justify-content: space-between;
  padding: 8px;
  background: #3f51b517;
}
.recent-over-container {
  margin: 0.8rem 0;
}
.recent-over-text {
  background-color: #3f51b517;
  padding: 8px;
}
.recent-over-details table {
  width: 100%;
  font-size: 0.9rem;
}
.recent-over-runs {
  display: flex;
  justify-content: space-around;
  gap: 4px;
  padding: 8px;
}
.recent-over-total-run {
  text-align: center;
  border-left: 1px solid #ccc;
  font-weight: bold;
  padding: 0 2px;
}
.text-center {
  text-align: center;
}
.score-board-container {
  margin: 0.8rem 0;
}
.score-board-innings {
  display: flex;
  justify-content: space-between;
  padding: 6px 4px;
  color: #fff;
  background: #737373;
  font-size: 0.95rem;
}
.score-board-text {
  color: #fff;
  background: #404040;
  padding: 8px;
}

```

```

.sb-batting {
  display: flex;
  align-items: center;
  justify-content: flex-start;
}
.sb-batting table {
  width: 100%;
  border: none;
  border-spacing: unset;
}
.sb-batting table thead {
  font-size: 0.9rem;
}
.sb-batting table tbody {
  font-size: 0.85rem;
}
.sb-batting table td {
  padding: 8px 4px;
}
.score-board-extras {
  margin: 0.8rem 0;
  display: flex;
  justify-content: space-between;
  padding: 6px 4px;
  font-size: 0.85rem;
  background: #3f51b517;
}

```

```

.sb-bowling {
  display: flex;
  align-items: center;
  justify-content: flex-start;
}
.sb-bowling table {
  width: 100%;
  border: none;
  border-spacing: unset;
}
.sb-bowling table thead {
  font-size: 0.9rem;
}
.sb-bowling table tbody {
  font-size: 0.85rem;
}
.sb-bowling table td {
  padding: 8px 4px;
}

```

## **Chapter – 6**

### **SOFTWARE TESTING**

#### **6.1. Software Testing**

Software testing is a critical element of software quality assurance and represents the ultimate reuse of specification. Design and code testing represents interesting anomaly for the software during earlier definition and development phase, it was attempted to build software from an abstract concept to tangible implementation.

The testing phase involves, testing of the development of the system using various techniques such as White Box Testing, Control Structure Testing.

#### **6.2. Testing Strategies:**

A strategy for software testing must accommodate low-level tests that are necessary to verify that a small source code segment has been correctly implemented as well as high level against customer requirements.

- 1) Unit Testing
- 2) Integration testing
- 3) System testing.
- 4) Acceptance testing.

#### **6.3. Testing Techniques:**

##### **6.3.1. White Box Testing:**

White box testing is a test case design method that uses the control structure of the procedural design to derive test cases.

##### **6.3.1.1. Control Structure Testing**

The following tests were conducted and it was noted that the BCBS is performing them well.

- Basic path Testing
- Condition Testing
- Data Flow Testing
- Loop Testing

#### **Implementation:**

There are many type of testing including in white box and black box testing.

**Usability Testing:**

- Usability testing is nothing but the User-friendliness check.
- In Usability testing, the application flow is tested so that a new user can understand the application easily.
- Basically, system navigation is checked in Usability testing.

**Usability Test Cases:**

- Web page content should be correct without any spelling or grammatical errors.
- All fonts should be same as per the requirements.
- All the text should be properly aligned.
- All the error messages should be correct without any spelling or grammatical errors and the error message should match with the field label.
- Tool tip text should be there for every field.
- All the fields should be properly aligned.
- Enough space should be provided between field labels, columns, rows, and error messages.
- All the buttons should be in a standard format and size.
- Home link should be there on every single page.
- Disabled fields should be grayed out.
- Check for broken links and images.
- Confirmation message should be displayed for any kind of update and delete operation.
- Check the site on different resolutions (640 x 480, 600×800 etc.?)
- Check the end user can run the system without frustration.
- Check the tab should work properly.
- Scroll bar should appear only if required.

- If there is an error message on submit, the information filled by the user should be there.
- Title should display on each web page
- All fields (Textbox, dropdown, radio button, etc) and buttons should be accessible by keyboard shortcuts and the user should be able to perform all operations by using keyboard.
- Check if the dropdown data is not truncated due to the field size. Also, check whether the data is hardcoded or managed via administrator

### **Compatibility Testing-**

Compatibility testing is used to determine if your software is compatible with other elements of a system with which it should operate, e.g. Browsers, Operating Systems, or hardware.

#### **Compatibility Test Scenarios:**

- Test the website in different browsers (IE, Firefox, Chrome, Safari and Opera) and ensure the website is displaying properly.
- Test the HTML version being used is compatible with appropriate browser versions.
- Test the images display correctly in different browsers.
- Test the fonts are usable in different browsers.
- Test the java script code is usable in different browsers.
- Test the Animated GIF's across different browsers.

### **Database Testing--**

In Database testing backend records are tested which have been inserted through the web or desktop applications.

The data which is displaying in the web application should match with the data stored in the Database.

#### **Test Cases for Database Testing:**

- Verify the database name: The database name should match with the specifications.



- Verify the Tables, columns, column types and defaults: All things should match with the specifications.
- Verify whether the column allows a null or not.
- Verify the Primary and foreign key of each table.
- Verify the Stored Procedure:
- Test whether the Stored procedure is installed or not.
- Verify the Stored procedure name
- Verify the parameter names, types and number of parameters.
- Test the parameters if they are required or not.
- Test the stored procedure by deleting some parameters
- Test when the output is zero, the zero records should be affected.
- Test the stored procedure by writing simple SQL queries.
- Test whether the stored procedure returns the values
- Test the stored procedure with sample input data.
- Verify the behavior of each flag in the table.
- Verify the data gets properly saved into the database after each page submission.
- Verify the data if the DML (Update, delete and insert) operations are performed.
- Check the length of every field: The field length in the back end and front end must be same.
- Verify the database names of QA, UAT and production. The names should be unique.
- Verify the encrypted data in the database.
- Verify the database size. Also test the response time of each query executed.
- Verify the data displayed on the front end and make sure it is same in the back end.
- Verify the data validity by inserting the invalid data in the database.

- Verify the Triggers.

### **What is Security Testing?**

Security Testing involves the test to identify any flaws and gaps from a security point of view.

#### **Test Scenarios for Security Testing:**

- Verify the web page which contains important data like password, credit card numbers, secret answers for security question etc should be submitted via HTTPS (SSL).
- Verify the important information like password, credit card numbers etc should display in encrypted format.
- Verify password rules are implemented on all authentication pages like Registration, forgot password, change password.
- Verify if the password is changed the user should not be able to login with the old password.
- Verify the error messages should not display any important information.
- Verify if the user is logged out from the system or user session was expired, the user should not be able to navigate the site.
- Verify to access the secured and non-secured web pages directly without login.
- Verify the “View Source code” option is disabled and should not be visible to the user.
- Verify the user account gets locked out if the user is entering the wrong password several times.
- Verify the cookies should not store passwords.
- Verify if, any functionality is not working, the system should not display any application, server, or database information. Instead, it should display the custom error page.
- Verify the SQL injection attacks.
- Verify the user roles and their rights. For Example, the requestor should not be able to access the admin page.

- Verify the important operations are written in log files, and that information should be traceable.
- Verify the session values are in an encrypted format in the address bar.
- Verify the cookie information is stored in encrypted format.
- Verify the application for Brute Force Attacks

## **Chapter – 7**

### **Maintenance**

#### **7.1. Introduction**

Software maintenance refers to the process of modifying and updating a software system after it has been delivered to the customer. This can include fixing bugs, adding new features, improving performance, or updating the software to work with new hardware or software systems.

The goal of software maintenance is to keep the software system working correctly, efficiently, and securely, and to ensure that it continues to meet the needs of the users.

Maintenance is required to maintain a building's initial performance capacity. Without maintenance, performance will not meet the demand and eventually will drop below the limit of acceptance of residents. In practice, both the demand and the limit of acceptance will gradually rise over time as a result of improved technology, rising standards, and growing prosperity.

Improvement and renewal are required to answer the accordingly rising expectations. As a result, the total life cycle costs will generally be a multiple of the initial building costs. Maintenance is a combination of all technical and associated administrative actions during the service life to retain a building or its parts in a state in which it can perform its required functions.

#### **7.2. Administrator**

Work for the cookies on cricket maintenance due to manage restaurant management system for the order to provide you are easily found free of the future.

Weaknesses and if the project cricket score maintenance system as per your consent prior to batch mode execution. Efforts of this project on cricket score system is totally built using vb projects published on rails management system project is high as the team captain.

Thesis to that the project report on cricket score maintenance due to view menu is not available. Task schedule and the working on maintenance system for booking, customer online role playing area of cricket score management system is a wide variety of stations.

Possible features that, project report on cricket score system online, and may resume the restaurant management system is not have source code of the customer make the details. Materials in the cookies on cricket score maintenance due to this project in proper resource management system will view the modern periodic table. Especially zip so,

project on cricket score system is important to store and reservation table project with monitoring the purpose of these cookies are categorized as the source software.

Printing and the working on cricket score maintenance system can occur without the screen and the time. At all is a report on cricket score maintenance system allows customer was developed by using the administration. Website and monitor the project on cricket system project of charges customer will need a web application.

### **7.3. Types of website maintenance services**

#### **1) Updating Website Software:**

If you are using any kind of content management system like Word Press or any kind of script like PHP, it needs to always be up to date. Word Press usually updates itself automatically but there are instances where it might need to be updated manually.

Additionally, if you are using a website host, a web service maintenance provider will ensure that they are updating their core server software on a regular basis. For instance, they will check if things like FTP service running on the server are up to date. If however, you run your own web server, they will manually test and apply updates.

#### **2) Improving Website Speed:**

Did you know that twenty five percent of users navigate out of a website if it takes more than three seconds to load? Page loading is obviously an important part of the user experience. However, we tend to let it slide in order to accommodate new nifty functionality or esthetic website design.

In reality, web visitors do not care about all the bells and whistles. They care about accessing the information they want as fast as possible. Another important factor to consider is how page load speed affects your search engine rankings. A slow website increases bounce rate and if people are navigating out of your web page as fast as they got there, then Google will think your website isn't important so they will push it further down the rankings.

#### **3) Fixing HTML Errors**

HTML Code isn't easy especially if you know nothing about website coding. You need to pay attention to detail to ensure that it's working well. What might seem like a minor error could lead to a number of problems such as funky looking pages, missing multimedia, etc. Reliable website support should be able to inspect, find and fix issues related to HTML.

#### **4) Backing up Files:**

Imagine how frustrating it would be if you were to lose all your website's essential files.

This includes images, pages, blog posts, plugging, etc. If you run your website on Word Press, you might have a few automated backup options. Website maintenance services ensure that file and database backups is automatically performed at least on a weekly basis.

#### **5) Developing New Content:**

Your content strategy might be on point but it's always a good idea to improve it by creating new types of content. Your web content also needs to be up to date. If you wrote a post that was published years ago, it might not be relevant right now and some of the information might need to be updated.

#### **6) Search Engine Optimization:**

Regularly checking your website for areas where search ability can be improved can go a long way to improving search engine optimization. Your blog posts can be altered to include popular and competitive keywords. Your landing pages can also be updated to include more relevant and timely keywords. Even your "About Us" page can be updated regularly to take advantage of popular keywords in your industry. Keyword research is also important when creating new content for your website.

As part of the SEO strategy, it's also important to identify structural issues within your website that may affect how search engines view your site.

#### **7) Ensuring Design Consistency across All Pages:**

Your website should function properly in all the latest versions of major website browsers like Chrome, Safari, Firefox, etc. These browsers update frequently and if your website doesn't adapt to the updates it might not show up or it might not look the way you want it to. This could negatively affect your business and your branding. It's also important to ensure that you have a responsive website design that looks good on mobile devices.

Another factor to consider about your website's design is uniformity. If you have a lot of branding elements that represent your company, they should be included uniformly across all your pages.

#### **8) Fixing Broken Links:**

Every link on your website should lead exactly to the right place but sometimes links can go bad. A site or resource you linked to earlier might disappear, a page on

your website might become unavailable or you might move a post and forget to update others that link to it.

Broken links can be detrimental to your website. They could give visitors the impression that:

- a) Your website is not user friendly
- b) You don't regularly update your website
- c) You are not credible

## **9) Reviewing Your Website Analytics:**

Your website's performance needs to be monitored to ensure that everything is working as it should. An experienced website support specialist will examine high-level metrics like how many new and returning visitors your website has received for the past week or which blog posts are the most read. They will then make adjustments your website accordingly to ensure all the metrics are at their peak.

This is just a minimum starting point of what website maintenance is all about. Some of these tasks can be overly time consuming. Additionally, if you do not know what you are doing, you could end up doing more damage. The better option is to hire professional website maintenance services.

## **Chapter – 8**

### **Literature Review**

#### **Abstract**

Proper design has become a critical element needed to engage website and mobile application users. However, little research has been conducted to define the specific elements used in effective website and mobile application design. We attempt to review and consolidate research on effective design and to define a short list of elements frequently used in research. The design elements mentioned most frequently in the reviewed literature were navigation, graphical representation, organization, content utility, purpose, simplicity, and readability. We discuss how previous studies define and evaluate these seven elements. This review and the resulting short list of design elements may be used to help designers and researchers to operationalize best practices for facilitating and predicting user engagement.

**Keywords:** Website design, usability, navigation, organization, simplicity

#### **8.1.Introduction**

Internet usage has increased tremendously and rapidly in the past decade (“Internet Use Over Time,” 2014). Websites have become the most important public communication portal for most, if not all, businesses and organizations. As of 2014, 87% of American adults aged 18 or older are Internet users (“Internet User Demographics,” 2013). Because business-to-consumer interactions mainly occur online, website design is critical in engaging users (Flavián, Guinalíu, & Gurrea, 2006; Lee & Kozar, 2012; Petre, Minocha, & Roberts, 2006). Poorly designed websites may frustrate users and result in a high “bounce rate”, or people visiting the entrance page without exploring other pages within the site (Google.com, 2015). On the other hand, a well-designed website with high usability has been found to positively influence visitor retention (revisit rates) and purchasing behavior (Avouris, Tselios, Fidas, & Papachristos, 2003; Flavián et al., 2006; Lee & Kozar, 2012).

Little research, however, has been conducted to define the specific elements that constitute effective website design. One of the key design measures is usability (International Standardization Organization, 1998). The International Standardized Organization (ISO) defines usability as the extent to which users can achieve desired tasks (e.g., access desired information or place a purchase) with effectiveness (completeness and accuracy of the task), efficiency (time spent on the task), and satisfaction (user experience) within a system. However, there is currently no consensus on how to properly operationalize and assess website usability (Lee & Kozar, 2012). For example, Nielson associates usability with learn ability, efficiency, memo ability, errors, and satisfaction (Nielsen, 2012). Yet, Palmer (2002) postulates



that usability is determined by download time, navigation, content, interactivity, and responsiveness. Similar to usability, many other key design elements, such as scan ability, readability, and visual aesthetics, have not yet been clearly defined (Bevan, 1997; Brady & Phillips, 2003; Kim, Lee, Han, & Lee, 2002), and there are no clear guidelines that individuals can follow when designing websites to increase engagement.

This review sought to address that question by identifying and consolidating the key website design elements that influence user engagement according to prior research studies. This review aimed to determine the website design elements that are most commonly shown or suggested to increase user engagement. Based on these findings, we listed and defined a short list of website design elements that best facilitate and predict user engagement. The work is thus an exploratory research providing definitions for these elements of website design and a starting point for future research to reference.

## **8.2. Materials and Methods**

### **8.2.1. Selection Criteria and Data Extraction**

We searched for articles relating to website design on Google Scholar (scholar.google.com) because Google Scholar consolidates papers across research databases (e.g., Pubmed) and research on design is listed in multiple databases. We used the following combination of keywords: design, usability, and websites. Google Scholar yielded 115,000 total hits. However, due to the large list of studies generated, we decided to only review the top 100 listed research studies for this exploratory study.

Our inclusion criterion for the studies was: (1) publication in a peer-reviewed academic journal, (2) publication in English, and (3) publication in or after 2000. Year of publication was chosen as a limiting factor so that we would have enough years of research to identify relevant studies but also have results that relate to similar styles of websites after the year 2000. We included studies that were experimental or theoretical (review papers and commentaries) in nature. Resulting studies represented a diverse range of disciplines, including human-computer interaction, marketing, e-commerce, and interface design, cognitive science, and library science. Based on these selection criteria, thirty-five unique studies remained and were included in this review.

## **8.3. Final Search Term**

### **8.3.1. (Design) AND (usability) AND (websites)**

The search terms were kept simple to capture the higher level design/usability papers and allow Google scholar's ranking method to filter out the most popular studies. This method also allowed studies from a large range of fields to be searched.

### 8.3.2. Analysis

The literatures review uncovered 20 distinct design elements commonly discussed in research that affect user engagement. They were (1)organization—is the website logically organized,

(2) content utility – is the information provided useful or interesting, (3) navigation – is the website easy to navigate, (4) graphical representation – does the website utilize icons, contrasting colors, and multimedia content, (5) purpose – does the website clearly state its purpose (i.e. personal, commercial, or educational), (6) memorable elements – does the website facilitate returning users to navigate the site effectively (e.g., through layout or graphics), (7) valid links – does the website provide valid links, (8) simplicity – is the design of the website simple, (9) impartiality – is the information provided fair and objective,

(10) credibility – is the information provided credible, (11) consistency/reliability – is the website consistently designed (i.e., no changes in page layout throughout the site), (12) accuracy – is the information accurate, (13) loading speed – does the website take a long time to load, (14) security/privacy – does the website securely transmit, store, and display personal information/data, (15) interactive – can the user interact with the website (e.g., post comments or receive recommendations for similar purchases), (16) strong user control capabilities– does the website allow individuals to customize their experiences (such as the order of information they access and speed at which they browse the website), (17) readability – is the website easy to read and understand (e.g., no grammatical/ spelling errors),(18) Efficiency – is the information presented in a way that users can find the information they need quickly,(19) scan ability –can users pick out relevant information quickly, and (20) learn ability – how steep is the learning curve for using the website. For each of the above, we calculated the proportion of studies mentioning the element.

In this review, we provide a threshold value of 30%. We identified elements that were used in at least 30% of the studies and include these elements that are above the threshold on a short list of elements used in research on proper website design. The 30% value was an arbitrary threshold picked that would provide researchers and designers with a guideline list of elements described in research on effective web design. To provide further information on how to apply this list, we present specific details on how each of these elements was discussed in research so that it can be defined and operationalized.

## 8.4.Results

### 8.4.1. Popular Website Design Elements (Table 1)

Seven of the website design elements met our threshold requirement for review. Navigation was the most frequently discussed element, mentioned in 22 articles (62.86%). Twenty-one studies (60%) highlighted the importance of graphics. Fifteen

studies (42.86%) emphasized good organization. Four other elements also exceeded the threshold level, and they were Content utility (n=13, 37.14%), purpose (n=11, 31.43%), simplicity (n=11, 31.43%), and readability (n=11, 31.43%).

Elements below our minimum requirement for review include memorable features (n=5, 14.29%), links (n=10, 28.57%), impartiality (n=1, 2.86%), credibility (n=7, 20%), consistency/reliability (n=8, 22.86%), accuracy (n=5, 14.29%), loading speed (n=10, 28.57%), security/privacy (n=2, 5.71%), interactive features (n=9, 25.71%), strong user control capabilities (n=8, 22.86%), efficiency (n=6, 17.14%), scan ability (n=1, 2.86%), and learn ability (n=2, 5.71%).

**Table 1 – Frequency of website design elements used in research (2000-2014)**

Authors	Year	Elements*					
		Organization	Content Utility	Navigation	Graphical Representation	Purpose	Simplicity
Rosen & Purinton	2004	1	1	1	1	1	1
Tan & Wei	2007	1		1	1	1	1
Cyr, Head, & Larios	2010				1		
Arroyo, Selker, & Wei	2006	1		1			1
Tarafdar & Zhang	2008	1	1	1			
Flavián et al.	2006			1			
George	2005	1		1	1		1
Zhang & Von Dran	2000		1		1	1	
Thompson, Braddy, & Wuensch	2008	1		1	1	1	
Williamson, Lepak, & King	2003			1		1	
Maurer & Liu	2007		1		1	1	
Braddy, Meade, & Kroustalis	2008				1		
Atterer, Wnuk, & Schmidt	2006			1			1
Belanche, Casaló, & Guinalíu	2012	1		1	1	1	1
Djonov	2007	1		1		1	
Lee & Kozar	2012			1			1
Dastidar	2009	1	1	1		1	
Banati, Bedi, & Grover	2001	1	1	1	1	1	1
Djamasbi, Siegel, & Tullis	2010			1	1		
Raward	2001			1	1		1
De Angeli, Sutcliffe, & Hartmann	2006		1	1	1		

Blackmon, Kitajima, & Polson							
Song & Zahedi							
Lowry et al							
Avouris et al							
Auger							
Green & Pearson							
Zhang, Small, Von Dran, & Barcellos							
Shneiderman & Hochheiser							
Petrie, Hamilton, & King							
Petre et al							
Lim							
<b>Total</b>							
<b>%</b>							
<b>Ranking</b>							
Sutcliffe							
Cyr, Ilsever, Bonanni, & Bowes	2004	1		1	1	1	
Blackmon, Polson, Kitajima, & Lewis	2002	1			1	1	

\* Elements in table all met the 30%+ threshold; elements not meeting the 30% threshold are not shown.

#### **8.4.2. Defining Key Design Elements for User Engagement (Table 2)**

In defining and operationalizing each of these elements, the research studies suggested that effective navigation is the presence of salient and consistent menu/navigation bars, aids for navigation (e.g., visible links), search features, and easy access to pages (multiple pathways and limited clicks/backtracking). Engaging graphical presentation entails 1) inclusion of images, 2) proper size and resolution of images, 3) multimedia content, 4) proper color, font, and size of text, 5) use of logos and icons, 6) attractive visual layout, 7) color schemes, and 8) effective use of white space. Optimal organization includes 1) cognitive architecture, 2) logical, understandable, and hierarchical structure, 3) information arrangement and categorization, 4) meaningful labels/headings/titles, and 5) use of keywords. Content utility is determined by 1) sufficient amount of information to attract repeat visitors, 2) arousal/motivation (keeps visitors interested and motivates users to continue exploring the site), 3) content quality, 4) information relevant to the purpose of the site, and 5) perceived utility based on user needs/requirements. The purpose of a website is clear when it 1) establishes a unique and visible brand/identity, 2) addresses visitors' intended purpose and expectations for visiting the site, and 3) provides information about the organization and/or services. Simplicity is achieved by using 1) simple subject headings, 2) transparency of information (reduce search time), 3) website design optimized for computer screens, 4) uncluttered layout, 5) consistency in design throughout website, 6) ease of using (including first-time users), 7) minimize redundant features, and 8) easily understandable functions. Readability is optimized by content that is 1) easy to read, 2) well-written, 3) grammatically correct, 4) understandable, 5) presented in readable blocks, and 6) reading level appropriate.

**Table 2 – Definitions of Key Design Elements**

Key Elements	Definition
Content Utility	<ul style="list-style-type: none"> <li>• Salient menu/navigation bar</li> <li>• Consistency of navigation bar</li> <li>• Aids for navigation (e.g..visible links)</li> <li>• Easy access to web pages(e.g. no excessive backtracking/client and reach through multiple pathways)</li> <li>• Search features</li> <li>• Users feel in control/ease of managing</li> </ul>
Purpose	
Simplicity	
Readability	
Navigation	
Graphical Representation	<ul style="list-style-type: none"> <li>• Inclusion of images</li> <li>• Size and resolution of images</li> <li>• Multimedia content (e.g. animation or audio)</li> <li>• Color, font, and size of text</li> <li>• Distinct logos and icons</li> <li>• Visual attractiveness/layout</li> <li>• Color schemes</li> <li>• Effective use of white space/ avoid visual overload</li> <li>• Minimizing loading time for visual elements</li> </ul>
Organization	<ul style="list-style-type: none"> <li>• Cognitive mapping architecture</li> <li>• Understandable structure</li> <li>• Logical organization</li> </ul>

## 8.5.Discussion

The seven website design elements most often discussed in relation to user engagement in the reviewed studies were navigation (62.86%), graphical representation (60%), organization (42.86%), content utility (37.14%), purpose (31.43%), simplicity (31.43%), and readability (31.43%). These seven elements exceeded our threshold level of 30% representation in the literature and were included into a short list of website design elements to operationalize effective website design. For further analysis, we reviewed how studies defined and evaluated these seven elements. This may allow designers and researchers to determine and follow best practices for facilitating or predicting user engagement.

A remaining challenge is that the definitions of website design elements often overlap. For example, several studies evaluated organization by how well a website incorporates cognitive architecture, logical and hierarchical structure, systematic information arrangement and categorization, meaningful headings and labels, and keywords.

However, these features are also crucial in navigation design. Also, the implications of using distinct logos and icons go beyond graphical representation. Logos and icons also establish unique brand/identity for the organization (purpose) and can serve as visual aids for navigation. Future studies are needed to develop distinct and objective measures to assess these elements and how they affect user engagement (Lee & Kozar, 2012).

Given the rapid increase in both mobile technology and social media use, it is surprising that no studies mentioned cross-platform compatibility and social media integration. In 2013, 34% of cell phone owners primarily use their cell phones to access the Internet, and this number continues to grow ("Mobile Technology Factsheet," 2013). With the rise of different mobile devices, users are also diversifying their web browser use. Internet Explorer (IE) was once the leading web browser.

However, in recent years, Fire Fox, Safari, and Chrome have gained significant traction (W3schools.com, 2015). Website designers and researchers must be mindful of different platforms and browsers to minimize the risk of losing users due to compatibility issues. In addition, roughly 74% of American Internet users use some form of social media (Duggan, Ellison, Lampe, Lenhart, & Smith, 2015), and social media has emerged as an effective platform for organizations to target and interact with users. Integrating social media into website design may increase user engagement by facilitating participation and interactivity.

There are several limitations to the current review. First, due to the large number of studies published in this area and due to this study being exploratory, we selected from the first 100 research publications on Google Scholar search results. Future studies may benefit from defining design to a specific topic, set of years, or

other area to limit the number of search results. Second, we did not quantitatively evaluate the effectiveness of these website design elements. Additional research can help to better quantify these elements.

It should also be noted that different disciplines and industries have different objectives in designing websites and should thus prioritize different website design elements.

For example, online businesses and marketers seek to design websites that optimize brand loyalty, purchase, and profit (Petre et al., 2006). Others, such as academic researchers or healthcare providers, are more likely to prioritize privacy/confidentiality, and content accuracy in building websites (Horvath, Ecklund, Hunt, Nelson, & Toomey, 2015). Ultimately, we advise website designers and researchers to consider the design elements delineated in this review, along with their unique needs, when developing user engagement strategies.



## 8.6.References

Arroyo, Ernesto, Selker, Ted & Wei, Willy (2006) Usability tool for analysis of web designs using mouse tracks. Paper presented at the CHI'06 Extended Abstracts on Human Factors in Computing Systems.

Atterer, Richard, Wnuk, Monika & Schmidt, Albrecht (2006) Knowing the user's every move: user activity tracking for website usability evaluation and implicit interaction. Paper presented at the Proceedings of the 15th international conference on World Wide Web.

Auger, Pat (2005) the impact of interactivity and design sophistication on the performance of commercial websites for small businesses. *Journal of Small Business Management*, 43(2), 119-137.

Avouris, Nikolaos, Tselios, Nikolaos, Fidas, Christos & Papachristos, Eleftherios. (2003). Website evaluation: A usability-based perspective *Advances in Informatics* (pp. 217- 231): Springer.

Banati, Hema, Bedi, Punam, & Grover, PS. (2006). Evaluating web usability from the user's perspective. *Journal of Computer Science*, 2(4), 314.

Belanche, Daniel, Casaló, Luis V, & Guinalú, Miguel. (2012). Website usability, consumer satisfaction and the intention to use a website: The moderating effect of perceived risk. *Journal of retailing and consumer services*, 19(1), 124-132.

Bevan, Nigel. (1997). Usability issues in web site design. Paper presented at the HCI (2).

Blackmon, Marilyn Hughes, Kitajima, Muneo, & Polson, Peter G. (2003). Repairing usability problems identified by the cognitive walkthrough for the web. Paper presented at the Proceedings of the SIGCHI conference on Human factors in computing systems.

Blackmon, Marilyn Hughes, Polson, Peter G, Kitajima, Muneo, & Lewis, Clayton. (2002).

Cognitive walkthrough for the web. Paper presented at the Proceedings of the SIGCHI conference on human factors in computing systems.

Braddy, Phillip W, Meade, Adam W, & Kroustalis, Christina M. (2008). Online recruiting: The effects of organizational familiarity, website usability, and website attractiveness on viewers' impressions of organizations. *Computers in Human Behavior*, 24(6), 2992-3001.

Brady, Laurie, & Phillips, Christine. (2003). Aesthetics and usability: A look at color and balance. *Usability News*, 5(1), 2003.

Cyr, Dianne, Head, Milena, & Larios, Hector. (2010). Colour appeal in website design within and across cultures: A multi-method evaluation. *International journal of human-computer studies*, 68(1), 1-21.

Cyr, Dianne, Ilsever, Joe, Bonanni, Carole, & Bowes, John. (2004). Website Design and Culture: An Empirical Investigation. Paper presented at the IWIPS.

Dastidar, Surajit Ghosh. (2009). Impact of the factors influencing website usability on user satisfaction.

De Angeli, Antonella, Sutcliffe, Alistair, & Hartmann, Jan. (2006). Interaction, usability and aesthetics: what influences users' preferences? Paper presented at the Proceedings of the 6th conference on Designing Interactive systems.

Djamasbi, Soussan, Siegel, Marisa, & Tullis, Tom. (2010). Generation Y, web design, and eye tracking. *International journal of human-computer studies*, 68(5), 307-323.

Djonov, Emilia. (2007). Website hierarchy and the interaction between content organization, webpage and navigation design: A systemic functional hypermedia discourse analysis perspective. *Information Design Journal*, 15(2), 144-162.

Duggan, M., Ellison, N., Lampe, C., Lenhart, A., & Smith, A. (2015). Social Media update 2014. Washington, D.C.: Pew Research Center.

Flavián, Carlos, Guinalíu, Miguel, & Gurrea, Raquel. (2006). The role played by perceived usability, satisfaction and consumer trust on website loyalty. *Information & Management*, 43(1), 1-14.

George, Carole A. (2005). Usability testing and design of a library website: an iterative approach. *OCLC Systems & Services: International digital library perspectives*, 21(3), 167-180.

Google.com. (2015). Bounce Rate. Analytics Help. Retrieved 2/11, 2015, from <https://support.google.com/analytics/answer/1009409?hl=en>

Green, D., & Pearson, J.M. (2006). Development of a web site usability instrument based on ISO 9241-11. *Journal of Computer Information Systems*, Fall.

Horvath, Keith J, Ecklund, Alexandra M, Hunt, Shanda L, Nelson, Toben F, & Toomey, Traci L. (2015). Developing Internet-Based Health Interventions: A Guide for Public Health Researchers and Practitioners. *J Med Internet Res*, 17(1), e28. doi: 10.2196/jmir.3770 International Standardization Organization. (1998). ISO 2941-11:1998 Ergonomic requirements for office work with visual display terminals (VDTs) -- Part 11: Guidance on usability: International Standardization Organization (ISO). Internet Use Over Time. (2014, January 2). Retrieved February 15, 2015, from <http://www.pewinternet.org/data-trend/internet-use/internet-use-over-time/>

Internet User Demographics. (2013, November 14). Retrieved February 11, 2015, from <http://www.pewinternet.org/data-trend/internet-use/latest-stats/>

- Kim, Jinwoo, Lee, Jungwon, Han, Kwanghee, & Lee, Moonkyu. (2002). Businesses as Buildings: Metrics for the Architectural Quality of Internet Businesses. *Information Systems Research*, 13(3), 239-254. doi: doi:10.1287/isre.13.3.239.79
- Lee, Younghwa, & Kozar, Kenneth A. (2012). Understanding of website usability: Specifying and measuring constructs and their relationships. *Decision Support Systems*, 52(2), 450-463.
- Lim, Sun. (2002). The Self-Confrontation Interview: Towards an Enhanced Understanding of Human Factors in Web-based Interaction for Improved Website Usability. *J. Electron. Commerce Res.*, 3(3), 162-173.
- Lowry, Paul Benjamin, Spaulding, Trent, Wells, Taylor, Moody, Greg, Moffit, Kevin, & Madariaga, Sebastian. (2006). A theoretical model and empirical results linking website interactivity and usability satisfaction. Paper presented at the System Sciences, 2006. HICSS'06. Proceedings of the 39th Annual Hawaii International Conference on.
- Maurer, Steven D, & Liu, Yuping. (2007). Developing effective e-recruiting websites: Insights for managers from marketers. *Business Horizons*, 50(4), 305-314.
- Mobile Technology Fact Sheet. (2013, December 27). Retrieved August 5, 2015, from <http://www.pewinternet.org/fact-sheets/mobile-technology-fact-sheet/>
- Nielsen, Jakob. (2012). Usability 101: introduction to Usability. Retrieved 2/11, 2015, from <http://www.nngroup.com/articles/usability-101-introduction-to-usability/>
- Palmer, Jonathan W. (2002). Web Site Usability, Design, and Performance Metrics. *Information Systems Research*, 13(2), 151-167. doi: doi:10.1287/isre.13.2.151.88
- Petre, Marian, Minocha, Shailey, & Roberts, Dave. (2006). Usability beyond the website: an empirically-grounded e-commerce evaluation instrument for the total customer experience. *Behaviour & Information Technology*, 25(2), 189-203.
- Petrie, Helen, Hamilton, Fraser, & King, Neil. (2004). Tension, what tension?: Website accessibility and visual design. Paper presented at the Proceedings of the 2004 international cross-disciplinary workshop on Web accessibility (W4A)
- Raward, Roslyn. (2001). Academic library website design principles: development of a checklist. *Australian Academic & Research Libraries*, 32(2), 123-136.
- Rosen, Deborah E, & Purinton, Elizabeth. (2004). Website design: Viewing the web as a cognitive landscape. *Journal of Business Research*, 57(7), 787-794.
- Shneiderman, Ben, & Hochheiser, Harry. (2001). Universal usability as a stimulus to advanced interface design. *Behaviour & Information Technology*, 20(5), 367-376.
- Song, Jaeki, & Zahedi, Fatemeh "Mariam". (2005). A theoretical approach to web design in e-commerce: a belief reinforcement model. *Management Science*, 51(8), 1219-1235.
- Sutcliffe, Alistair. (2001). Heuristic evaluation of website attractiveness and usability *Interactive systems: design, specification, and verification* (pp. 183-198): Springer. Tan,

- Gek Woo, & Wei, Kwok Kee. (2007). An empirical study of Web browsing behaviour: Towards an effective Website design. *Electronic Commerce Research and Applications*, 5(4), 261-271.
- Tarafdar, Monideepa, & Zhang, Jie. (2008). Determinants of reach and loyalty-a study of Website performance and implications for Website design. *Journal of Computer Information Systems*, 48(2), 16.
- Thompson, Lori Foster, Braddy, Phillip W, & Wuensch, Karl L. (2008). E-recruitment and the benefits of organizational web appeal. *Computers in Human Behavior*, 24(5), 2384-2398. [W3schools.com](http://www.w3schools.com)).
- Browser Statistics and Trends. Retrieved 1/15, 2015, from [http://www.w3schools.com/browsers/browsers\\_stats.asp](http://www.w3schools.com/browsers/browsers_stats.asp) Williamson, Ian O, Lepak, David P, & King, James. (2003).
- The effect of company recruitment web site orientation on individuals' perceptions of organizational attractiveness. *Journal of Vocational Behavior*, 63(2), 242-263.
- Zhang, Ping, Small, Ruth V, Von Dran, Gisela M, & Barcellos, Silvia. (2000). A two factor theory for website design. Paper presented at the System Sciences, 2000.
- Proceedings of the 33rd Annual Hawaii International Conference on.
- Zhang, Ping, & Von Dran, Gisela M. (2000). Satisfiers and dissatisfiers: A two-factor model for website design and evaluation. *Journal of the American society for information science*, 51(14), 1253-1268.

## BIBLIOGRAPHY

**Material UI:** - <https://mui.com/material-ui/getting-started/overview/>

**React:** - <https://reactjs.org/docs/add-react-to-a-website.html>

<https://madewithreact.com/tag/dashboards/>

<https://reactjsexample.com/tag/website/>

**Node JS:** - <https://nodejs.org/en/>