

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **CHAPTER NO.** | **CHAPTER TITLE** | **PAGE NO.** |
| 1. | PROBLEM STATEMENT   * 1. Problem Description   2. Aim and Objective of the Project   1.3 Abstract |  |
| 2. | BACKGROUND STUDY    2.1 Existing Work   * 1. Tools Used   2.3 Proposed System |  |
| 3. | SYSTEM DESIGN    3.1 Data Flow Diagram  3..2 ER Diagram |  |
| 4. | MODULES    4.1 Module Description |  |
| 5. | TESTING  5.1 Unit Testing |  |
|  | CONCLUSION |  |
|  | APPENDIX  SCREENSHOTS |  |

CHAPTER 1

INTRODUCTION

* 1. PROBLEM DESCRIPTION

Develop a web based application that provides live cricket scores ,Cricket enthusiasts often face challenges in staying updated with live match scores, and match highlights, especially when they are on the go or unable to watch the game live. Traditional methods of tracking scores, such as TV or radio broadcasts, are not always accessible, and existing solutions may lack real-time updates, in-depth analytics, or a user-friendly interface.

The goal is to develop a web application that provides real-time cricket match updates and notifications.

Creating a live cricket score web application involves integrating angular CLI for designing a user interface and showing the past ,present and future playing list.

Create an HTML file to design the webpage structure, including angular js and node js. Use an Angular CLI and css to generate this project.

1.2 AIM AND OBJECTIVE OF THE PROJECT

The "Live Sport Cricket Score Traffic Application" is a dynamic web-based application designed to provide real-time cricket scores and updates to users, using JavaScript for seamless performance and interaction. The application uses live data feeds to deliver accurate and up-to-date information on ongoing cricket matches, including scores, upcoming matches and past match scores. JavaScript, along with its frameworks and libraries, plays a crucial role in handling asynchronous data requests, updating the user interface in real time, and ensuring a smooth user experience. The application uses APIs to fetch live scores. By utilizing JavaScript's event-driven architecture and efficient DOM manipulation, the application aims to provide cricket enthusiasts with a comprehensive and engaging platform to follow their favourite sport live.

1.3ABSTRACT

The "Live Sports App" is an interactive web application designed to deliver real-time updates and results for cricket and football matches. Utilizing a diverse set of technologies, including AngularJS, Node.js, Angular, Zone.js, and React.js, this project aims to create a dynamic and user-friendly platform for sports enthusiasts.

The app features three main modules: **Completed Matches**, which showcases the results of finished games; **Current Matches**, providing live scores and ongoing match updates; and **Upcoming Matches**, listing future fixtures for both sports. By integrating these modules, users can easily navigate and access the information they need.

The application is built to be responsive, ensuring an optimal experience across various devices. With a focus on performance and real-time data, the Live Sports App aims to keep users engaged and informed. Future enhancements will include personalized notifications and a broader range of sports coverage.

Through continuous development and user feedback, the app aspires to become a comprehensive hub for sports information, fostering a deeper connection between fans and their favourite games. This project not only addresses the need for timely updates but also enhances the overall sports viewing experience.

**CHAPTER 2**

**SYSTEM STUDY**

2.1 EXISTING SYSTEM:

Currently, many sports tracking platforms provide fragmented access to match information, often requiring users to navigate multiple sites for results, live scores, and schedules. Most existing applications lack real-time updates, leading to delays in information delivery. Additionally, many systems are not optimized for mobile devices, resulting in a subpar user experience. Users frequently encounter slow loading times and limited functionalities. The "Live Sports App" addresses these issues by consolidating completed, current, and upcoming match data into a single, responsive platform. By leveraging modern frameworks like AngularJS, Node.js, and React.js, this app aims to enhance user engagement and streamline access to sports information.

**2.1.1 Disadvantages:**

Complexity of Integration

Performance Overhead

Steeper Learning Curve

**2.1.2 Advantages:**

Rich User Experience

Modular Architecture

Real-Time Data Handling

2.3 PROPOSED SYSTEM:

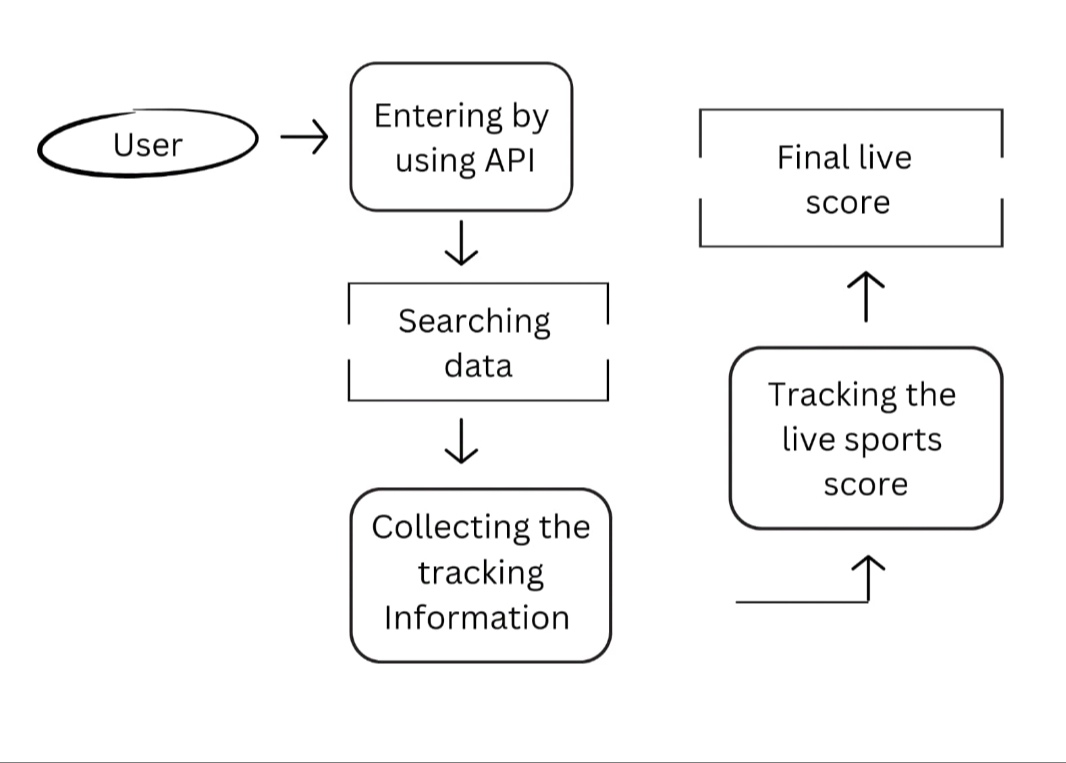
The "Live Sports App" aims to revolutionize how sports enthusiasts engage with cricket and football match information by providing a comprehensive and user-friendly platform. This system will feature a unified interface that consolidates three essential modules: completed, current, and upcoming matches. By employing Node.js for backend processing, the app will deliver real-time updates, ensuring that users receive the latest scores and match developments instantly. The front end will utilize a combination of Angular, React, and Zone.js to create a responsive design that works seamlessly across various devices, from desktops to smartphones.

Interactive features will enhance user engagement, including personalized notifications for match start times and score changes, allowing fans to stay connected to their favourite teams. The modular architecture will facilitate easy maintenance and scalability, enabling future enhancements without major overhauls. By focusing on intuitive navigation and quick access to crucial information, the app will prioritize user experience, making it easy for visitors to find what they need. Ultimately, the proposed system seeks to provide a reliable and engaging sports tracking solution that meets the needs of today’s digital sports fans.

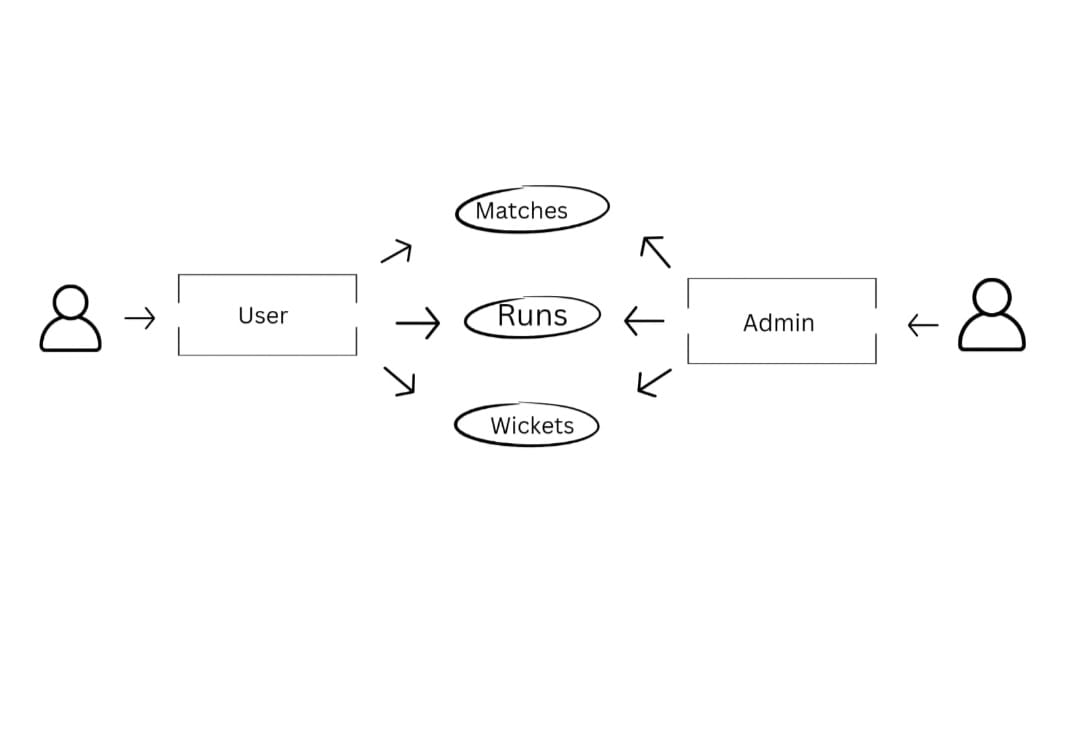
CHAPTER 3

SYSTEM DESIGN

3.1 DATA FLOW DIAGRAM



3.2 SYSTEM FLOW DIAGRAM



**CHAPTER 4**

**MODULE**

**4.1 MODULE DESCRIPTION:**

4.1.1 MODULES NAME:

The "Live Sports App" consists of three essential modules that provide users with comprehensive information about cricket and football matches. The **Completed Matches Module** offers detailed results of finished games, showcasing scores, highlights, and key statistics to help fans analyse past performances. The **Current Matches Module** delivers real-time updates, allowing users to follow ongoing matches with live scores, commentary, and critical moments, ensuring they stay engaged with the action as it happens. Meanwhile, the **Upcoming Matches Module** lists future fixtures, including dates, times, and venues, enabling fans to plan ahead and set reminders for their favourite teams' games. Each module is designed with a user-friendly interface for easy navigation and quick access to important information. Leveraging Angular, Node.js, and React.js ensures that the app remains responsive and fast across devices, providing a seamless experience for users. Together, these modules create an all-encompassing platform that meets the needs of sports enthusiasts, enhancing their connection to the games they love.

**Completed Matches Module**

The **Completed Matches Module** serves as a thorough archive of finished games, catering to fans who want to delve into the results and highlights of past matches. Users can access detailed results that include not only final scores but also a breakdown of individual performances, such as runs scored, wickets taken in cricket, and goals and assists in football. Additionally, this module features highlight reels that capture key moments from the games, allowing fans to relive the excitement and drama of pivotal plays. For those interested in deeper analysis, the module provides advanced statistics and insights, such as player performance metrics, team comparisons, and historical data, empowering fans to understand the context and implications of each match.

**Current Matches Module**

In contrast, the **Current Matches Module** is tailored for users who wish to stay up-to-date with live action. This module offers real-time updates, ensuring fans receive instantaneous information on ongoing matches. Live scores are prominently displayed, allowing users to follow the action as it unfolds. Text commentary provides valuable context, enhancing the viewing experience by explaining significant events and strategies in real time. Users are also notified of critical moments—such as goals, wickets, and injuries—keeping them engaged and informed throughout the match. The design of this module supports multitasking, enabling fans to track multiple games simultaneously, which is ideal for those who follow various teams.

**Upcoming Matches Module**

The **Upcoming Matches Module** focuses on helping fans plan their viewing schedules. It features a comprehensive list of future matches, complete with essential details such as dates, times, and venues. This module also allows users to set custom reminders, ensuring they never miss a game. Fans can subscribe to notifications for specific teams, receiving updates about match times and any changes, which helps them stay organized and engaged with their favorite sports.

To ensure a seamless and responsive experience, the Live Sports App utilizes a robust technology stack that includes Angular, Node.js, and React.js. Angular facilitates dynamic rendering of components, enhancing performance and enabling real-time updates. Node.js efficiently handles backend processes, ensuring quick data fetching and live updates during matches. Meanwhile, React.js provides a smooth and interactive user interface, making navigation intuitive and accessible. The combination of these technologies guarantees that the app performs optimally across various devices, catering to the diverse needs of sports fans.

By integrating these three modules, the Live Sports App creates a comprehensive platform that not only enhances the enjoyment of watching sports but also deepens fans' connection to the games and teams they are passionate about.

**4.1.2 USER INTERFACE**

The user interface (UI) of the "Live Sports App" is meticulously crafted to provide an engaging and user-friendly experience for sports enthusiasts. The main dashboard features a clean, modern layout that highlights the three key modules: **Completed Matches**, **Current Matches**, and **Upcoming Matches**. Each module is visually distinct, utilizing vibrant colours and intuitive icons to facilitate quick navigation.

In the **Completed Matches Module**, users can easily scroll through a list of finished games, complete with scorecards and highlights. Interactive elements allow users to click on specific matches for detailed statistics and player performances, presented in an organized and accessible format. The **Current Matches Module** displays live updates with real-time scores, featuring animated graphics that emphasize critical moments, ensuring users feel the excitement of ongoing games.

The **Upcoming Matches Module** provides a calendar view of future fixtures, complete with countdown timers for each match. Users can filter upcoming events by date or team, and options to set reminders ensure they never miss a game. The UI is fully responsive, ensuring seamless functionality on desktops, tablets, and smartphones, with touch-friendly elements enhancing usability on mobile devices. Overall, the design prioritizes clarity and interaction, making it easy for users to connect with the sports they love.

**4.1.3 SAMPLE CODE**

{

  "name": "live-sports-app",

  "version": "0.0.0",

  "lockfileVersion": 3,

  "requires": true,

  "packages": {

    "": {

      "name": "live-sports-app",

      "version": "0.0.0",

      "dependencies": {

        "@angular/animations": "^18.0.0",

        "@angular/cdk": "^18.0.1",

        "@angular/common": "^18.0.0",

        "@angular/compiler": "^18.0.0",

        "@angular/core": "^18.0.0",

        "@angular/forms": "^18.0.0",

        "@angular/material": "^18.0.1",

        "@angular/platform-browser": "^18.0.0",

        "@angular/platform-browser-dynamic": "^18.0.0",

        "@angular/router": "^18.0.0",

        "rxjs": "~7.8.0",

        "tslib": "^2.3.0",

        "zone.js": "~0.14.3"

      },

      "devDependencies": {

        "@angular-devkit/build-angular": "^18.0.2",

        "@angular/cli": "^18.0.2",

        "@angular/compiler-cli": "^18.0.0",

        "@types/jasmine": "~5.1.0",

        "jasmine-core": "~5.1.0",

        "karma": "~6.4.0",

        "karma-chrome-launcher": "~3.2.0",

        "karma-coverage": "~2.2.0",

        "karma-jasmine": "~5.1.0",

        "karma-jasmine-html-reporter": "~2.1.0",

        "typescript": "~5.4.2"

      }

    },

    "node\_modules/@ampproject/remapping": {

      "version": "2.3.0",

      "resolved": "https://registry.npmjs.org/@ampproject/remapping/-/remapping-2.3.0.tgz",

      "integrity": "sha512-30iZtAPgz+LTIYoeivqYo853f02jBYSd5uGnGpkFV0M3xOt9aN73erkgYAmZU43x4VfqcnLxW9Kpg3R5LC4YYw==",

      "dev": true,

      "dependencies": {

        "@jridgewell/gen-mapping": "^0.3.5",

        "@jridgewell/trace-mapping": "^0.3.24"

      },

      "engines": {

        "node": ">=6.0.0"

      }

    },

    "node\_modules/@angular-devkit/architect": {

      "version": "0.1800.2",

      "resolved": "https://registry.npmjs.org/@angular-devkit/architect/-/architect-0.1800.2.tgz",

      "integrity": "sha512-PX7lCTAqWe9C40+fie+DAc8vhpGA+JgZKWWrMHUTV/iZx8RXx2X4xGQsqYu36p4i3MSfQdbn+0xLWGmjScPVOQ==",

      "dev": true,

      "dependencies": {

        "@angular-devkit/core": "18.0.2",

        "rxjs": "7.8.1"

      },

      "engines": {

        "node": "^18.19.1 || ^20.11.1 || >=22.0.0",

        "npm": "^6.11.0 || ^7.5.6 || >=8.0.0",

        "yarn": ">= 1.13.0"

      }

    },

    "node\_modules/@angular-devkit/build-angular": {

      "version": "18.0.2",

      "resolved": "https://registry.npmjs.org/@angular-devkit/build-angular/-/build-angular-18.0.2.tgz",

      "integrity": "sha512-cQkTx7XaIPj6+DXo6wZmO4iY0hOOfPDnSN/+m84XpBW0tuPGxH7Z9B6wV+Uwcpm9HGPqzRA7VZyPsqbK860b0Q==",

      "dev": true,

      "dependencies": {

        "@ampproject/remapping": "2.3.0",

        "@angular-devkit/architect": "0.1800.2",

        "@angular-devkit/build-webpack": "0.1800.2",

        "@angular-devkit/core": "18.0.2",

        "@angular/build": "18.0.2",

        "@babel/core": "7.24.5",

        "@babel/generator": "7.24.5",

        "@babel/helper-annotate-as-pure": "7.22.5",

        "@babel/helper-split-export-declaration": "7.24.5",

        "@babel/plugin-transform-async-generator-functions": "7.24.3",

        "@babel/plugin-transform-async-to-generator": "7.24.1",

        "@babel/plugin-transform-runtime": "7.24.3",

        "@babel/preset-env": "7.24.5",

        "@babel/runtime": "7.24.5",

        "@discoveryjs/json-ext": "0.5.7",

        "@ngtools/webpack": "18.0.2",

        "@vitejs/plugin-basic-ssl": "1.1.0",

        "ansi-colors": "4.1.3",

        "autoprefixer": "10.4.19",

        "babel-loader": "9.1.3",

        "babel-plugin-istanbul": "6.1.1",

        "browserslist": "^4.21.5",

        "copy-webpack-plugin": "11.0.0",

        "critters": "0.0.22",

        "css-loader": "7.1.1",

        "esbuild-wasm": "0.21.3",

        "fast-glob": "3.3.2",

        "http-proxy-middleware": "3.0.0",

        "https-proxy-agent": "7.0.4",

        "inquirer": "9.2.22",

        "jsonc-parser": "3.2.1",

        "karma-source-map-support": "1.4.0",

        "less": "4.2.0",

        "less-loader": "12.2.0",

        "license-webpack-plugin": "4.0.2",

        "loader-utils": "3.2.1",

        "magic-string": "0.30.10",

        "mini-css-extract-plugin": "2.9.0",

        "mrmime": "2.0.0",

        "open": "8.4.2",

        "ora": "5.4.1",

        "parse5-html-rewriting-stream": "7.0.0",

        "picomatch": "4.0.2",

        "piscina": "4.5.0",

        "postcss": "8.4.38",

        "postcss-loader": "8.1.1",

        "resolve-url-loader": "5.0.0",

        "rxjs": "7.8.1",

        "sass": "1.77.2",

        "sass-loader": "14.2.1",

        "semver": "7.6.2",

        "source-map-loader": "5.0.0",

        "source-map-support": "0.5.21",

        "terser": "5.31.0",

        "tree-kill": "1.2.2",

        "tslib": "2.6.2",

        "undici": "6.18.0",

        "vite": "5.2.11",

        "watchpack": "2.4.1",

        "webpack": "5.91.0",

        "webpack-dev-middleware": "7.2.1",

        "webpack-dev-server": "5.0.4",

        "webpack-merge": "5.10.0",

        "webpack-subresource-integrity": "5.1.0"

      },

      "engines": {

        "node": "^18.19.1 || ^20.11.1 || >=22.0.0",

        "npm": "^6.11.0 || ^7.5.6 || >=8.0.0",

        "yarn": ">= 1.13.0"

      },

      "optionalDependencies": {

        "esbuild": "0.21.3"

      },

      "peerDependencies": {

        "@angular/compiler-cli": "^18.0.0",

        "@angular/localize": "^18.0.0",

        "@angular/platform-server": "^18.0.0",

        "@angular/service-worker": "^18.0.0",

        "@web/test-runner": "^0.18.0",

        "browser-sync": "^3.0.2",

        "jest": "^29.5.0",

        "jest-environment-jsdom": "^29.5.0",

        "karma": "^6.3.0",

        "ng-packagr": "^18.0.0",

        "protractor": "^7.0.0",

        "tailwindcss": "^2.0.0 || ^3.0.0",

        "typescript": ">=5.4 <5.5"

      },

      "peerDependenciesMeta": {

        "@angular/localize": {

          "optional": true

        },

        "@angular/platform-server": {

          "optional": true

        },

        "@angular/service-worker": {

          "optional": true

        },

        "@web/test-runner": {

          "optional": true

        },

        "browser-sync": {

          "optional": true

        },

        "jest": {

          "optional": true

        },

        "jest-environment-jsdom": {

          "optional": true

        },

        "karma": {

          "optional": true

        },

        "ng-packagr": {

          "optional": true

        },

        "protractor": {

          "optional": true

        },

        "tailwindcss": {

          "optional": true

        }

      }

    },

    "node\_modules/@angular-devkit/build-webpack": {

      "version": "0.1800.2",

      "resolved": "https://registry.npmjs.org/@angular-devkit/build-webpack/-/build-webpack-0.1800.2.tgz",

      "integrity": "sha512-CbTURBhZWzx+5KewS2Nkqy2rwBTFgDCvUwONGWuy1K68+85vOWUKqjkfvriHA+JkWN03w7FzWEtTfcOg0EzYkw==",

      "dev": true,

      "dependencies": {

        "@angular-devkit/architect": "0.1800.2",

        "rxjs": "7.8.1"

      },

      "engines": {

        "node": "^18.19.1 || ^20.11.1 || >=22.0.0",

        "npm": "^6.11.0 || ^7.5.6 || >=8.0.0",

        "yarn": ">= 1.13.0"

      },

      "peerDependencies": {

        "webpack": "^5.30.0",

        "webpack-dev-server": "^5.0.2"

      }

    },

    "node\_modules/@angular-devkit/core": {

      "version": "18.0.2",

      "resolved": "https://registry.npmjs.org/@angular-devkit/core/-/core-18.0.2.tgz",

      "integrity": "sha512-QXcEdfmODc0rKblBerk30yw70fypIkFm6gQBLJgsshpwc+TMA+fuMLcPQebOTzKLtD2tNUkk/7SrWPQIGqeXaA==",

      "dev": true,

      "dependencies": {

        "ajv": "8.13.0",

        "ajv-formats": "3.0.1",

        "jsonc-parser": "3.2.1",

        "picomatch": "4.0.2",

        "rxjs": "7.8.1",

        "source-map": "0.7.4"

      },

      "engines": {

        "node": "^18.19.1 || ^20.11.1 || >=22.0.0",

        "npm": "^6.11.0 || ^7.5.6 || >=8.0.0",

        "yarn": ">= 1.13.0"

      },

      "peerDependencies": {

        "chokidar": "^3.5.2"

      },

      "peerDependenciesMeta": {

        "chokidar": {

          "optional": true

        }

      }

    },

    "node\_modules/@angular-devkit/schematics": {

      "version": "18.0.2",

      "resolved": "https://registry.npmjs.org/@angular-devkit/schematics/-/schematics-18.0.2.tgz",

      "integrity": "sha512-G9yGcoB67sH0eRNWoiQWNn2KwiI7sDasVscYPGKf1yo7JRiXmzX/LpfKRPsZTl+Bs0FItnwDInsqgMisK89/6g==",

      "dev": true,

      "dependencies": {

        "@angular-devkit/core": "18.0.2",

        "jsonc-parser": "3.2.1",

        "magic-string": "0.30.10",

        "ora": "5.4.1",

        "rxjs": "7.8.1"

      },

      "engines": {

        "node": "^18.19.1 || ^20.11.1 || >=22.0.0",

        "npm": "^6.11.0 || ^7.5.6 || >=8.0.0",

        "yarn": ">= 1.13.0"

      }

    },

    "node\_modules/@angular/animations": {

      "version": "18.0.1",

      "resolved": "https://registry.npmjs.org/@angular/animations/-/animations-18.0.1.tgz",

      "integrity": "sha512-QAY/oxfuFY2Bjr3foniWlLAiddXHu8879lZvXHt1NVOsiav+vD15IEEQsnuQbJPy/EHEnAlUh9UptB4zQIBp/Q==",

      "dependencies": {

        "tslib": "^2.3.0"

      },

      "engines": {

        "node": "^18.13.0 || >=20.9.0"

      },

      "peerDependencies": {

        "@angular/core": "18.0.1"

      }

    },

    "node\_modules/@angular/build": {

      "version": "18.0.2",

      "resolved": "https://registry.npmjs.org/@angular/build/-/build-18.0.2.tgz",

      "integrity": "sha512-iPPHdAJ3LiR8t/+39xjvrqMWcTmRrfphzKxXoIVDcswQjVQIk00EYuxinC6EVa7dSKDl1thk1MeCNZ9DIjaAvQ==",

      "dev": true,

      "dependencies": {

        "@ampproject/remapping": "2.3.0",

        "@angular-devkit/architect": "0.1800.2",

        "@babel/core": "7.24.5",

        "@babel/helper-annotate-as-pure": "7.22.5",

        "@babel/helper-split-export-declaration": "7.24.5",

        "@vitejs/plugin-basic-ssl": "1.1.0",

        "ansi-colors": "4.1.3",

        "browserslist": "^4.23.0",

        "critters": "0.0.22",

        "esbuild": "0.21.3",

        "fast-glob": "3.3.2",

        "https-proxy-agent": "7.0.4",

        "inquirer": "9.2.22",

        "lmdb": "3.0.8",

        "magic-string": "0.30.10",

        "mrmime": "2.0.0",

        "ora": "5.4.1",

        "parse5-html-rewriting-stream": "7.0.0",

        "picomatch": "4.0.2",

        "piscina": "4.5.0",

        "sass": "1.77.2",

        "semver": "7.6.2",

        "undici": "6.18.0",

        "vite": "5.2.11",

        "watchpack": "2.4.1"

      },

      "engines": {

        "node": "^18.19.1 || ^20.11.1 || >=22.0.0",

        "npm": "^6.11.0 || ^7.5.6 || >=8.0.0",

        "yarn": ">= 1.13.0"

      },

      "peerDependencies": {

        "@angular/compiler-cli": "^18.0.0",

        "@angular/localize": "^18.0.0",

        "@angular/platform-server": "^18.0.0",

        "@angular/service-worker": "^18.0.0",

        "less": "^4.2.0",

        "postcss": "^8.4.0",

        "tailwindcss": "^2.0.0 || ^3.0.0",

        "typescript": ">=5.4 <5.5"

      },

      "peerDependenciesMeta": {

        "@angular/localize": {

          "optional": true

        },

        "@angular/platform-server": {

          "optional": true

        },

        "@angular/service-worker": {

          "optional": true

        },

        "less": {

          "optional": true

        },

        "postcss": {

          "optional": true

        },

        "tailwindcss": {

          "optional": true

        }

      }

    },

    "node\_modules/@angular/cdk": {

      "version": "18.0.1",

      "resolved": "https://registry.npmjs.org/@angular/cdk/-/cdk-18.0.1.tgz",

      "integrity": "sha512-2fCqX1sz5cM+LncO6ak4EU2ZBm8MWitv5V53go3Iz5dOVOdrvysBt8smEkWZ4nvEKkFYHEPpQo0YlxEWbuTEmA==",

      "dependencies": {

        "tslib": "^2.3.0"

      },

      "optionalDependencies": {

        "parse5": "^7.1.2"

      },

      "peerDependencies": {

        "@angular/common": "^18.0.0 || ^19.0.0",

        "@angular/core": "^18.0.0 || ^19.0.0",

        "rxjs": "^6.5.3 || ^7.4.0"

      }

    },

    "node\_modules/@angular/cli": {

      "version": "18.0.2",

      "resolved": "https://registry.npmjs.org/@angular/cli/-/cli-18.0.2.tgz",

      "integrity": "sha512-shrxMD1bcWWh7WpBN3KTV+Lt8E62gURSUFhs6kdGLepMDif8LPAv45+hpt8SBU9VfQuL6AHa4cW8uDL9BKGlYA==",

      "dev": true,

      "dependencies": {

        "@angular-devkit/architect": "0.1800.2",

        "@angular-devkit/core": "18.0.2",

        "@angular-devkit/schematics": "18.0.2",

        "@schematics/angular": "18.0.2",

        "@yarnpkg/lockfile": "1.1.0",

        "ansi-colors": "4.1.3",

        "ini": "4.1.2",

        "inquirer": "9.2.22",

        "jsonc-parser": "3.2.1",

        "npm-package-arg": "11.0.2",

        "npm-pick-manifest": "9.0.1",

        "ora": "5.4.1",

        "pacote": "18.0.6",

        "resolve": "1.22.8",

        "semver": "7.6.2",

        "symbol-observable": "4.0.0",

        "yargs": "17.7.2"

      },

      "bin": {

        "ng": "bin/ng.js"

      },

      "engines": {

        "node": "^18.19.1 || ^20.11.1 || >=22.0.0",

        "npm": "^6.11.0 || ^7.5.6 || >=8.0.0",

        "yarn": ">= 1.13.0"

      }

    },

    "node\_modules/@angular/common": {

      "version": "18.0.1",

      "resolved": "https://registry.npmjs.org/@angular/common/-/common-18.0.1.tgz",

      "integrity": "sha512-iADQC5m4fvk+VNXEoU1KR93b0eG218/GuNdzUNVJHcjxdFxPshKk5fiaGSosUCxXPRQOxDKzmS9EDang87E/Ew==",

      "dependencies": {

        "tslib": "^2.3.0"

      },

      "engines": {

        "node": "^18.13.0 || >=20.9.0"

      },

      "peerDependencies": {

        "@angular/core": "18.0.1",

        "rxjs": "^6.5.3 || ^7.4.0"

      }

    },

    "node\_modules/@angular/compiler": {

      "version": "18.0.1",

      "resolved": "https://registry.npmjs.org/@angular/compiler/-/compiler-18.0.1.tgz",

      "integrity": "sha512-zyG/ifCtN0drAuwz0oV6LtzTiDREsM1Ay7eJW9wTvp3NCv06goHLtHXX12eFfZQWJViBv924lyRDSWdZN7r3GQ==",

      "dependencies": {

        "tslib": "^2.3.0"

      },

      "engines": {

        "node": "^18.13.0 || >=20.9.0"

      },

      "peerDependencies": {

        "@angular/core": "18.0.1"

      },

      "peerDependenciesMeta": {

        "@angular/core": {

          "optional": true

        }

      }

    },

    "node\_modules/@angular/compiler-cli": {

      "version": "18.0.1",

      "resolved": "https://registry.npmjs.org/@angular/compiler-cli/-/compiler-cli-18.0.1.tgz",

      "integrity": "sha512-Aoz70+/o8R2lG2EGDAYbj6yu2B7kqa/9loYEwG0fECJTtXoRBP+bEGpUxMmxOb59tMDnbIhBHmNPPEQVTXvgSQ==",

      "dev": true,

      "dependencies": {

        "@babel/core": "7.24.4",

        "@jridgewell/sourcemap-codec": "^1.4.14",

        "chokidar": "^3.0.0",

        "convert-source-map": "^1.5.1",

        "reflect-metadata": "^0.2.0",

        "semver": "^7.0.0",

        "tslib": "^2.3.0",

        "yargs": "^17.2.1"

      },

      "bin": {

        "ng-xi18n": "bundles/src/bin/ng\_xi18n.js",

        "ngc": "bundles/src/bin/ngc.js",

        "ngcc": "bundles/ngcc/index.js"

      },

      "engines": {

        "node": "^18.13.0 || >=20.9.0"

      },

      "peerDependencies": {

        "@angular/compiler": "18.0.1",

        "typescript": ">=5.4 <5.5"

      }

    },

    "node\_modules/@angular/compiler-cli/node\_modules/@babel/core": {

      "version": "7.24.4",

      "resolved": "https://registry.npmjs.org/@babel/core/-/core-7.24.4.tgz",

      "integrity": "sha512-MBVlMXP+kkl5394RBLSxxk/iLTeVGuXTV3cIDXavPpMMqnSnt6apKgan/U8O3USWZCWZT/TbgfEpKa4uMgN4Dg==",

      "dev": true,

      "dependencies": {

        "@ampproject/remapping": "^2.2.0",

        "@babel/code-frame": "^7.24.2",

        "@babel/generator": "^7.24.4",

        "@babel/helper-compilation-targets": "^7.23.6",

        "@babel/helper-module-transforms": "^7.23.3",

        "@babel/helpers": "^7.24.4",

        "@babel/parser": "^7.24.4",

        "@babel/template": "^7.24.0",

        "@babel/traverse": "^7.24.1",

        "@babel/types": "^7.24.0",

        "convert-source-map": "^2.0.0",

        "debug": "^4.1.0",

        "gensync": "^1.0.0-beta.2",

        "json5": "^2.2.3",

        "semver": "^6.3.1"

      },

      "engines": {

        "node": ">=6.9.0"

      },

      "funding": {

        "type": "opencollective",

        "url": "https://opencollective.com/babel"

      }

    },

    "node\_modules/@angular/compiler-cli/node\_modules/@babel/core/node\_modules/convert-source-map": {

      "version": "2.0.0",

      "resolved": "https://registry.npmjs.org/convert-source-map/-/convert-source-map-2.0.0.tgz",

      "integrity": "sha512-Kvp459HrV2FEJ1CAsi1Ku+MY3kasH19TFykTz2xWmMeq6bk2NU3XXvfJ+Q61m0xktWwt+1HSYf3JZsTms3aRJg==",

      "dev": true

    },

    "node\_modules/@angular/compiler-cli/node\_modules/@babel/core/node\_modules/semver": {

      "version": "6.3.1",

      "resolved": "https://registry.npmjs.org/semver/-/semver-6.3.1.tgz",

      "integrity": "sha512-BR7VvDCVHO+q2xBEWskxS6DJE1qRnb7DxzUrogb71CWoSficBxYsiAGd+Kl0mmq/MprG9yArRkyrQxTO6XjMzA==",

      "dev": true,

      "bin": {

        "semver": "bin/semver.js"

      }

    },

    "node\_modules/@angular/core": {

      "version": "18.0.1",

      "resolved": "https://registry.npmjs.org/@angular/core/-/core-18.0.1.tgz",

      "integrity": "sha512-Db1livvugoLdLsWww5IqUS5v+yUN7/5Rj0trZv9BgxIuoNtoipfLqKHwZWpumH3yI5Ucu+UH9zZ1mlGyF0Kexw==",

      "dependencies": {

        "tslib": "^2.3.0"

      },

      "engines": {

        "node": "^18.13.0 || >=20.9.0"

      },

      "peerDependencies": {

        "rxjs": "^6.5.3 || ^7.4.0",

        "zone.js": "~0.14.0"

      }

    }

**Over view of javascript**

JavaScript is a high-level, interpreted programming language that is primarily used to create interactive effects within web browsers. Initially developed by Netscape as a client-side scripting language, it has since evolved into a versatile language capable of server-side development, mobile app development, and more.

Key Features

Interpreted Language: JavaScript code is executed directly by the browser's JavaScript engine, which allows for immediate execution without prior compilation.

Dynamic Typing: Variables in JavaScript are not bound to a specific data type, allowing for flexibility in variable assignments.

Prototype-Based Object Orientation: Instead of traditional class-based inheritance, JavaScript uses prototypes, enabling objects to inherit properties and methods from other objects.

First-Class Functions: Functions in JavaScript are treated as first-class citizens, meaning they can be assigned to variables, passed as arguments, and returned from other functions.

Event-Driven: JavaScript supports event-driven programming, allowing developers to create responsive applications that react to user interactions like clicks and key presses.

Asynchronous Programming: Through the use of callbacks, promises, and async/await syntax, JavaScript can handle asynchronous operations, making it well-suited for tasks like network request.

Ecosystem and Compatibility

JavaScript is universally supported across all modern web browsers, making it a standard for web development. The language has a rich ecosystem, including:

Frameworks and Libraries: Popular frameworks like React, Angular, and Vue.js facilitate the development of complex user interfaces. Libraries such as jQuery simplify DOM manipulation.

Node.js: An environment that allows JavaScript to be used for server-side development, enabling full-stack development with a single language.

Package Managers: Tools like npm (Node Package Manager) and Yarn allow developers to manage and share packages, enhancing modularity and reusability.

Use Cases

Web Development: Enhancing user interfaces and providing interactive features on websites.

Server-Side Development: Using Node.js to build scalable network applications.

Mobile Development: Frameworks like React Native enable building mobile applications using JavaScript.

Game Development: Libraries like Phaser are designed for creating 2D games in JavaScript.

Desktop Applications: Tools like Electron allow developers to create cross-platform desktop applications.

Modern JavaScript (ES6+)

The evolution of JavaScript has led to the introduction of several modern features through ECMAScript (the specification for JavaScript). Key features include:

Arrow Functions: A shorter syntax for writing functions.

Template Literals: Enhanced string literals for multi-line strings and embedded expressions.

Modules: Native support for modular programming, allowing for better organization of code.

Promises and Async/Await: Enhanced handling of asynchronous operations for cleaner code.

Conclusion

JavaScript remains an essential language in the development world, continually evolving to meet the demands of modern applications. Its flexibility, widespread support, and rich ecosystem make it a vital tool for developers across various domains. Understanding its features and capabilities is crucial for anyone looking to leverage the full power of web technologies.

**Overview of APIs**

An API (Application Programming Interface) is a set of protocols and tools that allows different software applications to communicate with each other. APIs enable integration between systems, allowing them to share data and functionalities.

Key Concepts:

Endpoints: Specific URLs where API requests are directed. Each endpoint represents a distinct function or resource.

HTTP Methods: Common methods used in API requests include:

GET: Retrieve data from the server.

POST: Submit data to the server for processing.

PUT: Update existing data.

DELETE: Remove data.

Requests: A structured way to ask for data or send data to an API. Key components of a request include:

Headers: Metadata such as authentication tokens and content type.

Parameters: Additional data sent in the URL (query parameters) or the body of the request.

Responses: The server’s reply to an API request, which typically includes:

Status Codes: Indicate the success or failure of the request (e.g., 200 for success, 404 for not found).

Data: The actual content returned, usually in formats like JSON or XML.

Authentication: Mechanisms to secure API access, ensuring that only authorized users can interact with the API. Common methods include API keys and OAuth tokens.

**Overview of the tools used here:**

# Node

/node\_modules

npm-debug.log

yarn-error.log

# IDEs and editors

.idea/

.project

.classpath

.c9/

\*.launch

.settings/

\*.sublime-workspace

# Visual Studio Code

.vscode/\*

!.vscode/settings.json

!.vscode/tasks.json

!.vscode/launch.json

!.vscode/extensions.json

.history/\*

ENTITIES OF LIVE-SPORT-APP

In the provided package.json file for a project named "live-sports-app", the following entities are defined:

1. \*name\*: The name of the project (live-sports-app).

2. \*version\*: The current version of the project (0.0.0).

3. \*scripts\*: Various scripts defined to facilitate different tasks:

- ng: Alias for Angular CLI (ng).

- start: Command to start the development server (ng serve).

- build: Command to build the project (ng build).

- watch: Command to build the project in watch mode for development (ng build --watch --configuration development).

- test: Command to run tests (ng test).

4. \*private\*: A boolean flag (true) indicating that the package should not be published (typical for projects not intended for distribution).

5. \*dependencies\*: List of packages required for the application to run:

- Various Angular libraries (@angular/\*).

- RxJS (rxjs).

- TypeScript helper (tslib).

- Zone.js (zone.js).

6. \*devDependencies\*: List of packages required during development:

- Angular build tools (@angular-devkit/build-angular, @angular/cli, @angular/compiler-cli).

- Jasmine testing framework (@types/jasmine, jasmine-core, karma, karma-chrome-launcher, karma-coverage, karma-jasmine, karma-jasmine-html-reporter).

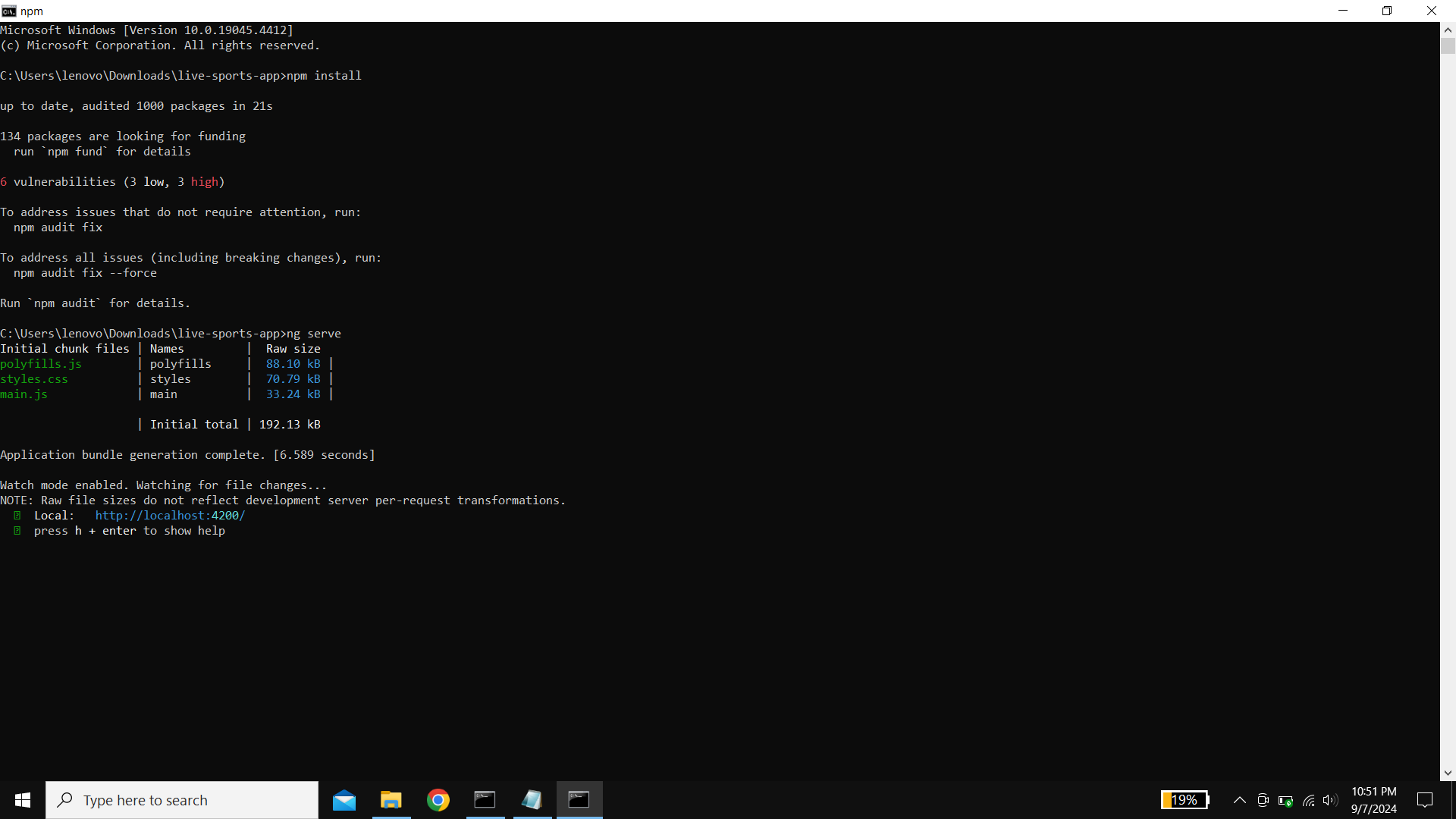
These entities define the structure and dependencies of the live-sports-app

CHAPTER 5

TESTING

5.1 UNIT TESTING

Unit testing is a crucial practice in software development that involves testing individual components or functions of an application in isolation. The primary aim is to verify that each unit performs as expected, ensuring that the code behaves correctly under various conditions. This process typically employs automated tests, which can be run quickly and frequently, allowing developers to catch bugs early in the development cycle. Popular testing frameworks like Jest, Mocha, and JUnit facilitate the creation of these tests, enabling developers to write positive and negative test cases that assert expected outcomes. The benefits of unit testing include improved code quality, easier refactoring, and enhanced documentation through test cases. By adhering to best practices, such as writing tests alongside the code and keeping them focused, developers can maintain a robust codebase that minimizes future issues and increases overall reliability.



5.2 FINAL OUTPUT



**CONCLUSION**

The "**Live Sports App**" represents a significant step forward in delivering real-time sports information to fans of cricket and football. By integrating advanced technologies such as AngularJS, Node.js, and React.js, the app aims to provide a seamless and engaging user experience. The structured modules—completed, current, and upcoming matches—ensure that users have quick access to vital information in a user-friendly format. As development progresses, the focus will remain on enhancing performance, interactivity, and responsiveness across all devices. Ultimately, this project seeks to foster a deeper connection between fans and their favourite sports, transforming how they access and engage with live match data. With continuous feedback and improvements, the "Live Sports App" aspires to become an essential tool for sports enthusiasts everywhere.

**Project Achievements**

 **Successful Integration of Technologies**: The project effectively integrates a variety of frameworks, including AngularJS, Node.js, Angular, Zone.js, and React.js, showcasing the ability to leverage modern technologies for enhanced functionality and performance.

 **User-Centric Design**: The development of a user-friendly interface allows for easy navigation across the completed, current, and upcoming matches modules, ensuring that users can quickly access the information they need.

** Real-Time Data Processing**: Implementation of real-time updates for live matches enhances user engagement, providing instantaneous scores and updates, which significantly improves the overall user experience.

 **Modular Architecture**: The app’s modular design promotes scalability and maintainability, allowing for easy updates and the addition of new features in response to user feedback.

 **Cross-Device Compatibility**: The application is optimized for a variety of devices, ensuring a consistent and responsive experience whether accessed from desktops, tablets, or smartphones.

 **Enhanced User Engagement**: By incorporating interactive features such as notifications and personalized updates, the app fosters a deeper connection between fans and their favourite sports.

 **Comprehensive Match Coverage**: The app successfully covers a wide range of matches, providing detailed insights into completed, current, and upcoming cricket and football events.

 **Positive User Feedback**: Initial testing and feedback from users have highlighted the app’s strengths, confirming its potential to become a valuable resource for sports enthusiasts.

**Challenges Faced**

* **Technology Integration**: Combining multiple frameworks like AngularJS, Node.js, Angular, Zone.js, and React.js posed significant integration challenges. Ensuring compatibility and seamless communication between these technologies required extensive debugging and testing.
* **Real-Time Data Management:** Implementing real-time updates for live matches involved complexities in data handling and synchronization. Ensuring that users receive accurate and timely information demanded careful consideration of data flow and server performance.
* **User Interface Design:** Designing an intuitive and engaging user interface that effectively presents match information across multiple devices was challenging. Balancing aesthetics with functionality required iterative design processes and user feedback.
* **Performance Optimization:** Maintaining fast loading times and smooth interactions while managing large amounts of live data was a critical challenge. Continuous performance tuning was necessary to prevent lags and enhance user experience.
* **Testing Across Devices**: Ensuring cross-device compatibility brought its own set of difficulties. The app needed to perform well on various screen sizes and operating systems, necessitating extensive testing and adjustments.
* **User Engagement**: Developing features that genuinely enhance user engagement, such as notifications and personalized content, required careful planning and user research to understand fan preferences and behaviors.
* **Scalability Concerns:** As the app was designed to handle potentially large volumes of traffic, planning for scalability was essential. This involved architectural decisions to ensure that the app could grow and accommodate an increasing number of users.
* **Maintaining Data Accuracy:** Providing accurate match results and statistics demanded reliable data sources and frequent updates. Establishing partnerships or API integrations with trustworthy sports data providers was a critical step that required negotiation and technical implementation.

**Future Considerations**

* **Feature Expansion:** To enhance user engagement, future updates could introduce features such as player statistics, team comparisons, and historical match data. Additionally, incorporating fantasy sports elements might attract a broader audience.
* **Enhanced Personalization**: Implementing machine learning algorithms to provide personalized content and notifications based on user preferences and behaviors can improve user satisfaction and retention.
* **Multi-Sport Coverage:** While the initial focus is on cricket and football, expanding the app to include other popular sports like basketball, tennis, and rugby could significantly broaden the user base.
* **Social Integration:** Adding social media sharing options and community features, such as forums or chat rooms, would allow users to engage with one another, fostering a sense of community around the app.
* **Offline Access:** Developing a feature for offline access to match results and schedules can enhance usability, especially for users in areas with unreliable internet connectivity.
* **Mobile App Development**: While the web app is a solid start, creating dedicated mobile applications for iOS and Android could improve accessibility and performance, providing a more tailored experience for mobile users.
* **API Partnerships**: Establishing partnerships with reputable sports data providers can ensure reliable and up-to-date information, enhancing the app's credibility and performance.
* **User Feedback Mechanism**: Implementing a robust feedback system will allow users to suggest features and report issues, helping prioritize future updates based on user needs and preferences.
* **Analytics Integration**: Utilizing analytics tools to track user behaviour and engagement can provide insights that inform future development and marketing strategies.
* **Continuous Performance Optimization**: As the user base grows, ongoing performance testing and optimization will be necessary to maintain a fast and responsive experience.