International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 09 Issue: 06 | June - 2025 SJIF Rating: 8.586 ISSN: 2582-3930

CRICKET CONNECT

Dr.M. Shalima Sulthana , M. Vaishnavi, T.SwethaB.Manvitha
Associate Professor
UG Student
Department of Computer Science and Engineering
Vignan's Institute of Management and Technology for Women Ghatkesar,Hyderabad
Email:m.s.sulthana2012@gmail.com
Email:vaishnavimanchukonda6@gmail.com
Email:swethathumukunta87635@gmail.com
Email:biragonimanvitha@gmail.com

***Abstract*— Cricket ConnectCricket Connect is a dynamic, user-friendly web platform designed to serve as a one-stop destination for all cricket enthusiasts. Built using HTML, CSS, JavaScript, MongoDB and Python, this platform integrates multiple cricket-related features, including a specialized chatbot that answers cricket- specific queries, real-time live scores, IPL updates, and match information.**

The system offers user authentication via Login and Signup pages, enabling personalized access and interaction. It also features live app functionality and a chat discussion module where users can engage in real-time conversations about ongoing matches, teams, and player performances.

At the core of the platform is the cricket chatbot using Gemini ai, which delivers fast, relevant responses to questions about cricket rules, schedules, players, and history. The live score module fetches real-time data from external cricket APIs, keeping users updated with the latest developments. The IPL section presents dedicated content, match results, and team statistics in an intuitive and visually engaging layout.

The frontend ensures a smooth and responsive user experience through modern UI/UX practices using HTML, CSS, and JavaScript, while Python handles server-side logic, data integration, and chatbot intelligence.

Cricket Connect aims to create an interactive, real-time, and informative ecosystem that brings cricket fans together through technology.

I.INTRODUCTION

In the digital age, sports enthusiasts increasingly rely on online platforms for real-time updates, intelligent assistance, and interactive experiences. Cricket, being one of the most followed sports globally—especially in countries like India, Australia, England, and South Africa—demands a technology-driven ecosystem that caters to fans' dynamic information needs. Existing sports platforms often provide fragmented services, lacking intelligent interaction, unified access to data, or community engagement features.

Cricket Connect is proposed as a comprehensive web-based platform that addresses these challenges by integrating multiple cricket-focused services into a single, user-centric interface. The platform is built using HTML, CSS, and JavaScript for the frontend and Python for backend logic. It is underpinned by MongoDB, a NoSQL database that ensures flexible, scalable data management for user records, match data, and chat logs.

The distinguishing feature of Cricket Connect is its integrated AI-powered chatbot, developed using Google's Gemini AI. Unlike traditional rule-based bots, Gemini AI leverages advanced natural language understanding to interpret, process, and respond to cricket-related queries with human-like fluency and contextual awareness. Users can ask questions about live matches, team statistics, historical records, rules of the game, and upcoming fixtures—and receive intelligent, conversational answers in real-time.

Beyond intelligent Q&A, the platform offers a range of features to enhance user engagement and experience. It includes Login and Signup pages for personalized access, live scores synchronized with external cricket APIs, and an interactive IPL module that displays current standings, schedules, and detailed team information. The platform is also compatible with live mobile apps, allowing users to stay connected on the go. Additionally, a discussion forum enables fans to participate in real-time chats about ongoing matches and the latest cricket news.

The distinguishing feature of Cricket Connect is its integration of Gemini AI, an intelligent chatbot designed to provide users with real-time match updates, IPL insights, player statistics, and contextual cricket knowledge. Additionally, the platform incorporates community-driven features such as live chat, match discussions, and fan forums, fostering an interactive digital cricket ecosystem. By combining information retrieval, intelligent interaction, and social engagement, Cricket Connect aims to redefine how cricket fans experience the sport online

.

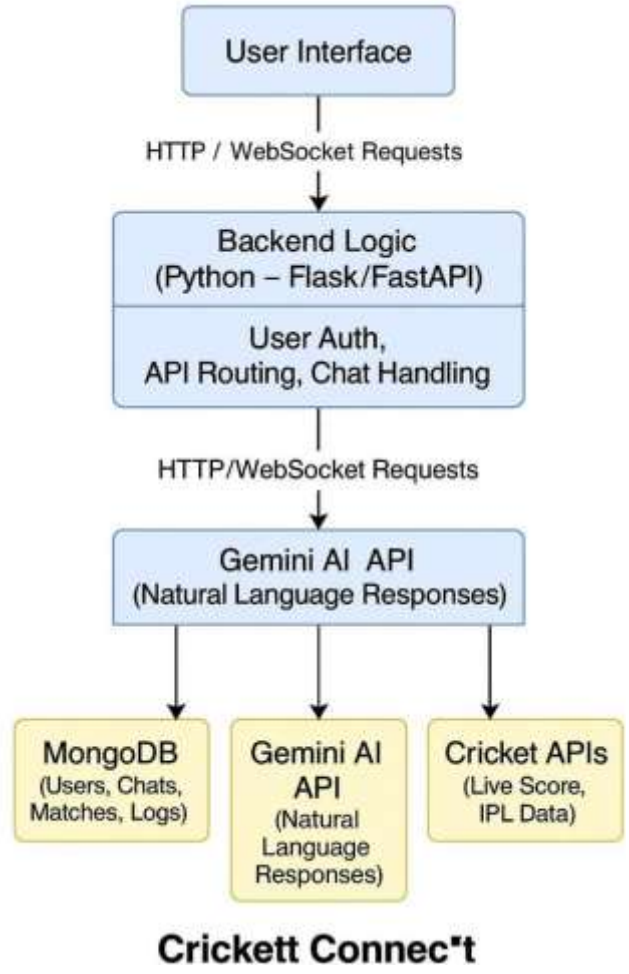
II. LITERATURE SURVEY

The convergence of artificial intelligence, real-time data streaming, and web technologies has transformed how sports content is delivered and consumed. Numerous platforms such as ESPN, Cricbuzz, and the ICC official site have successfully provided cricket updates and analytics. However, they typically lack conversational AI interfaces and community-centric features tailored to specific user needs.

Recent research in conversational agents (e.g., BERT, ChatGPT, Gemini AI) has highlighted the potential of LLMs (Large Language Models) in providing contextual assistance. Studies have demonstrated that AI-driven chatbots can improve user engagement, personalization, and content accessibility across domains including education, customer support, and healthcare. However, their application in sports platforms remains underexplored.

MongoDB has emerged as a scalable, document-based database that fits unstructured and semi-structured data such as chat logs, user queries, and match details. Its ability to handle real-time writes makes it suitable for platforms that support live interactions and dynamic content.

The integration of these technologies into a single sports platform presents an opportunity to enhance fan engagement, offer intelligent support, and democratize cricket knowledge. The development of Cricket Connect contributes to this niche by combining Gemini AI, real-time APIs, and interactive community features, creating a next-gen solution for cricket enthusiast.



III. SYSTEM ARCHIECHTURE

The architecture of Cricket Connect is structured around a modular client-server model that ensures high performance, real- time interaction, and easy scalability. The frontend is developed using HTML, CSS, and JavaScript, and it provides a clean, intuitive user interface that includes the login and signup pages, live scoreboards, match details, IPL data visualizations, and the chatbot/discussion sections. This layer ensures full responsiveness across browsers and mobile devices and communicates with the backend using RESTful APIs and WebSockets.

The backend, built using Python with a web framework like Flask or FastAPI, acts as the central processing unit. It is responsible for managing business logic, authenticating users, handling requests from the frontend, integrating live cricket data from third-party APIs, and connecting with the Gemini AI chatbot service. This backend layer routes user queries from the chatbot interface to the Gemini AI API, processes the responses, and sends back intelligent, contextual replies based on cricket-specific prompts.

To handle data persistence and retrieval efficiently, MongoDB is used as the backend database. It stores collections such as user profiles, chat logs, match data, IPL schedules, and live discussion threads. MongoDB's document-based structure supports dynamic schemas, making it highly suitable for storing both structured (e.g., user accounts) and semi-structured data (e.g., chatbot logs, real-

time match info). Additionally, a WebSocket-based channel enables real-time chat and discussions, allowing fans to communicate live during match events.

IV. METHODOLOGY

The development of Cricket Connect followed a structured and iterative methodology aimed at integrating intelligent conversational systems with real-time cricket data access. The core objective was to build a user-centric web platform that not only delivers timely cricket updates but also fosters engagement through interactive features, including a cricket-specific chatbot. The methodology was divided into several key phases, beginning with requirement gathering and analysis. During this phase, a comprehensive study was conducted to understand the expectations of cricket enthusiasts and evaluate the technical viability of incorporating a large language model (LLM)-driven chatbot. Requirements were categorized into functional components such as user authentication, live score updates, IPL highlights, and interactive chat features, as well as non-functional requirements like scalability, low latency, and real-time responsiveness.

Following this, an appropriate technology stack was selected to address the identified needs. The frontend was developed using HTML5, CSS3, and JavaScript to ensure responsive and intuitive user interfaces. For the backend, Python frameworks such as Flask or FastAPI were chosen to handle business logic and RESTful API endpoints. MongoDB served as the database due to its document-oriented design and scalability. Gemini AI, Google's advanced LLM, was integrated as the core conversational engine to respond to cricket-related queries. Additionally, third-party cricket APIs such as CricAPI or RapidAPI were employed for retrieving live match data, while WebSockets facilitated real-time communication features for chat-based discussions.

The system was then architected using a modular design approach to promote separation of concerns and ease of maintenance. Each feature—such as login/signup, chatbot interactions, live match data visualization, and chat room discussions—was developed independently as a discrete module. The frontend incorporated user-friendly interfaces for navigating between live scores, match details, chatbot queries, and chat discussions. On the backend, RESTful APIs were developed to manage data flow, process user inputs, and integrate with external APIs. A communication bridge was established to seamlessly connect user inputs with Gemini AI's endpoints, managing prompts, categorizing queries, and delivering context-aware responses. MongoDB collections were structured to store information related to users, match data, chat logs, and chatbot sessions efficiently.

Testing and validation played a critical role in ensuring system reliability. Unit tests were conducted for individual components such as backend APIs, chatbot logic, and user interface behavior. Integration testing was carried out to confirm seamless interaction among the frontend, backend, database, and third-party APIs. Load testing was also performed to assess system performance under concurrent user interactions, particularly during live match sessions when real-time responsiveness is essential.

Finally, the deployment phase involved launching the platform on scalable and accessible cloud services. The frontend was deployed using platforms such as Netlify or GitHub Pages for quick and secure access. The backend APIs were hosted on cloud services like Heroku or AWS, while MongoDB was managed via MongoDB Atlas to ensure high availability and simplified cloud storage management. This end-to-end methodology enabled the successful development of Cricket Connect—a robust, scalable, and engaging platform that provides users with intelligent cricket-related services and real-time interaction capabilities.

V. ALGORITHM

1. Start the Application
2. User Login/Signup
   * If new user → Signup → Save to MongoDB
   * If existing user → Login → Validate from MongoDB
   * If valid → Go to Dashboard
   * Else → Show error message
3. Chatbot Query Handling
   * User enters a cricket-related question
   * Send question to *Gemini AI API*
   * Get and display the response
   * Save conversation to MongoDB
4. Fetch Live Scores and Match Data
   * Connect to Cricket API
   * Get current scores, schedules, IPL data
   * Show results on the dashboard
   * Refresh data at regular intervals (e.g., every 20 seconds)
5. Chat Discussion Feature
   * User sends a message in discussion forum
   * Message is sent through WebSocket
   * All connected users see the message live
   * Store message in MongoDB
6. Logout
   * End session and redirect to login page

VI. RESULT

FIG: Home page with the chatbot integrated with gemini ai

FIG: Live Matches

FIG: Account creation and Signup.

FIG:Live Score

FIG: Login page

FIG :Group Chat

VII. CONCLUSION

Cricket Connect successfully demonstrates the integration of conversational AI and real-time data delivery in a unified, user-friendly platform tailored for cricket enthusiasts. By combining web technologies like HTML, CSS, and JavaScript with a Python backend and a MongoDB database, the platform provides seamless user interaction, live match updates, IPL-specific content, and community-driven chat discussions. The inclusion of Gemini AI as the chatbot engine adds an intelligent and engaging layer to the user experience, offering accurate and relevant responses to cricket-related queries.

The platform not only serves as an information hub but also fosters interaction and engagement among fans through discussion features. It emphasizes modularity, scalability, and responsiveness, making it adaptable for future improvements. Overall, Cricket Connect offers a practical and innovative solution for digital sports engagement, highlighting the potential of AI-powered web applications in enhancing the way users consume and interact with sports content online

VIII. FUTURE SCOPE

The Cricket Connect platform, while functional and user-friendly, offers numerous opportunities for future development and enhancement. These possibilities can improve performance, expand functionality, and enrich the user experience. Below are some key areas for future scope:

Voice-Based Interaction

Integrating voice commands using Speech-to-Text and Text-to-Speech APIs would allow users to interact with the chatbot hands-free, making the platform more accessible.

Multilingual Chatbot Support

Expanding the chatbot's language capabilities beyond English to support regional Indian languages such as Hindi, Tamil, Telugu, etc., would allow broader audience engagement.

Mobile App Development

Developing native Android and iOS applications would improve accessibility and user convenience, especially during live matches.

Fantasy Cricket Integration

Integrating fantasy league features, where users can create and manage teams, could boost user engagement and retention.

Push Notifications for Live Updates

Enabling browser and mobile push notifications for live scores, upcoming matches, and breaking news would enhance real-time engagement.

AI-Powered Match Predictions

Implementing machine learning models to predict match outcomes, player performance, and winning probabilities could add a predictive analytics layer to the platform.

Gamification Features

Adding badges, leaderboards, and user achievements for active participants in discussions or quizzes can enhance user involvement.

Sentiment Analysis in Chat

Using AI to analyze user sentiment in chat discussions can help detect offensive content and maintain a positive environment.

Offline Mode (Progressive Web App)

Making the web app installable as a Progressive Web App (PWA) would allow users to access cached content even without internet connectivity.

Admin Panel for Content Moderation

Creating an admin dashboard to monitor user activity, chat discussions, and manage content would ensure security and maintain platform integrity

IX. REFERENCES

1. “Current trends Jakaria Apu (2024). A Data- Driven Approach to Predict Scores in T20 Cricket Match Using Machine Learning Classifier.

https://www.irjet.net/archives/V8/i12/IRJET-V8I12205.pdf

2. Polepaka Sanjeeva, Jampana Ajith Varma, Valaparla Sathvik, Attemla Abhinav Sai Ratan, Sanjay Mishra (2023). Automated Cricket Score Prediction.

https://ijarcce.com/wp-content/uploads/2021/03/IJARCCE.2021.10223.pdf

3. J. Verma (2022). Real-Time Cricket Match Analysis Using

AI and Big Data.

https://www.ijnrd.org/papers/IJNRDTH000_99.pdf

4. Parag Shah (2017). Predicting Outcome of Live Cricket Match Using DuckworthLewis Par Score.

https://ijrpr.com/uploads/V3ISSUE4/IJRPR352_8.pdf