Sports Content Generator

Course Name: GenAI

Institution Name: Medicaps University – Datagami Skill Based Course

Submitted by:

Sr no	Student Name	Enrolment Number
1.	AANCHAL MARATHE	EN22CS301015
2.	AAHMAD BURHANI	EN22CS301010
3.	ABHISHEK SANKHALA	EN22CS301035
4.	YASHASVI MODI	EN22CS3011114
5.	AKSHAT YADAV	EN22CS301095

Group Name:10 D1

Group Number: GAI-10

Industry Mentor Name:

University Mentor Name: AJAJ KHAN

Academic Year: 2026

Table of Contents

1. Introduction

The rapid growth of Artificial Intelligence has significantly transformed the way digital content is created and consumed. In the sports industry, professionals such as journalists, analysts, bloggers, and media teams are required to produce high-quality match summaries and reports within very short timeframes. Manual content creation is time-consuming, inconsistent in tone, and requires extensive domain knowledge.

The **Sports Content Generator** is an AI-powered application designed to automate the generation of professional sports content using Generative AI techniques. The system leverages Large Language Models (LLMs), Prompt Engineering, and Vector Databases to convert basic match information into structured and high-quality outputs such as match recaps.

By integrating Retrieval-Augmented Generation (RAG), the application retrieves relevant historical sports data and combines it with user inputs to produce accurate, contextual, and well-formatted reports. The project aims to reduce manual effort while maintaining professional writing standards and consistency across generated content.

Problem Statement

Sports professionals often need to create match summaries and analytical reports quickly after games. Writing these reports manually requires significant time, effort, and expertise. Existing general AI tools lack domain specialization and consistency in sports terminology and formatting.

Therefore, there is a need for a specialized AI system that can generate structured, professional sports content automatically while maintaining contextual accuracy.

1.1 Scope of the Document

- Generation of match recap content for multiple sports.
- Integration of LLM APIs for natural language generation.
- Storage and retrieval of sports data using Vector Database.
- Web-based user interface for input and output interaction.
- Backend API development using FastAPI.

1.2 Intended Audience

The Sports Content Generator is intended for sports journalists, content writers, analysts, media agencies, and team management who require fast and professional match reports. It also benefits sports bloggers, commentators, digital marketing teams, and social media managers for creating engaging sports content. Additionally, students, researchers, and

developers interested in Generative AI, prompt engineering, and content automation can use the system for learning and experimentation.

1.3 System Overview

The **Sports Content Generator** is an AI-powered web application designed to automatically generate sports-related content such as match recaps, highlights, summaries, and analytical descriptions based on user input.

The system combines Artificial Intelligence (LLM models**)** with a Vector Database (semantic search**)** to produce context-aware and realistic sports narratives. Instead of generating generic text, the application retrieves relevant past match information and combines it with user prompts to create high-quality content.

The platform follows a modular architecture consisting of:

- Frontend Interface (User Interaction)
- Backend API (Processing Layer)
- Vector Database (Knowledge Storage)
- AI Large Language Model (Content Generation Engine)

This architecture ensures scalability, faster response time, and intelligent content generation.

2. System Design

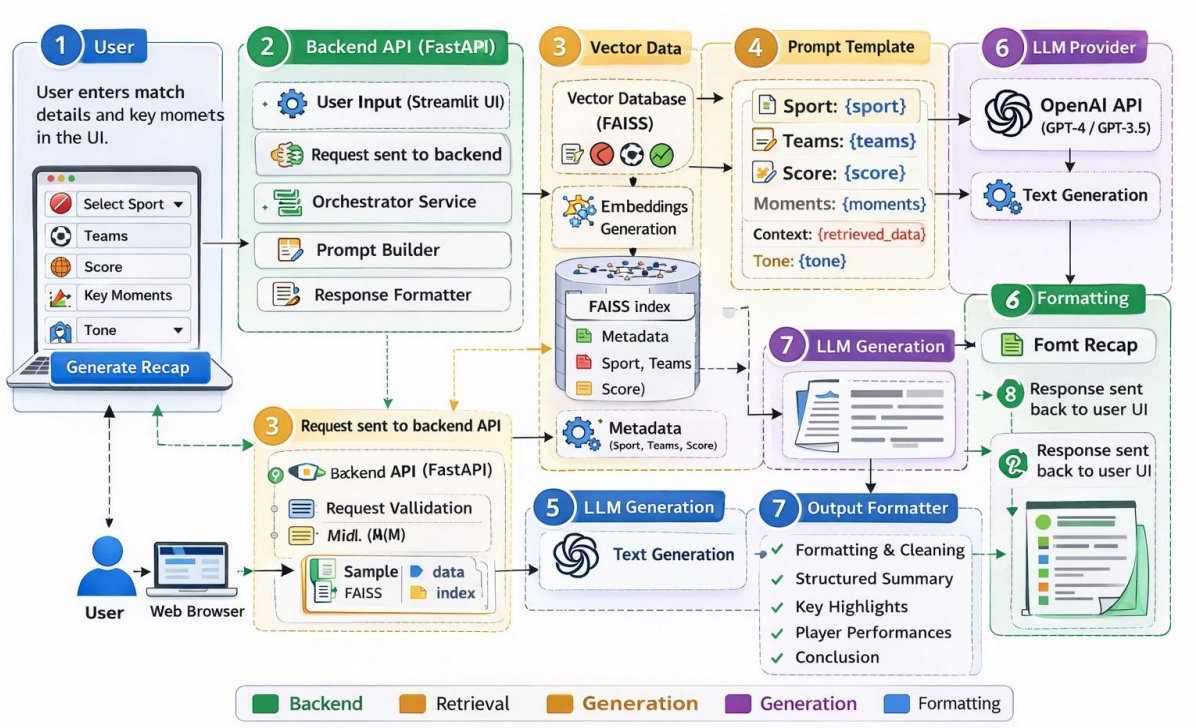
2.1 Application Design

The system follows a five-layer architecture where each layer is loosely coupled to allow independent scaling and maintainability.

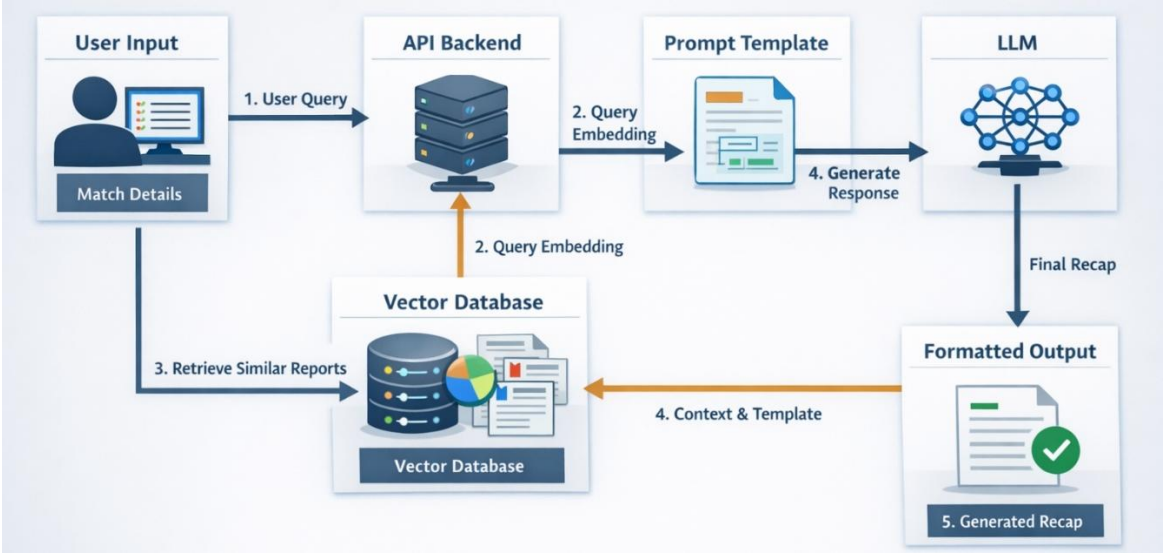
Module	Purpose	Technology
Frontend	Takes user input and displays generated sports content	Streamlit
API Layer	Handles requests between frontend and backend	FastAPI
Prompt Processing	Converts user data into AI prompts	Python
LLM Service	Generates match recap/content	OpenAI API
Vector Database	Stores and retrieves similar sports data	FAISS / Vector DB
Output Module	Shows final generated result	Web Interface

2.2 Process Flow

- **User Input** – User enters sport type, teams, and match details on the frontend.
- **Request Sent** – Frontend sends data to backend API.
- **Prompt Creation** – Backend formats input into a structured AI prompt.
- **Vector DB Retrieval** – Relevant sports context is fetched from the vector database.
- **AI Content Generation** – LLM generates professional match recap content.
- **Response Processing** – Backend formats the generated output.
- **Result Display** – Final sports content is shown on the frontend to the user.



2.3 Information Flow

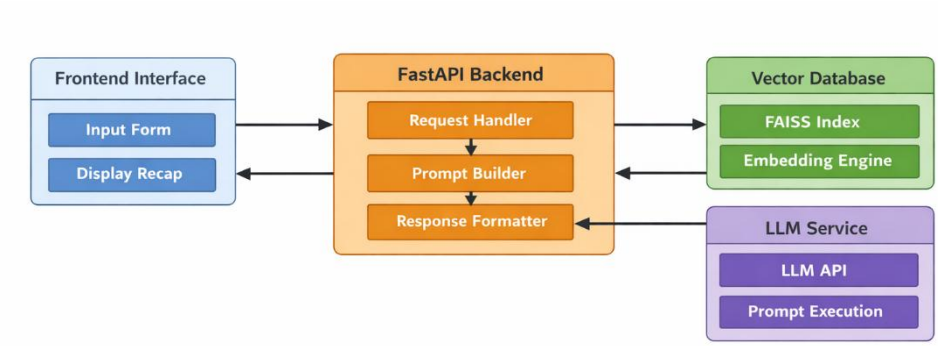


2.4 Components Design

Frontend: A streamlit based web interface serves as the user's entry point, capturing requests and displaying the final AI-generated content.

Backend (Fast API): This layer manages the core logic:

- The **Content Generator Service** coordinates with the AI models.
- The **Vector Search Service** handles the retrieval of specific sports data.
- **AI & Data Layer**: Employs Retrieval-Augmented Generation (RAG) by using an Embedding Model to search a FAISS Vector Database. This ensures the OpenAI LLM generates responses based on actual facts from the Sports Dataset rather than relying solely on pre-trained knowledge.
- **Storage**: A dedicated repository containing the Sports Dataset, which provides the foundational facts for all generated recaps.

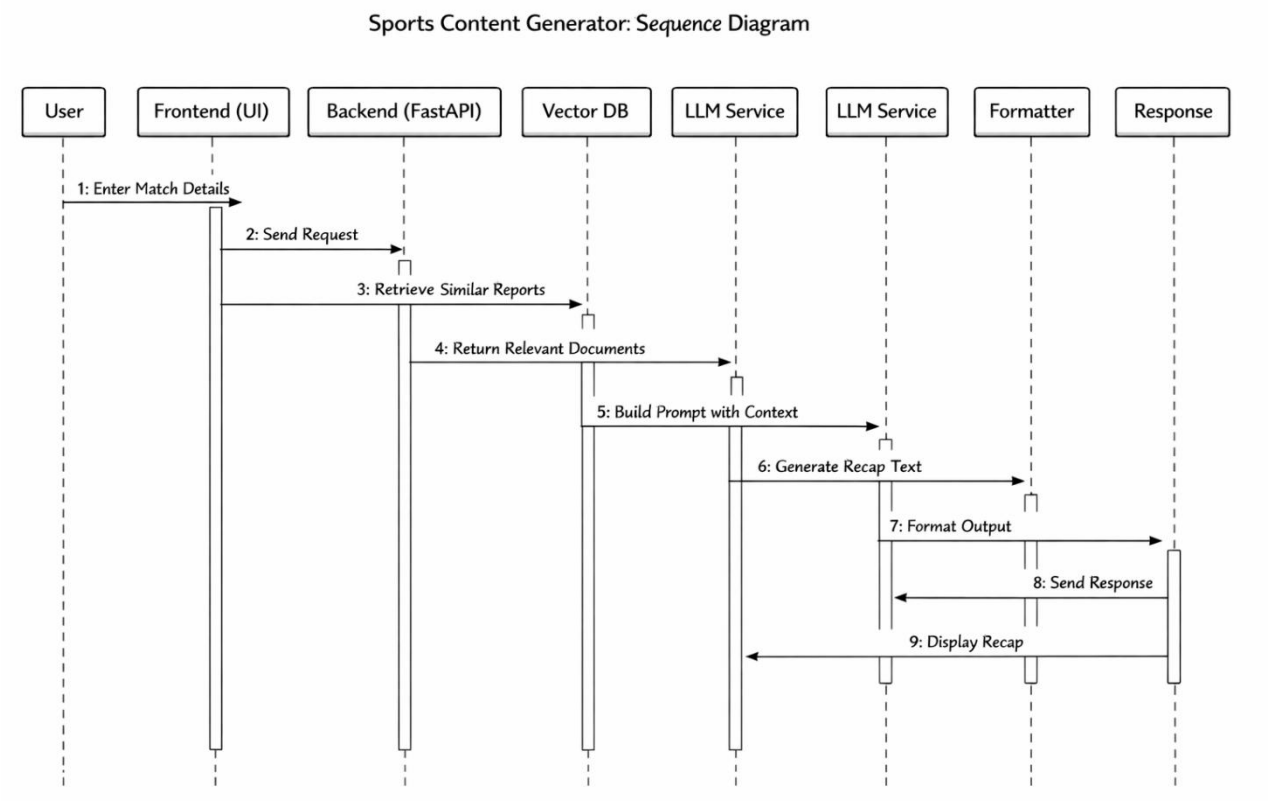


2.5 Key Design Considerations

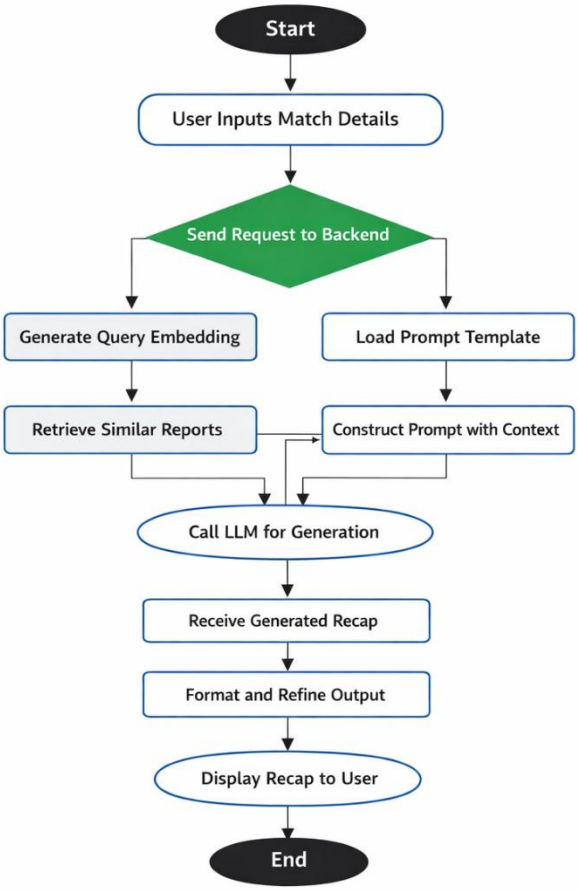
- **Scalability** – Supports multiple users and future expansion.
- **Performance** – Fast content generation using optimized APIs.
- **Accuracy** – Vector database provides relevant sports context.
- **Modular Design** – Separate frontend, backend, and AI services.
- **User-Friendly UI** – Simple and easy interaction.
- **Security** – API keys protected using environment variables.
- **Maintainability** – Clean and organized project structure.
- **Extensibility** – Easy to add new sports and features.

UML Diagrams:-

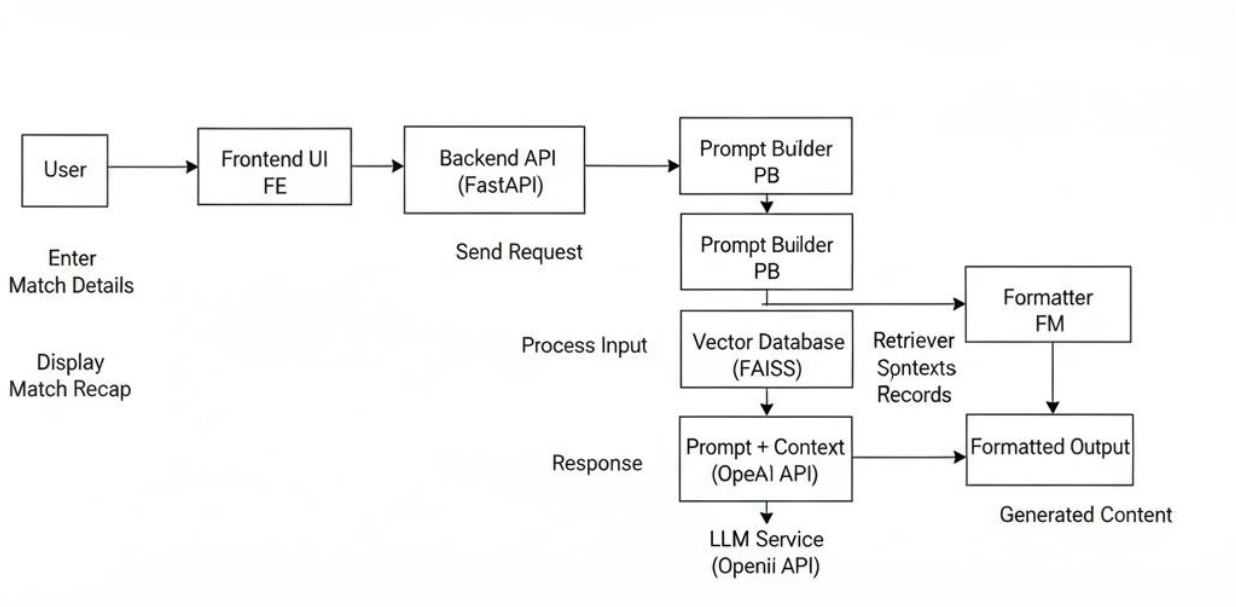
Sequence Diagram



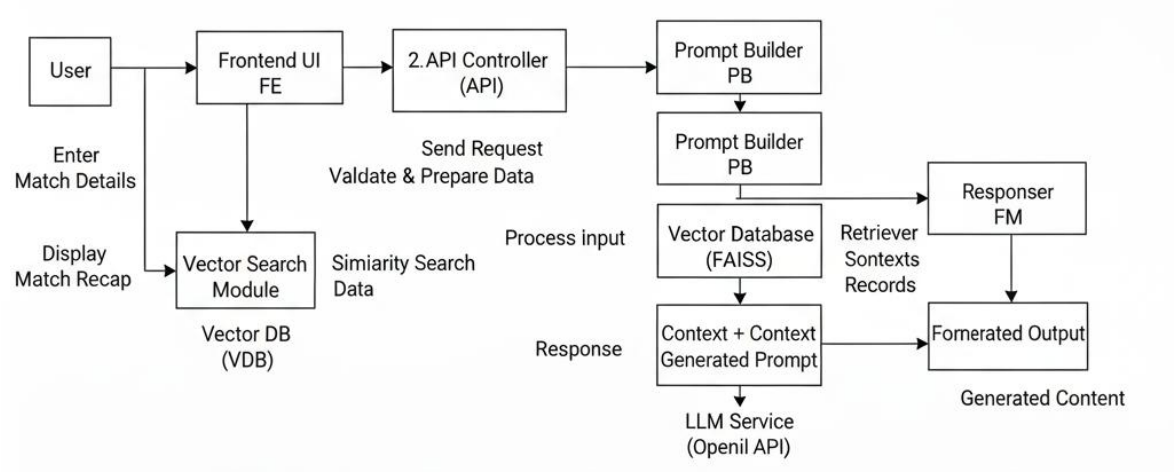
Activity Diagram



Data Flow Diagram - Level 0

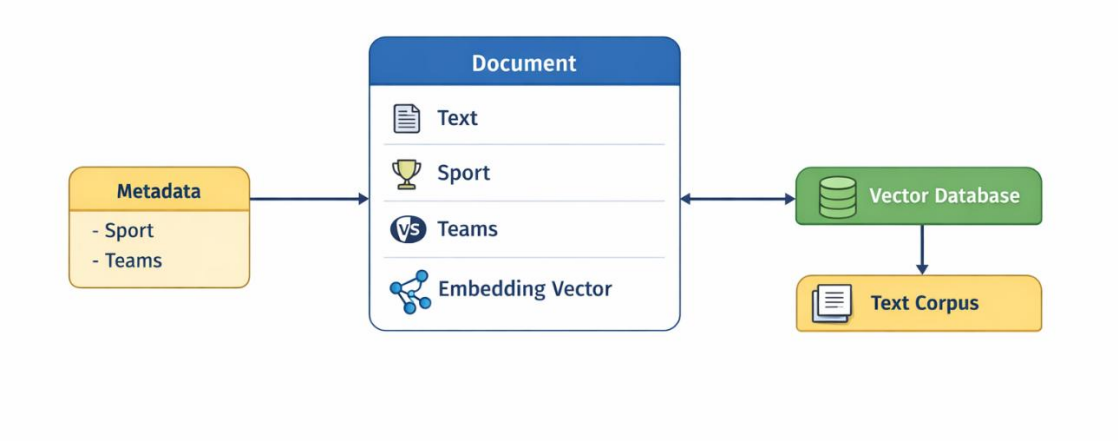


Data Flow Diagram – Level 1

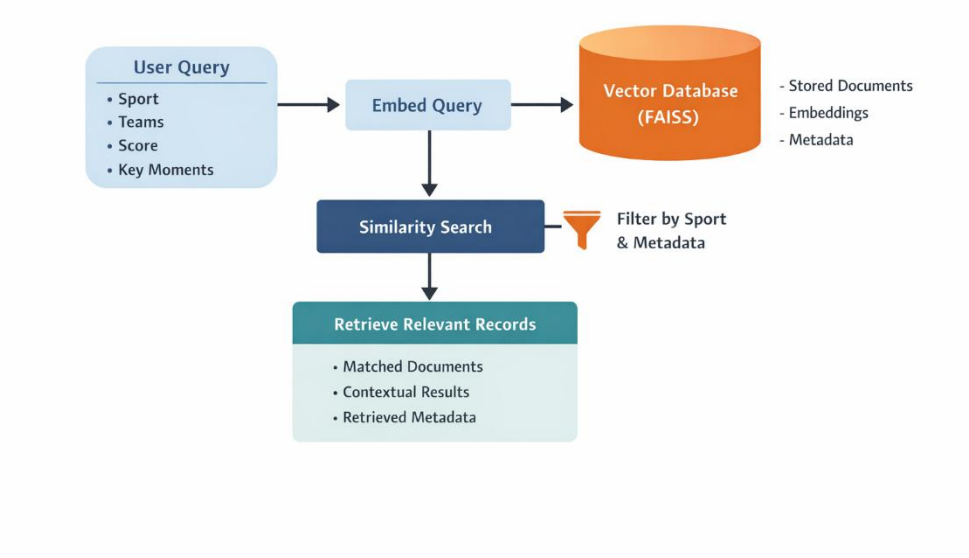


3. Data Design

3.1 Data Model



3.2 Data Access Mechanism

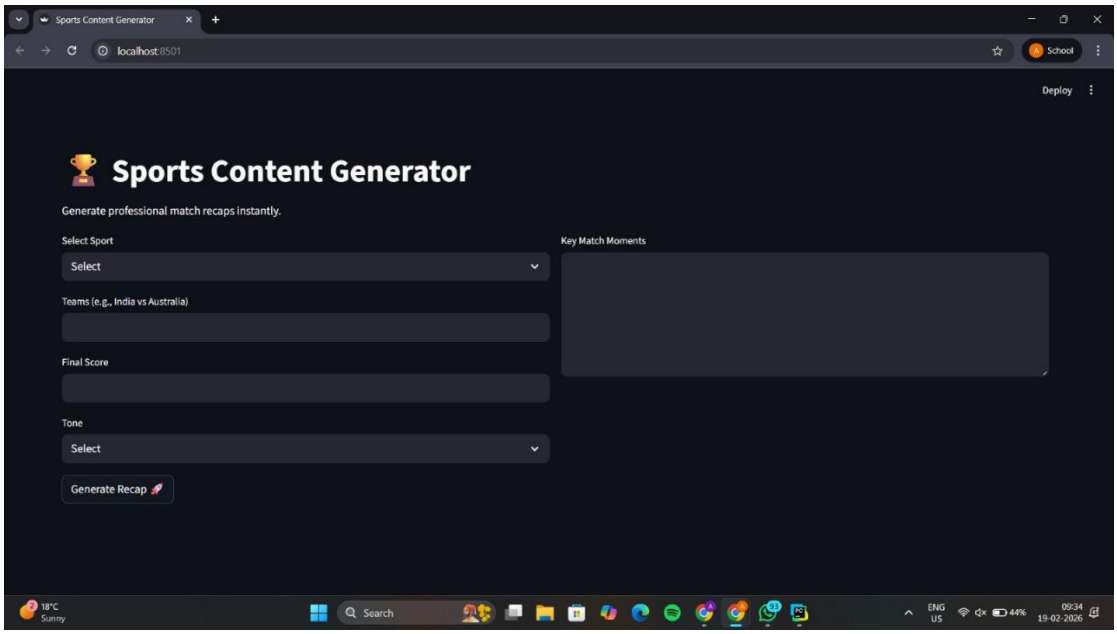


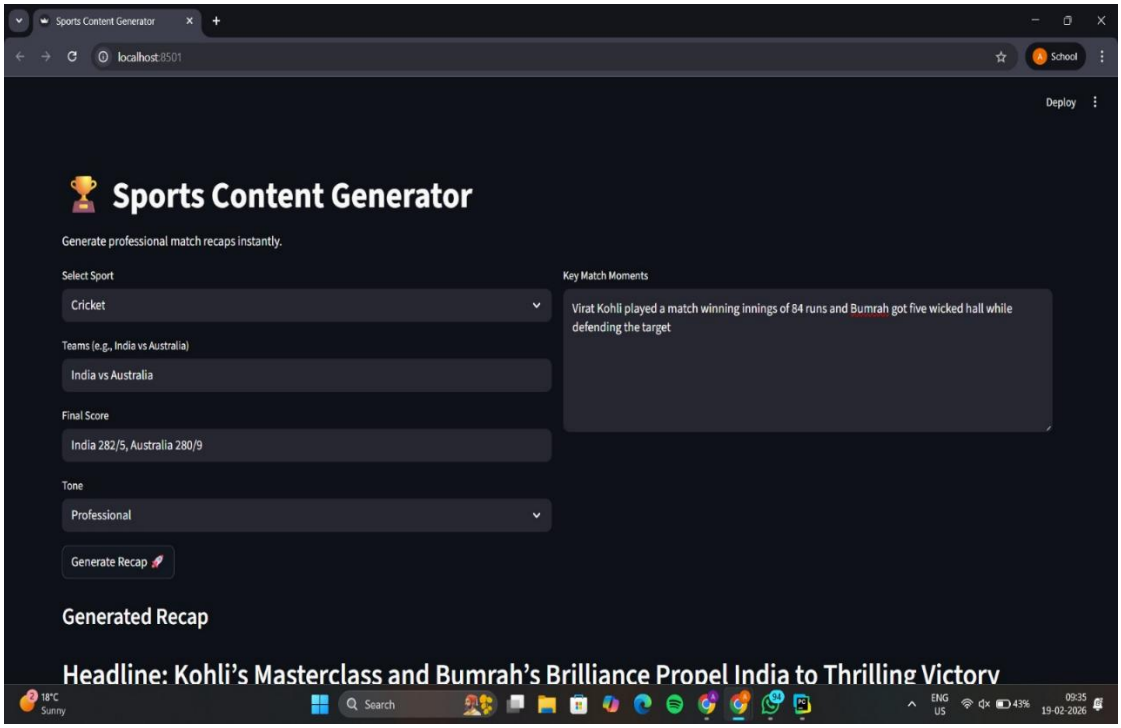
4. Interfaces

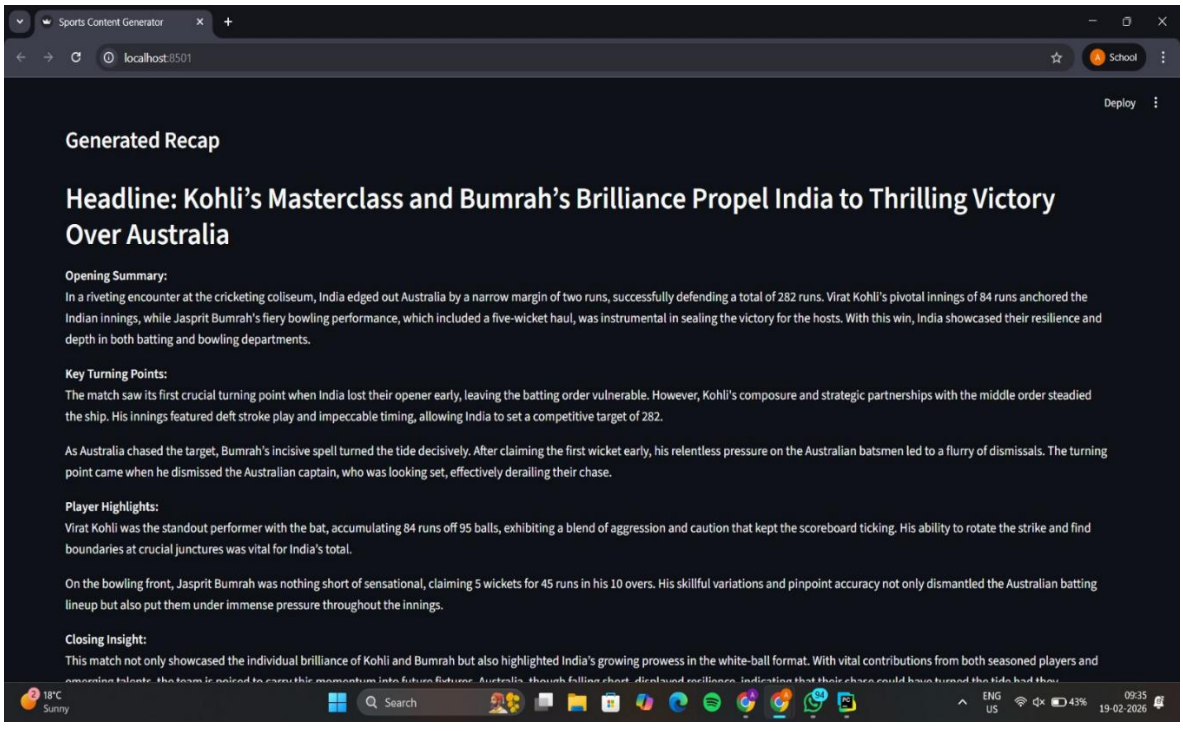
This section describes the interfaces between different system components including the **User Interface, LLM services, and Content Generation services**.

Types of Interfaces:

- User Interface (Web-based Frontend)
- Content Generation Interface




Sports Content Generator
localhost:8501
School
Deploy
Sports Content Generator
Generate professional match recaps instantly.
Select Sport
Cricket
Teams (e.g., India vs Australia)
India vs Australia
Final Score
India 282/5, Australia 280/9
Tone
Professional
Generate Recap
Key Match Moments
Virat Kohli played a match winning innings of 84 runs and Bumrah got five wicked hall while defending the target
Generated Recap
Headline: Kohli's Masterclass and Bumrah's Brilliance Propel India to Thrilling Victory


Sports Content Generator
localhost:8501
School
Deploy
Generated Recap
Headline: Kohli's Masterclass and Bumrah's Brilliance Propel India to Thrilling Victory Over Australia
Opening Summary:
In a riveting encounter at the cricketing coliseum, India edged out Australia by a narrow margin of two runs, successfully defending a total of 282 runs. Virat Kohli's pivotal innings of 84 runs anchored the Indian innings, while Jasprit Bumrah's fiery bowling performance, which included a five-wicket haul, was instrumental in sealing the victory for the hosts. With this win, India showcased their resilience and depth in both batting and bowling departments.
Key Turning Points:
The match saw its first crucial turning point when India lost their opener early, leaving the batting order vulnerable. However, Kohli's composure and strategic partnerships with the middle order steadied the ship. His innings featured deft stroke play and impeccable timing, allowing India to set a competitive target of 282.
As Australia chased the target, Bumrah's incisive spell turned the tide decisively. After claiming the first wicket early, his relentless pressure on the Australian batsmen led to a flurry of dismissals. The turning point came when he dismissed the Australian captain, who was looking set, effectively derailing their chase.
Player Highlights:
Virat Kohli was the standout performer with the bat, accumulating 84 runs off 95 balls, exhibiting a blend of aggression and caution that kept the scoreboard ticking. His ability to rotate the strike and find boundaries at crucial junctures was vital for India's total.
On the bowling front, Jasprit Bumrah was nothing short of sensational, claiming 5 wickets for 45 runs in his 10 overs. His skillful variations and pinpoint accuracy not only dismantled the Australian batting lineup but also put them under immense pressure throughout the innings.
Closing Insight:
This match not only showcased the individual brilliance of Kohli and Bumrah but also highlighted India's growing prowess in the white-ball format. With vital contributions from both seasoned players and

5. State and Session Management

The Sports Content Generator manages user interaction using lightweight session handling to maintain request continuity during content generation.

- Each user request is treated as a session while generating sports content.
- Backend APIs temporarily store input data such as match details, sport type, and generation parameters.
- Session state helps track user prompts and generated responses during processing.
- No long-term personal data is stored; only generated content metadata may be saved.
- Vector database maintains contextual embeddings instead of user sessions for intelligent retrieval.

6. Caching & Future Optimizations

To improve performance and reduce response time, caching mechanisms are used in the system:

- Frequently generated sports content is temporarily stored in cache.
- Similar user queries can reuse previously generated results instead of calling the AI model again.
- Embeddings stored in **vector database** avoid recomputation during semantic search.
- API responses can be cached to reduce external API calls and cost.

Benefits:

- Faster content generation
- Reduced API usage cost
- Improved scalability
- Better user experience

7. Deployment Architecture

Service	Platform	Details
Frontend	Streamlit.io	Auto-deploys on git push to main; preview URL per PR; SPA fallback via vercel.json
Backend	Railway/Render	Python 3.11 auto-detected; start command via Procfile; root directory:

		/backend
Vector Database	ChromaDB Cloud	Managed cloud instance; accessed via API key, tenant, and database credentials
LLM + Embeddings	Open AI API	openembedding-001 (embeddings)
CI/CD	GitHub	Both Railway/Render and Streamlit.io auto-deploy on every push to main branch

Environment Variables

Sensitive credentials are never stored in the repository. All secrets are configured as environment variables on Railway/Render (backend) and Vercel (frontend). The Firebase service account JSON is passed as a single-line string in OPEN_API_KEY.

8. Non-Functional Requirements

Category	Requirement
Performance	System should generate sports content within 2–5 seconds for normal requests.
Scalability	Backend should support multiple users simultaneously using API-based architecture.
Availability	Application should be accessible 24/7 with minimal downtime.
Security	API keys and user inputs must be protected using environment variables and secure requests.
Reliability	System should handle API failures and invalid inputs gracefully without crashing.
Usability	Simple and intuitive UI for easy content generation by non-technical users.

Category	Requirement
Maintainability	Modular backend structure (services, API routes, vector DB layer) for easy updates.
Compatibility	Works across modern browsers and devices (desktop & mobile).
DataPrivacy	User prompts and generated content should not be permanently stored unless required.
Extensibility	New sports, datasets, or AI models can be added without redesigning the system.

9. Known Limitations & Future Enhancements

Known Limitations

- Generated content quality depends on available sports data.
- Limited real-time match updates (uses stored or sample datasets).
- API rate limits may affect content generation speed.
- Vector database performance may reduce with very large datasets.
- Currently supports limited sports categories.

Future Enhancements

- Integration with real-time sports APIs for live match data.
- Multi-language content generation support.
- Advanced analytics dashboard for performance insights.
- Improved semantic search using optimized embeddings.
- User authentication and personalized content history.
- Cloud deployment for scalability and faster response.

10. References

- OpenAI. OpenAI API Documentation. https://platform.openai.com/docs
- Lewis et al., 2020. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks.
- Facebook AI Research. FAISS Similarity Search Library. https://github.com/facebookresearch/faiss
- Reimers & Gurevych, 2019. Sentence-BERT: Sentence Embeddings using Siamese Networks. https://arxiv.org/abs/1908.10084
- FastAPI Documentation. https://fastapi.tiangolo.com/
- Streamlit Documentation. https://docs.streamlit.io/
- Bommasani et al., 2021. On the Opportunities and Risks of Foundation Models. https://arxiv.org/abs/2108.07258