NEW SUMMITCOLLEGE

Affiliated To

Tribhuvan University

Institute of Science and Technology

A Final Year Project Report On

“Game Plan”

In partial fulfillment of the requirement of the completion of seventh semester project of Bachelor Degree in Computer Science and Information Technology

Submitted to

Department of Computer Science and Information Technology

New Summit College

Under The Supervision of

Rishav Acharya

Submitted By:

Abhishek Khadka [25911/077]

Kaustub Karki [25927/077]

Naman Bikram Karki [25934/077]

February, 2025

Acknowledgement

We are delighted to have the chance to express our sincere appreciation to everyone who has supported us during this initiative. Our supervisor, **Mr. Rishav Acharya**, has our sincere gratitude for his knowledgeable direction, unwavering support, and readiness to take time out of his often, hectic schedule to monitor the project's progress. His unceasing motivation has enabled us to finish this project and meet its goal.

We would also like to express our deepest appreciation to **Mr. Chok Raj Dawadi**, Principal, New Summit College, for his constant motivation, support and for providing us with a suitable working environment.

We sincerely acknowledge direct and indirect help, suggestions and feedback offered by our colleagues before, during and after the development and implementation of the project.

At last, our special thanks go to all staff members of the CSIT department at New Summit College who kindly extended their hands in making this project work a success.

Your Sincerely,

Abhishek Khadka [25911/077]

Kaustub Karki [25927/077]

Naman Bikram Karki [25934/077]

Abstract

GamePlan is an advanced football video analysis and tracking system, designed to deliver precise, automated insights and interactive analytics for sports enthusiasts and professionals. The system utilizes a Python-based backend integrated with advanced machine learning models, such as YOLO, Byte Track and Optical flow for object detection and tracking, alongside K-Means clustering for team assignment. The Lucas-Kanade algorithm is employed for accurate camera movement estimation, enhancing the precision of tracking data. Additionally, perspective transformation and real-world data mapping further refine the analysis of player movements, distance and speed. The backend is complemented by an intuitive frontend built with HTML, CSS, and JavaScript, providing seamless video uploads, analytics visualization, and user interaction. Key functionalities include player tracking, ball ownership detection, speed and distance estimation, comprehensive video annotation, which provides personalized and real-time analytics. By leveraging automation, robustness, and user-friendly interfaces, this project redefines traditional sports analytics and tracking. The Football Video Analysis System is particularly focused on transforming football analytics in the sports community, offering a dynamic platform that promotes collaboration and data-driven insights. Through this intelligent system, the project aspires to revolutionize sports analysis by delivering efficient, accurate, and interactive solutions.

Keywords: *Football Analytics, Object Tracking, Machine Learning, Automation, Lucas-Kanade Algorithm, Real-Time Analysis, Robustness*

List of Abbreviations

AI	Artificial Intelligence
API	Application Programming Interface
Byte Track	Bounding Box Track
CV	Computer Vision
FPS	Frames Per Second
K-Means	K-Means Clustering Algorithm
ML	Machine Learning
Node.js	JavaScript Runtime Environment
OpenCV	Open-Source Computer Vision Library
RGB	Red Green Blue
RTAs	Real-Time Applications
SIGLIP	Sigmoid Loss for Language Image Pre-Training
UMAP	Uniform Manifold Approximation and Projection
VS Code	Visual Studio Code
YOLO	You Only Look Once

Table of Contents

List of Figures

List of Tables

Chapter 1: Introduction

1.1 Introduction

Football is a very fast paced sports where a single second count and each player can have a huge impact on the game. An effective football analytics system is essential a football organization to ensure accurate and efficient tracking of player performance, team dynamics, and game strategies. Recent advancements in technology have revolutionized the methods used for football analysis, offering innovative tools for tracking and recording insights. One such innovation is the integration of intelligent video analytics powered by computer vision and machine learning.

With these advancements, football analytics can now be seamlessly controlled and managed in a variety of environments, including professional sports arenas, educational institutions, and training facilities. The system employs cutting-edge computer technologies, such as YOLO [1] for object detection and the Lucas-Kanade [2] algorithm for camera movement estimation, to deliver precise and real-time insights. Players, referees, and ball trajectories are tracked using robust tracking algorithms like Byte Track [3], ensuring accuracy and reliability.

By combining camera systems with advanced machine learning models, this project offers a comprehensive solution for football analytics. The system captures gameplay from an eagle-eye perspective, processes video data with high precision, and provides detailed insights into player speed, distance, team dynamics, and ball ownership. This intelligent integration of vision technology and machine learning establishes a simple yet powerful method for football analytics, ensuring accuracy, efficiency, and reliability in performance tracking.

1.2 Problem Statement

Traditional football analytics rely heavily on manual observation and labor-intensive processes, leading to inefficiencies, inaccuracies, and inconsistent data. Coaches and analysts often struggle to track crucial metrics such as player movements, ball trajectories, and team strategies, primarily due to the lack of automated systems. This reliance on manual methods not only consumes significant time and resources but also introduces the risk of human error, undermining the reliability of insights crucial for decision-making. As the demand for actionable and real-time insights in sports analytics continues to grow, the limitations of existing solutions become evident. Many current systems fail to provide the robustness required to process real-time video streams or come with prohibitive costs, making them inaccessible to smaller organizations or grassroots initiatives. Additionally,

these solutions often lack seamless integration between video analysis and interactive analytics platforms, resulting in a fragmented workflow that hampers usability for teams, coaches, and analysts. Addressing these gaps necessitates the development of an intelligent, cost-effective, and fully automated football analytics system that ensures accuracy, efficiency, and accessibility.

1.3 Objectives

The main objectives of this project are:

- To automate the upload, detection, and processing of football match videos using advanced computer vision and machine learning techniques.
- To enable real-time tracking of players, ball, and referees with accurate insights into speed, distance, and team dynamics
- To ensure efficient and reliable data analysis through the integration of modern algorithms like K-Means, YOLO, Byte Track, and Lucas-Kanade.

1.4 Scope and Limitations

1.4.1 Scope

a. Advanced Football Analytics:

- The system provides real-time tracking of players, referees, and the ball during football matches using machine learning models such as YOLO and Byte Track.
- Offers insights into player speed, distance covered, team dynamics, and ball possession.

b. Seamless Integration:

- Combines video upload, processing, and result visualization into a single, user-friendly platform.
- Supports video uploads through a React.js [4] frontend, with processing handled by a Python-based backend using Fast API [5].

c. Perspective Transformation:

- Converts video coordinates into real-world dimensions for accurate measurement of player metrics.

d. Automation and Efficiency:

- Automates tasks like object detection, tracking, and analytics generation, reducing manual intervention and potential errors.

e. Accessibility:

- Designed to be scalable and affordable, making it suitable for sports organizations, educational institutions, and grassroots initiatives.

f. Interactive Outputs:

- Annotates processed videos with metrics such as speed, distance, and team ball control, offering an engaging and informative experience.

1.4.2 Limitations

- Accuracy Challenges: Environmental factors like lighting conditions, camera angles, and changes in appearance (e.g., clothes changes by mud or snow) can affect the accuracy of football analysis.
- Limited Computation: The effectiveness of the system may be limited if the videos are lengthy and may require higher performing chipset of CPUs or GPUs.
- Camera constraint: The system only works with a fixed camera angle as the model has been trained in such a way.

1.5 Development Methodology

Agile Model:

Figure 1.1: Agile Model *[17]*

a. Requirements: The Agile process begins with gathering requirements in the form of user stories or use cases. For the Football Analytics System, we identified critical needs like player detection, ball tracking, and performance analytics by collaborating with stakeholders.

b. Design: In this phase, modular and adaptable system architecture was created. We designed individual components for object detection, tracking, clustering, and field transformation, ensuring they could integrate seamlessly into the overall system.

c. Development: Development was carried out in short iterations (sprints). For example, we implemented YOLO for player detection, Byte Track for tracking, and K-Means for clustering analytics. The Python programming language, along with libraries such as OpenCV [6], and Ultralytics [7], was used extensively during this phase.
d. Testing: Testing was continuous and iterative. Each feature, such as player detection or camera movement estimation, was validated through unit and integration testing to ensure functionality and reliability after each sprint.
e. Deployment: Features were deployed incrementally, allowing users to interact with the system early and provide feedback. For instance, the speed and distance measurement module were deployed after completing the real-world transformation feature.
f. Review: At the end of every sprint, stakeholders reviewed the delivered features. Their feedback helped refine functionalities, adjust priorities, and plan for the next sprint, ensuring the system met user expectations.

1.6 Report Organization

This report is organized into five chapters:

Chapter 1: "Introduction"- This chapter introduces the problem statement, objectives and limitations of the project.

Chapter 2: "Requirement and Feasibility Analysis"- This chapter describes the functional and non-functional requirements, economic feasibility, technical feasibility, operational feasibility and scheduling feasibility.

Chapter 3: "System Design"- This chapter introduces the system and interface design of the project app.

Chapter 4: "Implementation and Testing"- This chapter clearly illustrates the methods and tools used to implement the project.

Chapter 5: "Conclusion and Future Works"- This is the final chapter that concludes the project and talks about our future with the project.

Chapter 2: Background Study and Literature Review

2.1 Background Study

Football analytics has undergone a transformative journey, evolving from simple manual observations to the integration of cutting-edge technologies such as machine learning and computer vision. Today, analytics systems provide actionable insights into player performance, team strategies, and match dynamics. However, these advancements are often inaccessible to smaller organizations due to high costs, steep learning curves, and complex implementation processes.

With innovations in video processing and AI, modern football analytics systems have introduced features like automated player tracking, ball trajectory prediction, and heatmap generation. These technologies enable in-depth analysis, helping teams optimize their strategies and improve performance metrics. Despite this, many current systems fail to address practical usability concerns, particularly in terms of user-centric design and real-time data visualization.

Our project aims to fill these gaps by developing a comprehensive, cost-effective football analytics platform. Leveraging machine learning algorithms such as YOLO for object detection, Byte Track for player tracking, and K-Means clustering for data analytics, this system combines robust back-end processing with a user-friendly web-based interface. By incorporating real-time feedback and seamless integration, the platform ensures accessibility for users of varying technical expertise, from amateur leagues to professional teams.

While challenges such as environmental variability and camera calibration persist, the proposed system adopts adaptive techniques to improve object recognition and tracking accuracy. Beyond efficiency, the system enhances decision-making, optimizes infrastructure management, and boosts productivity at all levels of the game. This project aims to democratize football analytics with accessible technology and innovative design, empowering all stakeholders with precise, efficient, and strategic insights.

2.2 Literature Review

Football analytics systems widely use YOLO (You Only Look Once) models for real-time object detection due to their speed and accuracy. Studies such as Bochkovskiy [8] demonstrated the effectiveness of YOLOv4 in detecting players, referees, and the ball, even in challenging scenarios involving occlusions and fast movements. Projects like the one by Badrinarayanan [9] leveraged YOLOv3 [10] for football analytics but faced limitations in detection precision, especially for smaller objects like the ball. By contrast, this project employs YOLOv8, a more advanced version that offers better accuracy and speed, making it well-suited for real-time applications.

Tracking algorithms are integral to maintaining the continuity of detected objects across video frames. While previous projects have used DeepSORT [11] and Hungarian algorithms for multi-object tracking, their computational demands make them less efficient for real-time processing. ByteTrack [12], introduced by Zhang et al. (2021), has emerged as a lightweight alternative, capable of associating high-confidence detections while maintaining robustness against false positives. This project adopts ByteTrack to ensure smooth and accurate tracking of players and the ball, optimizing performance for real-time football analytics.

Accurate mapping of video frames to real-world dimensions is essential for calculating metrics like player speed and distance. Szeliski [13] (2010) highlighted the significance of homograph-based transformations in sports analytics, allowing for precise spatial analysis even with camera distortions. Building on this foundation, the proposed system uses perspective transformation combined with Lucas-Kanade [14] optical flow to stabilize camera movements. This approach enhances the accuracy of player tracking and metric calculations, ensuring reliable insights for football analytics.

Football Analytics using Computer Vision: Developed by Oneture Technologies [15], showcases a project that leverages on deep learning algorithm with use of computer vision which helps to generate statistics and prediction from football match videos. The given system provides components for object detection, multi-object tracking, player identification team differentiation, and event identification, to replace manual tagging in football analytics.

Chapter 3: System Analysis

3.1 System Analysis

This Football Analytics System aims to modernize traditional methods of football analysis by introducing an innovative and user-friendly platform powered by advanced video processing and machine learning technologies. This system eliminates the reliance on manual observation and data collection by automating tasks such as player tracking, ball detection, and tactical analysis. It provides administrators and coaches with real-time insights into player performance and game patterns, significantly reducing the effort and complexity associated with traditional approaches. By streamlining the football analytics process, this system ensures an efficient and accessible solution for both professional organizations and amateur teams, enhancing decision-making and improving the overall experience for users.

3.1.1 Requirement Analysis:

a) Functional:

- The system should have accurate object detection such as whether a person is a player from team A or team B, referee or goalkeeper and not recognize the audience.
- The system should perform automated Football Analytics recording for recognized teams.
- The users should be able to upload the video. The final annotated video should be available to download or stream for user

b) Use Case Diagram

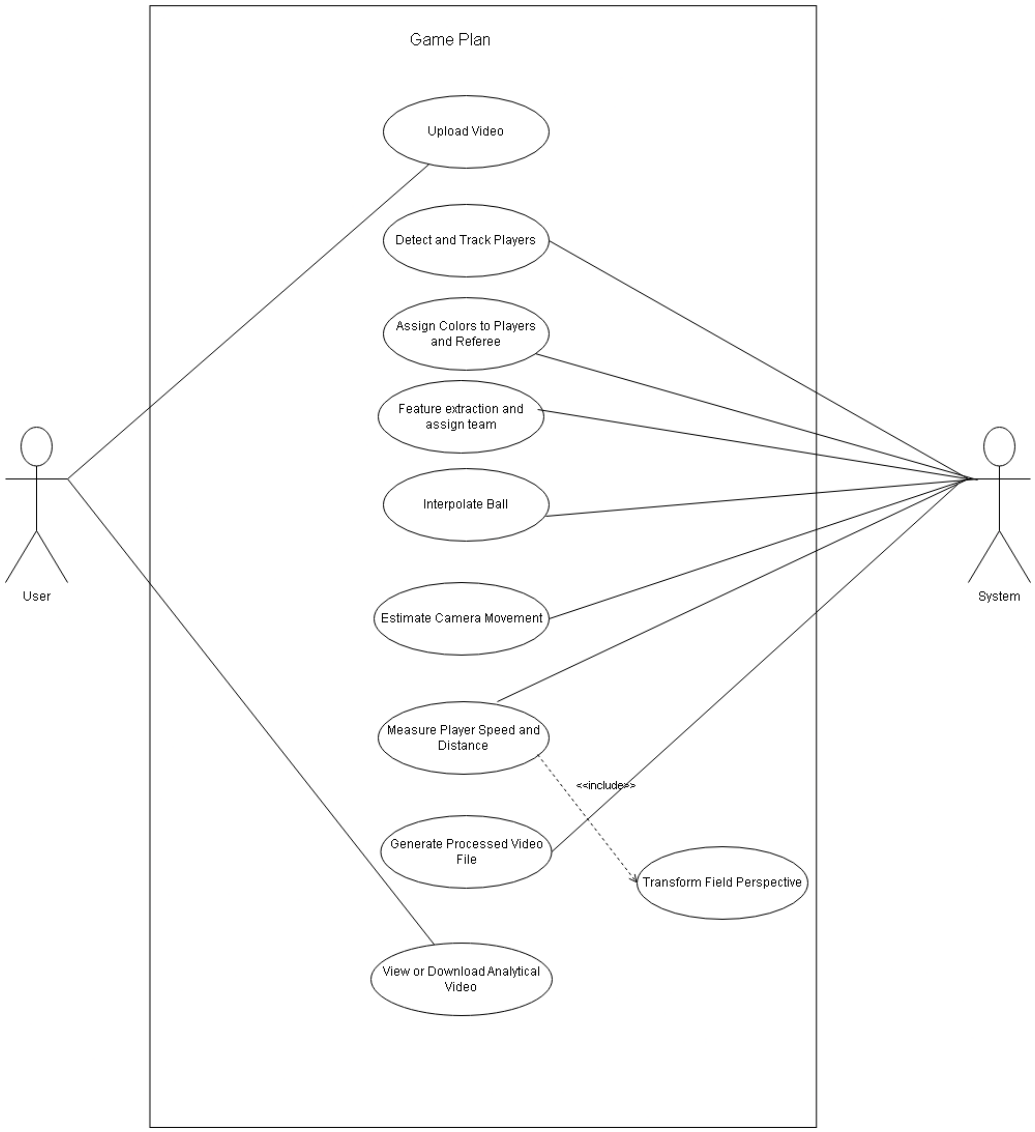


Figure 2.1: Use Case diagram

c) Non-Functional:

- The system should be easy to maintain and update, with a clear and well documented codebase.
- The system should be easy to use and understand for every user.
- The system should process a video within 5-10 minutes.

3.1.2 Feasibility Analysis:

A feasibility study evaluates a project's likelihood of success by analyzing various factors, including technical, operational, financial, and legal aspects. This assessment ensures the project's feasibility under specific constraints such as technical readiness, resource availability, and cost compliance.

- **Technical Feasibility:** Technical feasibility examines the availability of software, hardware, and expertise to execute the project. The Football Analytics System operates on computers with Python and Node.js [16] installed, leveraging their robust libraries and frameworks for efficient development. Python ensures modern standards are met, while Node.js supports the execution of JavaScript programs, confirming the technical readiness of the system.
- **Operational Feasibility:** Operational feasibility assesses whether the system benefits users and predicts its adoption. With the increasing familiarity of players, referees, and administrators with digital tools, this project aligns with user expectations, ensuring ease of use and broad acceptance. Thus, the system is operationally feasible.
- **Economic Feasibility:** Economic feasibility evaluates the balance between the system's costs and anticipated benefits. The analysis includes the cost of developing the system and acquiring necessary equipment, such as cameras. The projected benefits, including enhanced efficiency and detailed analytics, outweigh the development and operational costs, confirming economic feasibility.
- **Legal Feasibility:** The system complies with existing legal frameworks, with no restrictions on using cameras for football analytics. It adheres to the regulations of the region, ensuring the project is legally feasible.
- **Schedule Feasibility:** The time required to complete the project, and the time spent on each activity in the following weeks are represented in the Gantt chart below.

Gant Chart:

Table 1: Gantt Chart

Project Plan	Assigned to	Progress	TIMELINE
			August (in weeks) 1 2 3 4 | September (in weeks) 1 2 3 4 | October (in weeks) 1 2 3 4 | November (weeks) 1 2 3 4 | December (in weeks) 1 2 3 4
Sprint 1			
Requirement Gathering	Abhishek Khadka	100%	
Set Up Development Environment	Naman Karki, Kaustub Karki	100%	
Model Training	Naman Karki	100%	
Sprint 2			
Object Tracking	Kaustub Karki	100%	
Test object tracking	Abhishek Khadka	100%	
Sprint 3			
Develop team color assignment and ball ownership logic	Abhishek Khadka	100%	
Assign players to teams	Naman Karki	100%	
Assign ball ownership	Kaustub Karki		
Sprint 4			
Camera Movement Estimation	Naman Karki	100%	
Perspective Transformation	Naman Karki	100%	
Speed and Distance Estimation	Abhishek Khadka		
Sprint 5			
Frontend Design and Development	Kaustub Karki	100%	
Integration of frontend and backend	Kaustub Karki	100%	
Testing	Naman Karki, Kaustub Karki	100%	
Documentation	Abhishek Khadka	100%	

Chapter 4: System Design

4.1 Design

4.1.1 Flowchart:

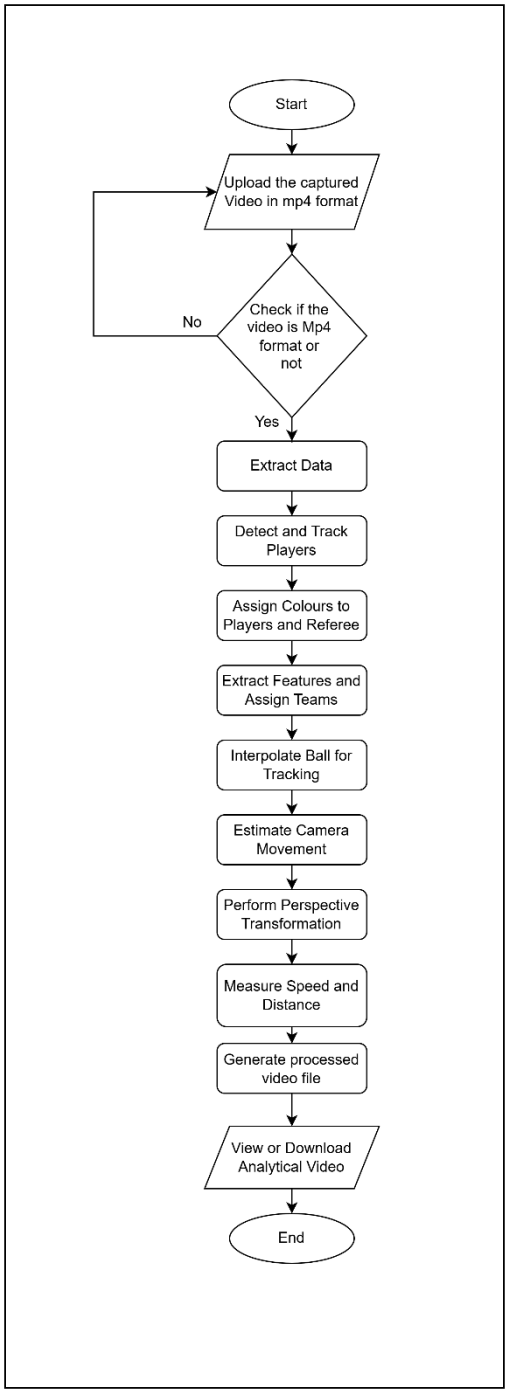


Figure 4.1: System Flowchart

4.1.2 Sequence Diagram:

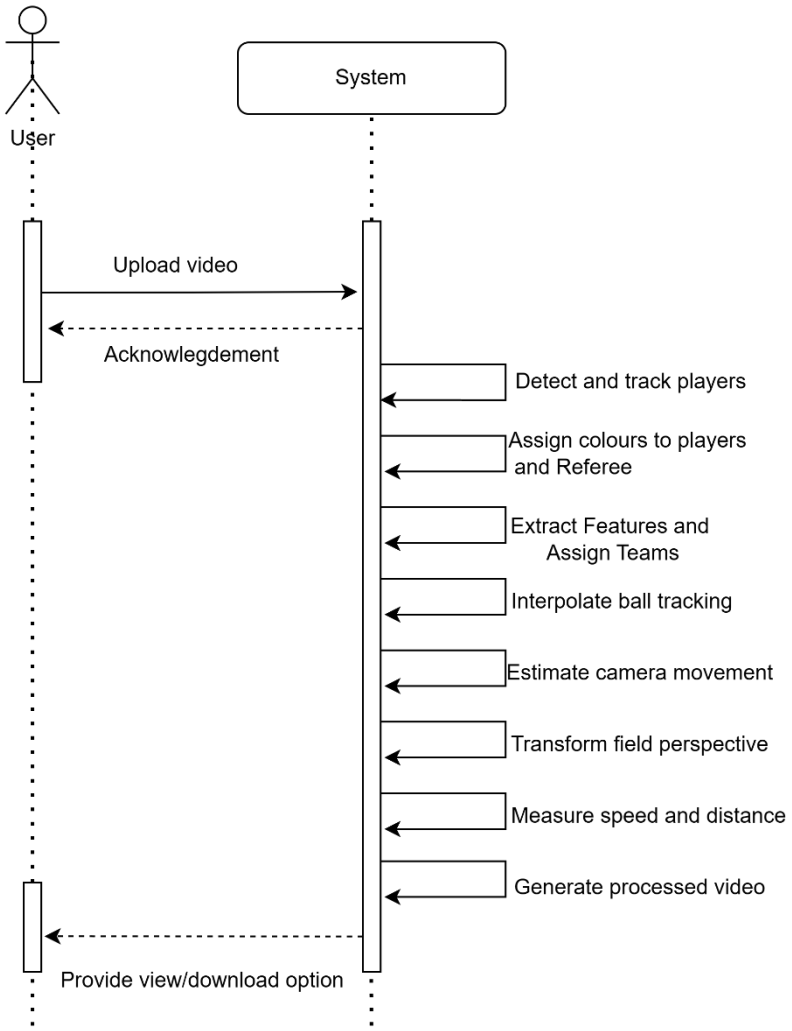


Figure 3.2: Sequence Diagram

4.1.3 Activity Diagram:

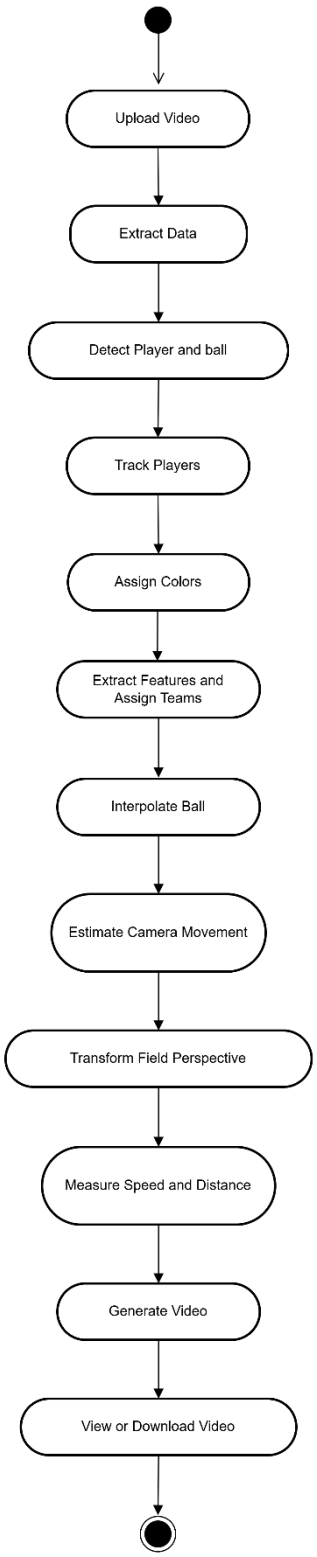


Figure 4.3: Activity Diagram

4.1.4 Class Diagram

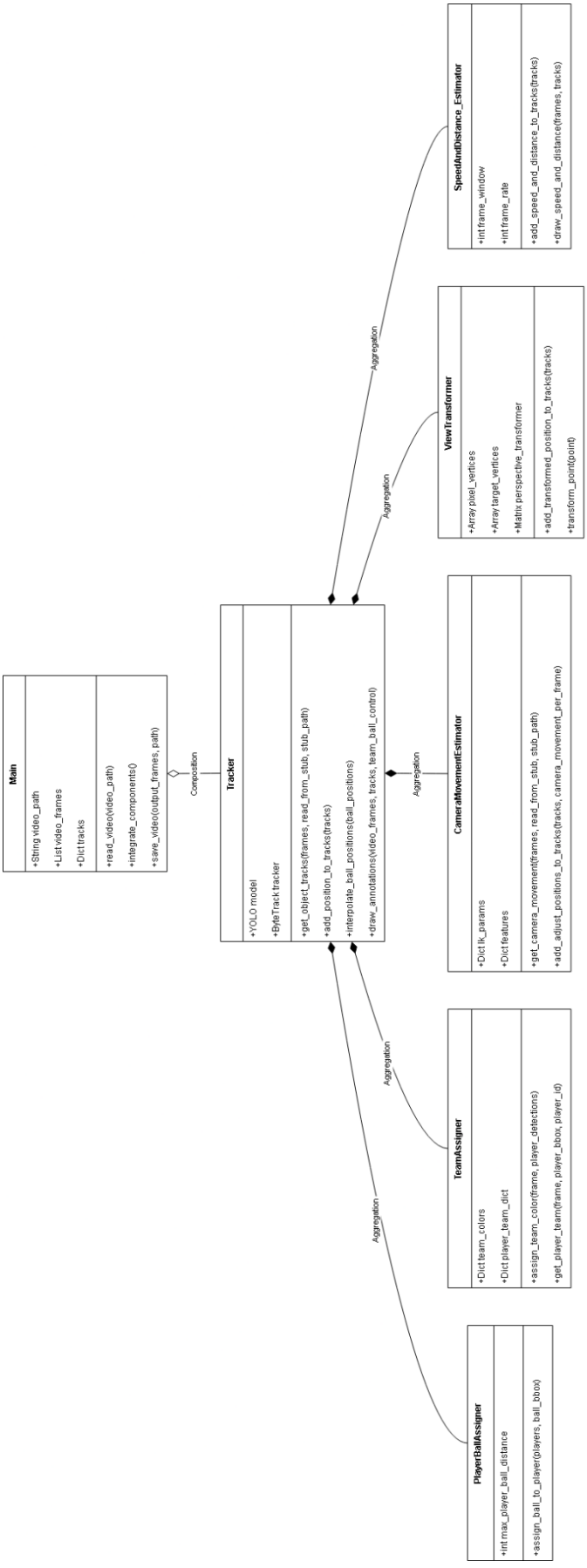


Figure 4.4: Class Diagram

Algorithm Used

The Football Analytics System utilizes advanced algorithms to process video data from football matches, providing detailed insights into player movements and game dynamics. The system integrates detection, tracking, and motion estimation techniques to deliver high accuracy and performance.

YOLO (You Only Look Once): YOLO is a real-time object detection algorithm employed to identify players, referees, and the ball in video frames. It processes entire frames in a single pass, ensuring efficient and accurate detection, even in dynamic football scenes.

UMAP (Uniform Manifold Approximation and Projection): UMAP is an algorithm which is used for reduction of dimensionality. It is widely used in machine learning and data visualization to reduce high dimensional data into lower dimensional data (for e.g.: imbedding space of deep learning models).

Byte Track Algorithm: Byte Track is a powerful multi-object tracking algorithm that maintains consistent identities of detected objects across frames. By combining high-confidence and low-confidence detections, Byte Track effectively handles occlusions, rapid movements, and complex interactions in football matches.

Lucas-Kanade Algorithm: The Lucas-Kanade optical flow algorithm is used to estimate camera movements and relative object motion. It calculates motion vectors between consecutive frames, which is essential for aligning the video view and stabilizing the data for further processing. This step ensures the system adapts to variable camera dynamics during matches.

K-Means Clustering: K-Means clustering is used to group players based on visual characteristics like jersey color or movement patterns. This facilitates the differentiation of teams and identification of referees, enabling a clear understanding of the game's structure.

Working Mechanism:

- Object Detection: YOLO efficiently detects players, referees, and the ball within video frames.
- Tracking: Byte Track ensures consistent tracking of detected entities across frames, handling occlusions and rapid movements seamlessly.
- Motion Estimation: The Lucas-Kanade algorithm estimates camera shifts and stabilizes the field view to adapt to changing perspectives.
- Reduced dimensionality: UMAP is used to represent complex motion data in a lower dimensional space.

- Role Assignment: K-Means clustering classifies players into teams and identifies referees, streamlining the analysis of game dynamics.

4.2 K-Means Clustering Algorithm

The Football Analytics System employs a custom implementation of the K-Means clustering algorithm to assign players to teams based on their jersey colors. This method ensures accurate and efficient team classification by analyzing visual data from player bounding boxes within video frames. The algorithm's workflow is tailored for handling real-time scenarios in football matches.

4.3 Workflow

a. Initialization:

- The Custom KMeans model is initialized with:
  - n_clusters=2: Two clusters representing the two teams.
  - max_iters=100: Maximum number of iterations for convergence.
  - random_state=42: Ensures reproducibility.

b. Feature Extraction & Preprocessing:

- Players' feature vectors are extracted using SiglipVisionModel.
- If stored in player_feature_cache, existing features are reused.
- UMAP reduces feature dimensions from high-dimensional space to 3D.

c. Clustering Process:

- The CustomKMeans.fit() method is used to:
  - Select two random points as initial cluster centroids.
- Iterate up to 100 times or until centroids no longer change:
  - Compute the Euclidean distance between each player's feature vector and centroids. Assign each player to the nearest cluster.
  - Compute new centroids as the mean of assigned players features.
  - Repeat until centroids stabilize.

d. Team Assignment:

- Each player's track ID is mapped to the assigned cluster (0 or 1) in player_team_mapping.
- If a player was not in previous frames, their team is determined by the nearest centroid using CustomKMeans.predict().
- If teams become unbalanced, the system assigns new players to the team with fewer members.

e. Edge Case Handling:

- For any given player, their jersey color is compared to the team color centroids using the custom K-Means predict method.
- If fewer than 2 players are detected, clustering is skipped. If a player's jersey color changes over time, reassign=True forces re-clustering. If the team assignment file exists, previous assignments are loaded to maintain consistency.

4.4 Working of K-Means Clustering Algorithm

Step 1: Initialization:

- Select the number of cluster k.
- Centroids are initialized using K-Means++, which selects the first centroid randomly and the remaining centroids based on data distribution to improve clustering accuracy.

Step 2: Assignment Step:

- For each data point xi, assign it to the nearest cluster Cj based on the Euclidean distance.

Step 3: Update Step:

- Recalculate the centroid (μj) for each cluster Cj by computing the mean of all points in the cluster.

Step 4: Convergence Check:

- Repeat Steps 2 and 3 until either:
  - The centroid (μj) does not change significantly, or
  - A maximum number of iterations is reached.

Step 5: Output:

- Return final cluster assignments (C1, C2, …., Ck) and centroids (μ1, μ2, …., μk).

4.5 Numerical Example

The application of K-Means clustering for player team assignment based on extracted feature vectors. The dataset consists of 6 players, each represented in a 3D feature space. We use the K-Means++ initialization method to ensure efficient centroid selection.

Step 1: Initial Data and Centroids

The following table presents the initial feature vectors for each player:

Initial Player Feature Vectors:

Player ID	Feature 1	Feature 2	Feature 3
1	1.5	2.0	3.2
2	2.1	1.8	3.0
3	4.5	5.0	4.8
4	4.8	5.3	4.6
5	1.3	2.2	3.1
6	4.7	5.1	4.9

Using the K-Means++ initialized method, the two initial centroids are selected as:

Initial Centroids from K-Means++

Centroid	Feature 1	Feature 2	Feature 3
Centroid 1	4.67	5.13	4.77
Centroid 2	1.63	2.00	3.10

Step 2: Euclidean Distance Calculations

Each player is assigned to the nearest centroid based on the Euclidean distance formula:

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2}$$

Step 3: Distance Calculations

Example 1: Distance from Player 3 to Centroids

$$d_{3,C1} = \sqrt{(4.5 - 4.67)^2 + (5.0 - 5.13)^2 + (4.8 - 4.77)^2} = 0.22$$

$$d_{3,C2} = \sqrt{(4.5 - 1.63)^2 + (5.0 - 2.00)^2 + (4.8 - 3.10)^2} = 4.48$$

Since Player 3 is closer to Centroid 1, they are assigned to Team 1

Example 2: Distance from Player 1 to Centroids

$$d_{1,C1} = \sqrt{(1.5 - 4.67)^2 + (2.0 - 5.13)^2 + (3.2 - 4.77)^2} = 4.72$$

$$d_{1,C2} = \sqrt{(1.5 - 1.63)^2 + (2.0 - 2.00)^2 + (3.2 - 3.10)^2} = 0.17$$

Since Player 1 is closer to Centroid 2, they are assigned to Team 2.

Step 4: Final Team Assignments

The final team assignments based on K-Means clustering are as follows:

Final Player Team Assignments:

Player ID	Assigned Team
1	Team 2
2	Team 2
3	Team 1
4	Team 1
5	Team 2
6	Team 1

Step 5: Conclusion

This process ensures that each player is assigned to the correct team based in extracted with feature similarities. By using K-means++, we avoid poor initialization, leading to more stable clustering results.

Chapter 5: Implementing and Testing

5.1 Implementation

Software implementation is the process of converting the designed system into the programs. This process includes not only the actual writing of the code but also the preparation of the requirements and objectives, the design of what is to be coded, and confirmations that what is developed has met the predefined objectives.

5.1.1 Tools Used:

- **Python**: Python serves as the backbone of the project, offering a flexible and user-friendly programming environment. Its rich library ecosystem, including tools like Byte Track, OpenCV for object detection, and YOLO, ensures efficient development and smooth integration of various project components.
- **Fast API**: Fast API is a modern web framework for Python, designed to simplify the creation of high-performance APIs. It stands out for its speed, support for type hinting, automatic documentation generation, and asynchronous capabilities, making it a powerful choice for API development.
- **VS Code**: Visual Studio Code is a lightweight and versatile code editor by Microsoft. It provides robust features for writing, editing, and debugging code, and makes it easy to publish applications. Its advanced debugging tools and intuitive interface make it a favorite among developers.
- **Draw.io**: Draw.io is a free, browser-based diagramming tool that's perfect for creating class diagrams, sequence diagrams, component diagrams, and deployment diagrams. As an open-source solution, it's widely used for visualizing project architectures and workflows.
- **Node.js**: Node.js is a popular platform for building real-time applications. Its asynchronous, event-driven architecture excels at handling intensive input-output tasks, making it ideal for creating modern, high-performance apps that meet user expectations.

5.2 Testing

Table 2: Unit Testing

S.N.	Test Case ID	Test Description	Steps Executed	Expected Result	Actual Result	Pass / Fail
1	UT-001	Invalid Format	Send a pdf format to a video input.	Program should not accept that file	Error sends and program didn't accept the file	Pass
2	UT-002	Read the video in frames	Run the file video_frames Video_utlis.py file.	Program should not be able to read the video file by each frame.	Video frames read and printed successfully.	Pass
3	UT-003	Initialize trackers and get tracks from stub files.	Run Tracker module and run it with a pretrained model.	Utilize a previous stubs file or create a new stubs file.	New stubs file created.	Pass
4	UT-004	Camera movement Estimator	Run the camera_movement _estimator.py file	Return two values to estimate the camera movement with respect to the field	X and Y value returned with each frame for camera movement estimator	Pass
5	UT-005	Interpolate Ball Position	Run the Tracker.py file which contains interpolate_ball_positio n and pass the track value.	Ball position with each passing frame should be returned.	Ball position returned in X and Y value with respect to field.	Pass

6	UT-006	Track Objects	Process tracked detections and calculate position as the center of bounding box.	Create a box covering the object i.e., players and keep the object in the center.	Boundary box is returned with respect frames and tack_id.	Pass
7	UT-007	Track Objects	Process tracked detections and calculate position as the center of bounding box	Create a box covering the object i.e., players and keep the object in center.	Boundary box is returned with respected frames and track_id.	Pass
8	UT-008	Ellipse to notify players	Function call draw_ellipse from utilis.py	Open CV draws an ellipse under a player with given boundary box value.	A small elegant ellipse is drawn which beautifully runs with the player.	Pass
9	UT-009	Draw Triangle	Function call draw_triangle from utilis.py.	The player that consists the ball should have a triangle over the head.	Both team player who has the ball have triangle over the head.	Pass
10	UT-010	Draw annotations	Run Player_ball_assigner.py	Annotations with speed and distance should be drawn.	Annotations with speed and distance are drawn.	Pass
11	UT-011	Returning the saved video to web browser through fast API	Send a request from frontend.	The processed video should be returned through API.	Video is available to view and download.	Pass

5.3 Result Analysis

The result analysis expects the difference in Football Detection with and without the use of K-Means algorithm to extract the data within the video.

a) Without K-Means algorithm: Not using K-Means algorithm means the use of UMAP alone will reduce the dimensionality of extracted features but won't categorized in distinct group. No automatic classification is done making it harder to extract insights like "the fastest player on the pitch of one team". Thus, this results in no difference than manual observation.

Figure 5.1: Without K-Means algorithm

b) With K-Means: Using K-Means algorithm means the use of UMAP and K-Means with dimensionality of extracted features and categorize in distinct group. Automatic classification is done making it easier to extract insights like "the fastest player on the pitch of our team". Thus, this results in vast difference than manual observation. In this, each data point is assigned a color-coded cluster for team difference which make it easier for interpretation of team's playstyle. Furthermore, the automatic player classification enables us for more opportunities for added functionality like heat maps, trajectories of teams, ball possession.

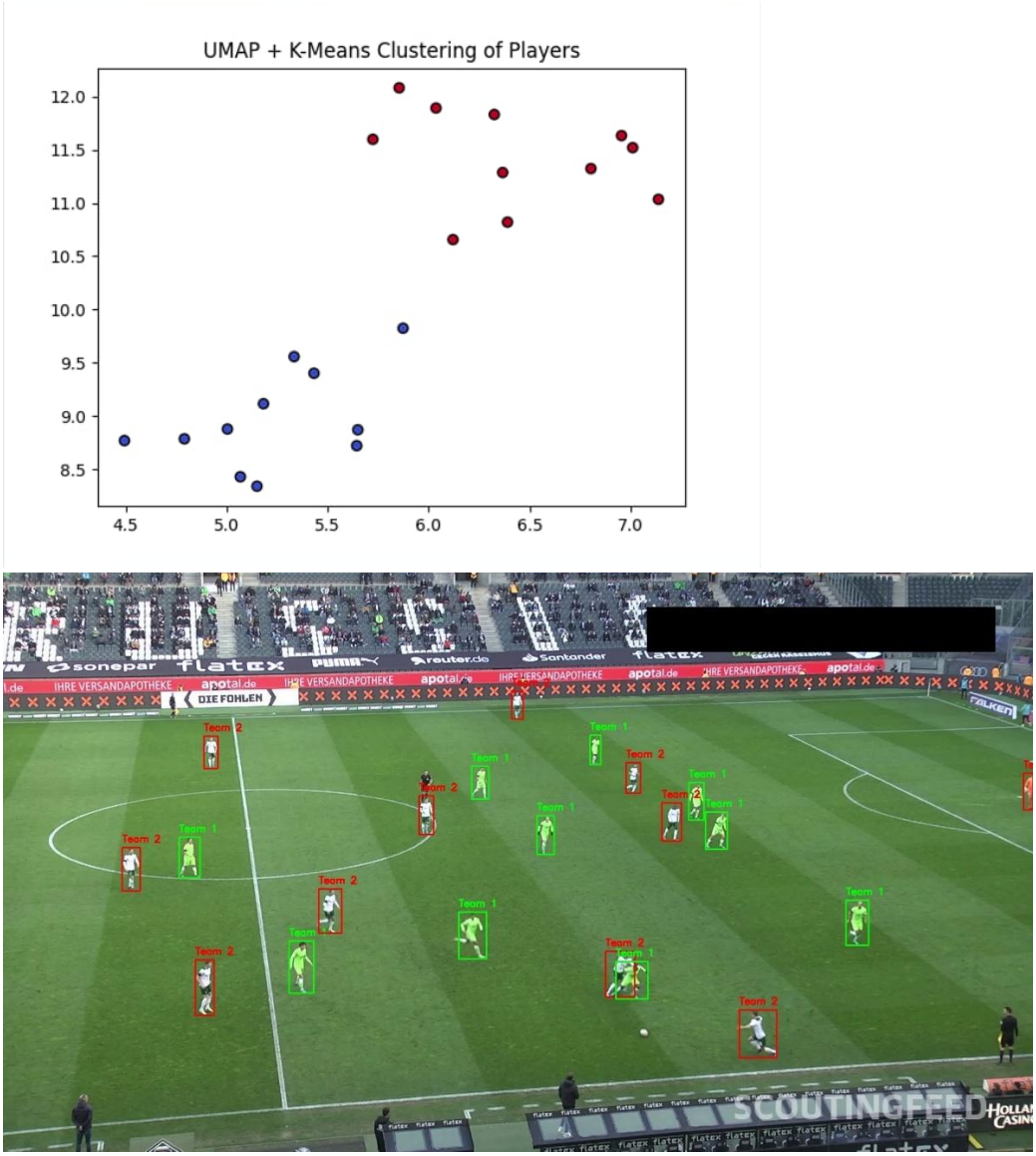


Figure 5.2: Using K-Means algorithm

Chapter 6: Conclusion and Future Recommendations

6.1 Conclusion

The development of the Football Analytics System represents a significant step forward in modernizing traditional methods of game analysis. By leveraging advanced algorithms such as YOLO for object detection, Byte Track for player tracking, and the Lucas-Kanade algorithm for motion estimation, the system provides automated, real-time insights into player movements and team dynamics. The integration of K-Means clustering for team assignment further enhances its analytical capabilities. This system promises to reduce manual effort, improve accuracy, and foster a data-driven approach to football analysis. To ensure its success, careful attention must be given to usability, regular updates, and responsiveness to user feedback, paving the way for a more accessible and efficient analytics solution for all levels of the football ecosystem.

6.2 Future Recommendations

The expected outcomes of implementing the Football Analytics System include:

- Efficiency Improvement: Automating player detection, tracking, and team assignment through advanced algorithms like YOLO and Byte Track is expected to significantly reduce manual analysis efforts, streamlining the overall process.
- Accuracy Enhancement: Leveraging machine learning algorithms ensures precise tracking of players and reliable team classification, minimizing errors commonly associated with manual or less advanced systems.
- Dynamic Analysis: The integration of the Lucas-Kanade algorithm for motion estimation and K-Means clustering for team assignments enables dynamic and adaptive insights, allowing the system to handle real-time scenarios and varied game conditions effectively.

References

[1] Ultralytics, [Online]. Available: https://docs.ultralytics.com/. [Accessed August 2024].

[2] K. Kitani, "Lucas-Kanade Optical Flow," Carnegie Mellon University, 1981.

[3] Vina, Abirami, "roboflow," 21 August 2024. [Online]. Available: https://blog.roboflow.com/what-is-bytetrack-computer-vision/. [Accessed 29 August 2024].

[4] META, "React Documentation," Facebook, [Online]. Available: https://react.dev/reference/react. [Accessed August 2024].

[5] "FastAPI," [Online]. Available: https://fastapi.tiangolo.com/. [Accessed August 2024].

[6] O. team, "openCV," [Online]. Available: https://opencv.org/. [Accessed August 2024].

[7] U. Inc., "ultralytics," Ultralytics Inc., [Online]. Available: https://www.ultralytics.com/. [Accessed August 2024].

[8] Alexey Bochkovskiy, Chien-Yao Wang, Hong-Yuan Mark Liao, "YOLOv4: Optimal Speed and Accuracy of Object Detection," 23 05 2020.

[9] Redmon, J., & Farhadi, A, "YOLOv3: An Incremental Improvement.," 2018. [Online]. Available: https://arxiv.org/abs/1804.02767. [Accessed August 2024].

[10] Shankara Narayanan, Syed Ashfaq Ahmed, Sneha Varsha, "Real-Time Object Detection for Football Analytics: Journal of Sports Analytics," 2021.

[11] Kouidri, Allan, "Mastering Deep Sort: The Future of Object Tracking Explained," ikomai, 11 October 2023. [Online]. Available: https://www.ikomia.ai/blog/deep-sort-object-tracking-guide. [Accessed August 2024].

[12] Yifu Zhang, Peize Sun, Yi Jiang, Dongdong Yu, Fucheng Weng, Zehuan Yuan, Ping Luo, Wenyu Liu, Xinggang Wang, "ByteTrack: Multi-Object Tracking by Associating Every Detection Box.," 2021. [Online]. Available: https://arxiv.org/abs/2110.06864. [Accessed August 2024].

[13] R. Szeliski, " Computer Vision: Algorithms and Applications," Springer, 2010.

[14] Lucas, B. D., & Kanade, T, "An Iterative Image Registration Technique with an Application to Stereo Vision. Proceedings of the 7th International Joint Conference on Artificial Intelligence (IJCAI)," 1981.

[15] "Oneture," Oneture Technologies, July 2023. [Online]. Available: https://oneture.com/blog/football-analytics-computer-vision. [Accessed August 2024].

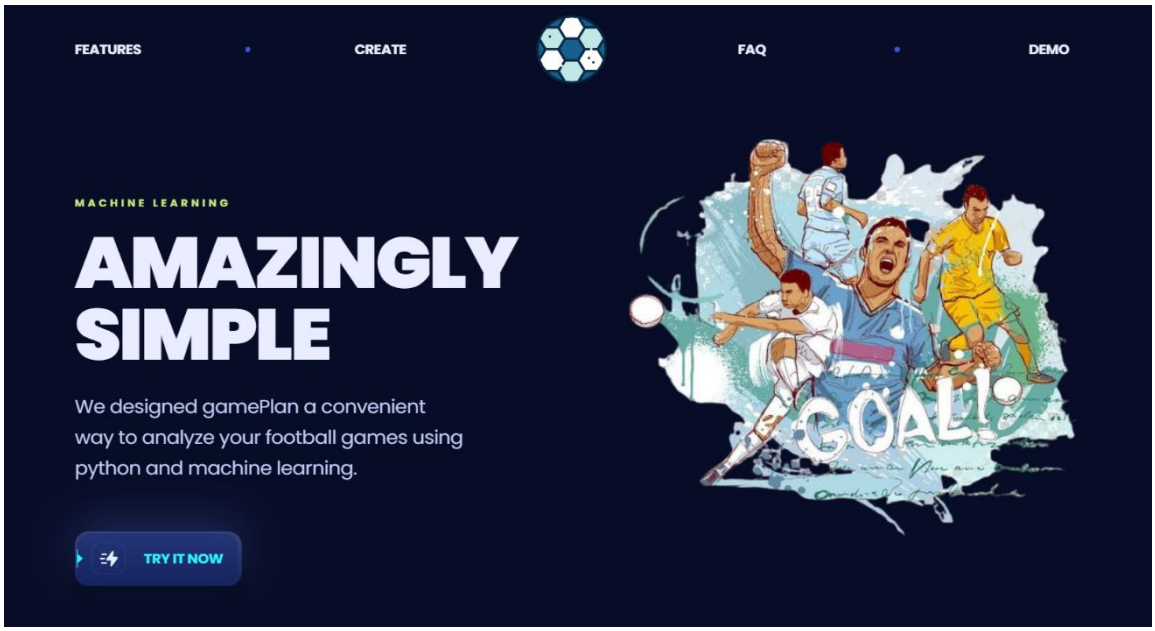
[16] N. Js, "Node JS," [Online]. Available: https://nodejs.org/en.

[17] Okeke, "Agile Methodology in System Developmen," 2021. [Online]. Available: https://targettrend.com/agile-methodology-meaning-advantagesdisadvantages-more.

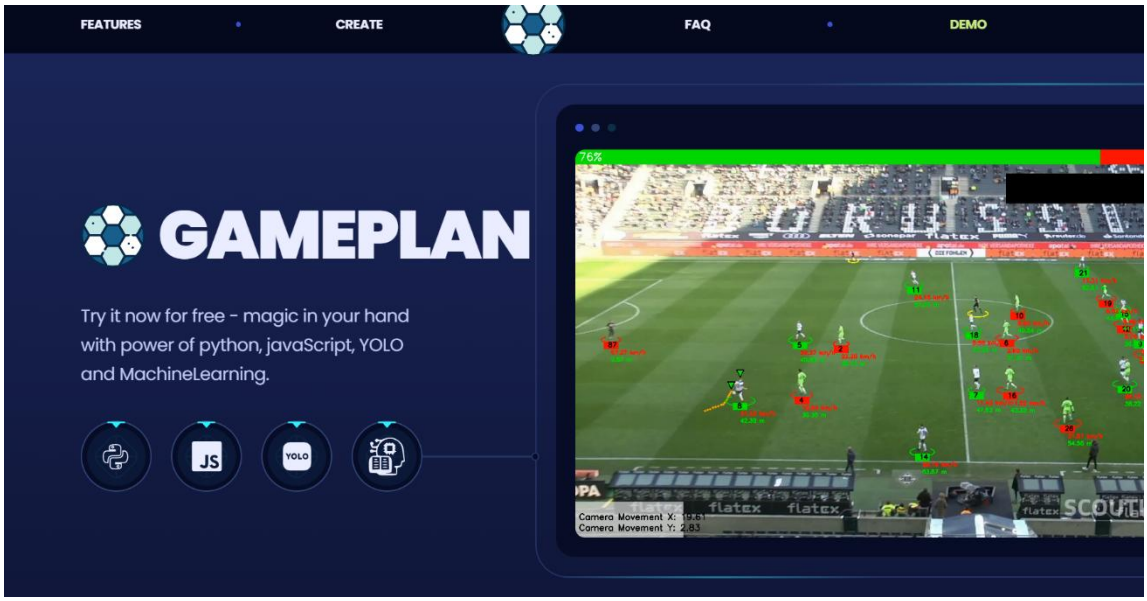
[18] OKeke, "research gate," targettrend, 2021. [Online]. Available: https://targettrend.com/agile-methodology-meaning-advantagesdisadvantages-more. [Accessed september 2024].

Appendices

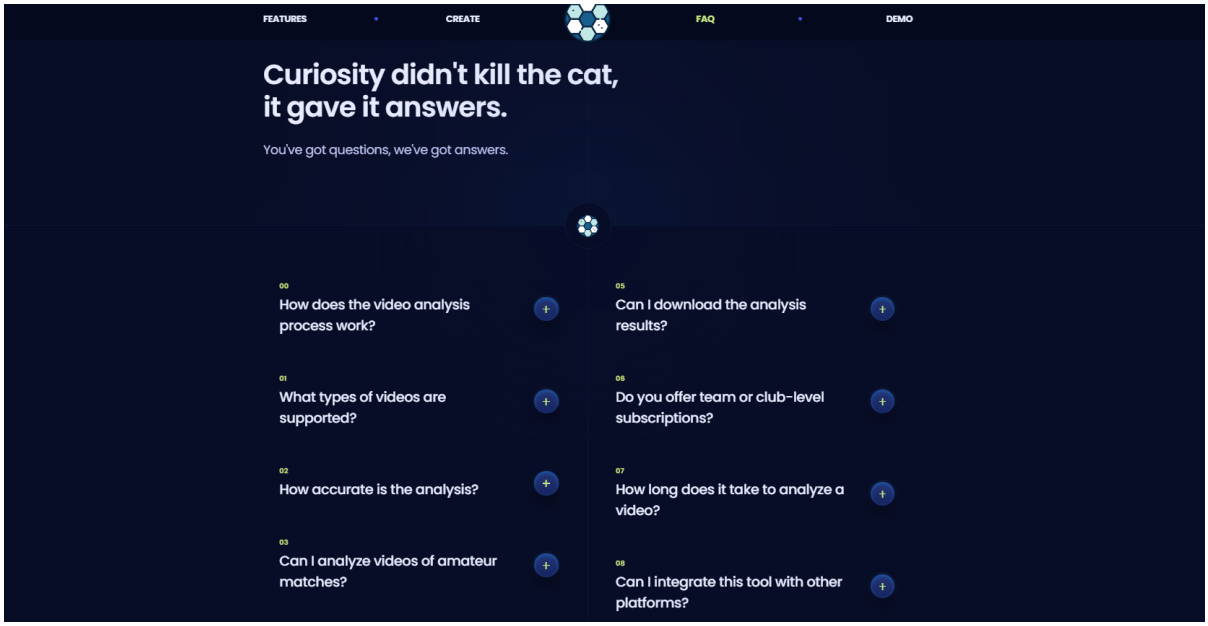
a) Snapshots



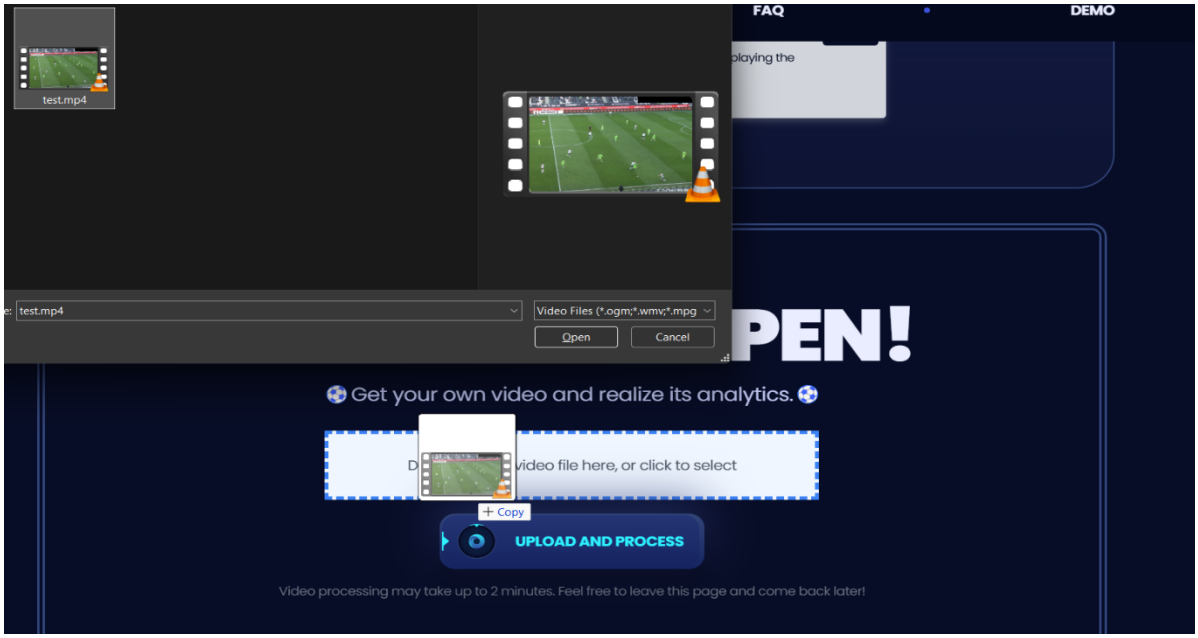
Snapshot 1: Hero section of our Web app



Snapshot 2: Demo Section



Snapshot 3: Frequently Asked Question

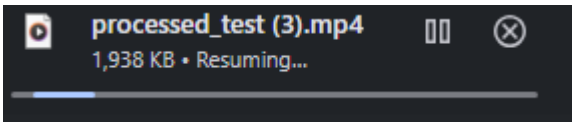


Snapshot 4: Drag and drop your video

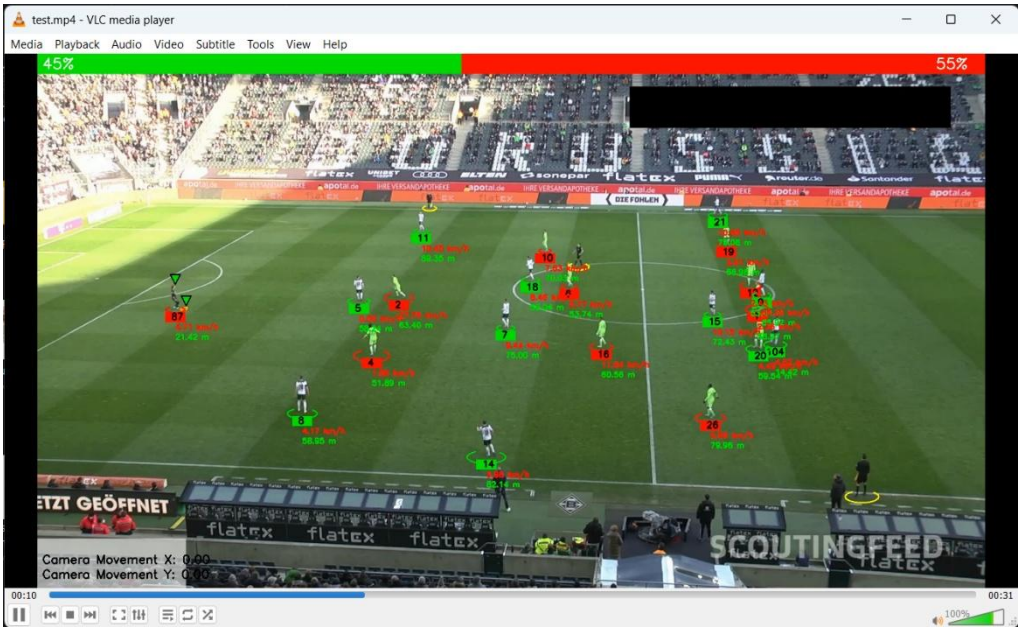
```
PS F:\subjects\CR7th sem\FastAPI-React-Integration\gameplan> cd .\backend\
PS F:\subjects\CR7th sem\FastAPI-React-Integration\gameplan\backend> .\venv\Scripts\activate
(venv) PS F:\subjects\CR7th sem\FastAPI-React-Integration\gameplan\backend> py .\main.py
INFO:     Started server process [22228]
INFO:     Waiting for application startup.
INFO:     Application startup complete.
INFO:     Uvicorn running on http://0.0.0.0:8000 (Press CTRL+C to quit)
INFO:     127.0.0.1:55089 - "OPTIONS /upload-video/ HTTP/1.1" 200 OK
Video uploaded to: input_videos\test.mp4
INFO:     127.0.0.1:55089 - "POST /upload-video/ HTTP/1.1" 202 Accepted
Using existing stub file: stubs/test_track_stubs.pkl
Loaded tracks from stub: stubs/test_track_stubs.pkl
Using existing stub file: stubs/camera_movement_stub.pkl
Camera movement data type: <class 'list'>
First element in camera movement list: [0, 0]
Cropped image saved at: output_videos\cropped_image.jpg
Video saved successfully to output_videos\processed_test.mp4
Processed video saved at: output_videos\processed_test.mp4
```

Snapshot 5: Backend server processing the input

Snapshot 6: Output stream



Snapshot 7: Downloading video



Snapshot 8: Downloaded video

b) Source Code for KMeans implementation:

```python
import numpy as np

class CustomKMeans:
    def __init__(self, n_clusters=2, max_iters=100, tol=1e-4, random_state=42):
        self.n_clusters = n_clusters
        self.max_iters = max_iters
        self.tol = tol  # Convergence threshold
        self.random_state = random_state
        self.centroids = None

    def initialize_centroids(self, X):
        """Initialize centroids using K-Means++ initialization."""
        np.random.seed(self.random_state)
        centroids = [X[np.random.choice(X.shape[0])]]

        for _ in range(1, self.n_clusters):
            distances = np.min([np.linalg.norm(X - c, axis=1) for c in centroids], axis=0)
            probabilities = distances / distances.sum()
            new_centroid = X[np.random.choice(X.shape[0], p=probabilities)]
            centroids.append(new_centroid)

        return np.array(centroids)
```

```python

def fit(self, X):
    """Run K-Means clustering on the dataset X."""
    self.centroids = self.initialize_centroids(X)

    for _ in range(self.max_iters):
        # Step 2: Assign each point to the closest centroid
        distances = np.linalg.norm(X[:, np.newaxis] - self.centroids, axis=2)  # Compute distances
        labels = np.argmin(distances, axis=1)  # Assign clusters

        # Step 3: Compute new centroids
        new_centroids = np.array([X[labels == i].mean(axis=0) for i in range(self.n_clusters)])

        # Step 4: Check for convergence
        if np.linalg.norm(self.centroids - new_centroids) < self.tol:
            break  # Stop if centroids do not change significantly

        self.centroids = new_centroids  # Update centroids

    return labels

def predict(self, X):
    """Assign clusters based on the fitted centroids."""
    distances = np.linalg.norm(X[:, np.newaxis] - self.centroids, axis=2)
    return np.argmin(distances, axis=1)

```