



TED UNIVERSITY

CMPE 491

Senior Design Project I

VENATOR

High-Level Design Report

SECTION 1 - GROUP 6

07/01/2024

Team Members:

- **Arda Uyaroğlu**
- **Berker Tomaç**
- **Selçuk Akif Topkaya**
- **Utku Oktay**

Table of Contents

1	Introduction.....	3
1.1.	Purpose of the System	3
1.2.	Design Goals.....	3
1.2.1	Usability	3
1.2.2	Reliability	3
1.2.3	Security	3
1.2.4	Reusability	3
1.2.5	Robustness	4
1.2.6	Maintainability	4
1.2.7	Availability	4
1.2.8	Efficiency	4
1.2.9	Probability	4
1.2.10	Flexibility.....	4
1.3.	Definitions, Acronyms and Abbreviations.....	4
1.3.1	Definitions	4
1.3.2	Acronyms and Abbreviations	5
1.4.	Overview	5
2	Current Software Architecture	6
3	Proposed Software Architecture	8
3.1.	Overview	8
3.2.	Subsystem Decomposition.....	8
3.3.	Hardware/Software Mapping	9
3.4.	Persistent Data Management	11
3.5.	Access Control and Security	11
3.6.	Global Software Control.....	12
3.7.	Boundary Conditions.....	12
4	Subsystem Services	13
5	Glossary	15
6	References	16

1 Introduction

1.1. Purpose of the System

The main purpose of the system is to detect and track various situations in a football match. Many events can happen during a football match. For example, to visualize, when you follow the players with your eyes while the match is being played, you cannot understand the average position of the players or players or the distance they have traveled. The companies that provide access to this data provide this data support with one or more cameras or equipment on the players during the match. Venator's basic principle of providing data is that it extracts the data correctly after uploading the match video to the system. Additionally, some prominent features of Venator are: it finds players, localizes and tracks their locations, calculates the distance traveled by each player, calculates the average position of each player in the match and produces a heat map, finds the estimated position of the ball and calculates the time of possession of the ball for each team.

1.2. Design Goals

1.2.1 Usability

Venator's easy-to-use interface and simple structure, enable people who have access to the system to quickly and easily reach which section and take the action they want to perform. Venator aims to facilitate operation and save time by offering a simple interface and website usage.

1.2.2 Reliability

Venator operates with acceptably high accuracy. Since Venator is a system that works on statistics in the field of football, it is very important that the data is accurate and measurable. It calculates these statistics by applying computer vision and presents them with a remarkable degree of accuracy.

1.2.3 Security

Venator provides a secure environment that protects privacy. It controls access to the data in its databases through the authentication process. Venator does not allow unauthorized users to use the features of the application. Since it has an encrypted access control mechanism, it has a secure system and does not allow any access to this data.

1.2.4 Reusability

The definition and submodules of the Venator system are as reusable as possible to allow for further upgrades. The modularity structure does not make it difficult to update the application, and thus system components can be reused.

1.2.5 Robustness

The Venator system has a structure that is resistant to any type of entry or situation. Any user action does not negatively affect any other action. Error management works carefully.

1.2.6 Maintainability

Venator creates errors to find a quick solution in case the system crashes. It is designed to allow possible problems to be easily found and resolved. In addition, it offers a quick solution as the system can be easily upgraded to newer versions.

1.2.7 Availability

Venator aims to offer ease of use because it has a simple structure. Since it does not involve any installation steps, it has a system that is easily accessible and usable in different teams, fields, environments, or locations. One of the goals of the system is that it is ready for use in any location where there is internet.

1.2.8 Efficiency

Venator, as a system, has a fast response time and fast conclusion structure, this structural situation increases the performance of the system. In this way, the desired tasks and statistics reach the user efficiently by spending less time.

1.2.9 Probability

Venator is designed to provide results by providing statistical data on the events and situations that occur during the match on the football field. It is a product that increases the possibility of obtaining accurate data by creating a new perspective by applying computer vision and blending traditional information.

1.2.10 Flexibility

As a system, Venator is open to changes in the future and is suitable for application in a new environment.

1.3. Definitions, Acronyms and Abbreviations

1.3.1 Definitions

Authentication: It is the name of the process of verifying a user's identity when logging into the system.

Heat Map: An indication of a player's effectiveness in various areas of the field.

Detection: Finding and identifying items inside a picture or video clip.

Tracking: Associate detections in more than one frame.

Computer Vision Techniques: Analyzes and interprets visual data from an image or video clip using various processing techniques and algorithms.

Encrypted Access Control Mechanism: Encrypts authentication and verification procedures to create a safe environment for data storage and retrieval.

Average distance covered : The average distance a player runs during a match.

Average position: The average location of possession of the ball by a team or player throughout a match.

Ball possession: The name given to the rate at which a team keeps the ball compared to the other team throughout the match.

Mean Average Precision: Measure the performance of the object detection model using precision values.

1.3.2 Acronyms and Abbreviations

YOLO: You Only Look Once

API: Application Programming Interface

UI: User Interface

1.4. Overview

Venator is a program that produces statistics from football match images and aims to help people or institutions that have this system. Instead of making a simultaneous and error-based calculation during the match, it has the principle of extracting data by examining the record of a completed match. It aims to extract statistics easily and quickly wherever the internet can be accessed, with a simple, understandable interface and a system designed for easy use. To explain in more detail, football teams play matches in many countries and locations other than their stadium and city throughout the season. This match video is available from the football team's coaching staff. Instead of watching the match video on video, the aim is to access the desired statistics quickly and without wasting time by uploading it to Venator during an away trip, during a team meeting, or while having breakfast. Venator provides access to this data by allowing the user to:

- It detects and tracks players' locations.
- Calculates the distance traveled by each player
- Calculates each player's average position in the match and produces a heat map
- Finds the estimated position of the ball
- Calculates the time of possession of the ball for both teams

Venator has a high accuracy rate, thanks to the object detection algorithm YOLO it uses.

2 Current Software Architecture

Our project can currently detect, localize and identify the ball, the players, the goalkeepers and the referees in a footage of the game. It can also identify the teams of the players and thus enables us to be able to calculate the team-based statistics such as ball-possession and formation even without identifying which player is which. For now, team identification requires manually specifying the jersey colors, but we have been working to automate this process by using techniques like k-means clustering.

The machine learning model embedded in the project provides the predictions. These predictions are then filtered depending on the identified classes and the confidence scores provided by the machine learning model. Since it loses the ball frequently, we use the information from other frames to predict the position of the ball. We then perform a color masking on the detected players to identify their teams.

A diagram explaining the data flow is given below:

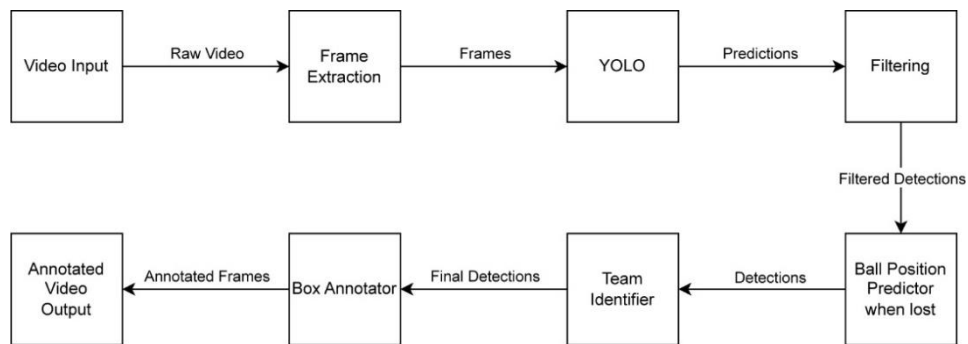



Figure 1: Data flow in current software architecture.

Although we have not started developing the back-end side, we have started developing the front-end part and it can illustrate the annotated video and display some statistics if manually provided.

Here are some screenshots regarding the front-end application of our project:

 Games

Create New Game Record

Name of the Record:

Team 1 Name:

Jersey Color:

Team 2 Name:

Jersey Color:

Video Recording of the Game:

Drag your files here or click in this area.

Figure 2: A screenshot of "Add New Game Record" page.



Figure 3: A screenshot of the analysis page. Both raw and annotated videos can be seen with the help of slider.



Figure 4: Another screenshot of the analysis page illustrating a whole frame of annotated video.

3 Proposed Software Architecture

3.1. Overview

Venator is a football analysis software designed for especially coaches and those working in football ecosystem. It is highly dependent on computationally expensive machine learning and image processing techniques and may require high-end components for fast evaluation under heavy load. Running the analysis on cloud will enable the users to perform the analyses using any device. Considering the technical staff of teams tend to use especially mobile devices such as tablets, access to the analyses and data from anywhere is an important requirement. Consequently, we decided to construct the project as a web-application in order to offload the computational load to the cloud and facilitate the use of the application.

We follow microservice architecture as it will be a lot easier and more maintainable to combine different solutions such as web APIs and analysis modules that will be built with different technologies.

3.2. Subsystem Decomposition

It would be proper to divide each sub system into different parts. We can define our system in 5 main parts and here are the explanation for all of them;

- **Object Detection Subsystem:** This system is basically the backbone of our system. As our goal is creating statistics for each team, we should be able to define what is a player, goalkeeper, ball and a referee. The system should be able to answer the question what are these keywords are and what do they looklike. This is what object detection is all about. Creating meanings like humans that are on the screen.
- **Object Tracking Subsystem:** Football isn't a sport where you just stand and play. All objects are objectively fast. Since the statistics are going to be created from the player's actions the system should track the position of each member of the system. It would be correct to make a relation between object detection and tracking. They are working together.
- **Image Processing Subsystem:** Football is a game played by two teams. That means statistics that are created should be different for each team. Our system needs to differentiate the teams in order to create statistics for them. We are using a color masking based on the color of each team's jersey. This process is being achieved by the help of the image processing.

- **Authentication Subsystem:** Our users are going to get their teams statistics by using our system. It means each player should be able to see different statistics for different teams. Our system provides an authentication system which protects the information of each member's team from different users.
- **Web Application Subsystem:** There should be an end product where the users are able to use our project. That is the part our clients are interacting with the project itself. In most basic words, they are using this subsystem to interact with the system to gather all of the information they want about their team.

3.3. Hardware/Software Mapping

There are two different components on our project. One of them is hardware components where the other one is the software component. In order to have a successful end to end system we need an application where these two components are connected to each other very well. Let's define what do we have inside of these components.

Hardware Components:

We should divide hardware components into two parts as well. First one is the components that are used by the developers and the second one that are used by the clients.

1- Developer Side

- **GPU :** Most of the hardware components that we need are PC Parts. First and the most important part is the GPU(Graphics Processing Unit). In order to achieve object detection and object tracking you should train the model to let the computer know what are the objects. This training process needs significant amount of time and processing power. Because of that reason it would be correct to state that this is the most important unit of our system
- **Camera:** We want to propose the best statistics to our stakeholder. In order to achieve that training our model with the real life examples is crucial. By using a camera we made some recordings of football games in our town.

2- Client Side:

- **Monitor:** Our clients doesn't need anything other than a working device which achieves the task of displaying graphics. By using a monitor and a web browser, our client can Access the whole part of our system.

Software Components:

Our system involves different disciplines of computer engineering such as machine learning, image processing, computer vision and web development. All of these subsystem requires different software programs to be used. Here are the software system that we are using for each subsystem:

- Machine Learning: Our application uses PyTorch and Pandas library for the machine learning tasks such as advanced techniques of the task for finding the “ball”.
- Image Processing: Since image processing isn’t one of the biggest subsystems of our application, usage of OpenCV Libraries are enough for us.
- Computer Vision: It wouldn’t be wrong to call this subsystem most important part of our application. We are using object detection library of ultralytics. We use YOLO v8 model to train and track our system. Also additional parts are achieved by the help of PyTorch Library as well.
- Web Development: We have two different sides for our web development part. First the front-end where the React library of javascript is used. Secondly the back-end part. For the back-end we are using the programming language called Golang and couchbase database to store any necessary information. We should state that the Technologies that are mentioned under this program can be changed over the development process of our application.

Mapping Between Them:

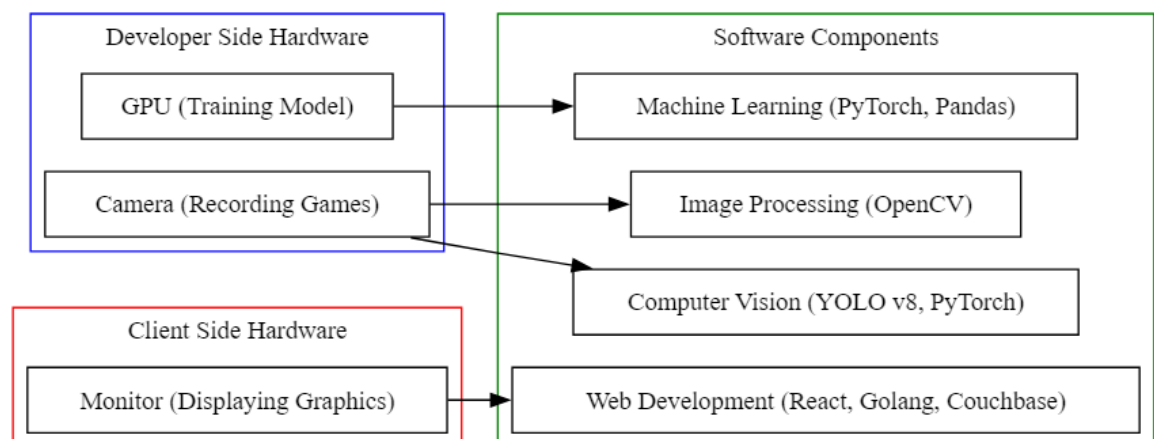


Figure 5: Hardware/Software Mapping

3.4. Persistent Data Management

Persistent data management of Venator will be handled by storing and managing the data permanently in a robust and maintainable manner, such that Venator's processes and stored data will continue to exist even after the system is turned off.

Users of the web application will be able to change & update their relevant information or reset & change their passwords on the database, which will be handled by the server in a secure and persistent manner.

To achieve the desired persistent data management, Venator will make use of the Couchbase as its database to give each user a unique ID and persistently store their e-mail address, password hash, the status of their account (whether they are verified or not), their verification string, password resetting code and the relevant information about the user that is displayed on the web application for user experience purposes.

3.5. Access Control and Security

Venator will manage its access control mechanism in a secure and reliable way by implementing an authentication & verification logic into its web application, where the end-users will be required to verify their e-mail addresses when they sign-up to the web application.

When a user creates an account by signing up either with an e-mail address and a password of their choosing or directly with their Google account using the third-party sign-up option embedded to the interface of the Sign-Up form, Venator will send them an automated verification e-mail and they will be asked to verify their e-mail address by following the steps indicated in that verification e-mail. Users will not be able to view the site content or enjoy the features of the web application unless they have their e-mail addresses marked as "verified" in Venator's database.

The passwords of a user will be encrypted regardless of the status of their account (whether they are verified or not) when they sign up to the site. The encryption is done by hashing the entered password with additional "salting" and "peppering" followed by the synchronous signing of this newly created password hash (with the JWT Secret that only the server knows) into a JSON Web Token string payload. The server generates JWTs and sends them to the client. The client (user), on the other hand, submits the JWT with every request s/he makes to the server.

3.6. Global Software Control

Global software control of Venator will be implemented for the initiation of the requests and the synchronization of the subsystems. Most requests are initiated by the button clicks throughout the different sections in the various pages of the website and the synchronization comes from awaiting the asynchronous handlings of the server endpoints (that are called with those initiations by button clicks) and performing the necessary actions according to the responses of successes or failures from those endpoints. Additionally, various time outs that are set for redirection logic between different pages of the web application contribute to the synchronization of the subsystems in those time frames.

3.7. Boundary Conditions

Boundary conditions of Venator describe the start-up, shutdown and error behavior of the system.

- **Start-up:** Start-up behavior includes navigating to the web application and the creation (initialization) of a verified account on the system. Users navigate to the site, register to the web application with an e-mail and password pair or their Google account and verify their newly created account.
- **Shutdown:** Shutdown behavior includes closing of the web application tab or logging out of an account. Users log out from their accounts by clicking on the “Log Out” button.
- **Error:** Error behavior includes the handling of different types of errors that may occur during the operation and usage of the web application. Venator handles errors from both its frontend interface and its server. It either displays a failure page to the user as a result of an error that occurred during signing up, logging in or creating/updating information using the interface or it sends an error status code as the response of the server to a request by the client.

4 Subsystem Services

Here are the detailed explanation of each subsystem of our project:

1. **Object Detection Subsystem:** As it mentioned in the second part of this report, object detection is the most crucial part of our project. Creating meanings for objects like people do in real life is one of the most challenging topics of computer engineering. In our system our main task is generating statistics for teams. These statistics are gathered by the actions of players with or without the ball. This is where object detection steps ahead. Computer doesn't recognize what is player, goalkeeper and a ball. That means we should train the computer with multiple images to let computer learn what are the meanings of the objects that we are describing to it. Here is how this step work for us:
 - Custom dataset is created by mixing recordings of the real games and professional ones.
 - The objects that are desired to be detected are labeled with data labeling tools.
 - A pre-trained general purpose pre-trained model is used to train our custom dataset with YOLO V8 model.
 - Trained custom model is inferenced by the pytorch library to make predictions of the objects that are required to found.
2. **Object Tracking Subsystem:** Object Tracking is the next part of the Object Detection part. Found objects needs to be tracked since every object in the football game are in constant movement. Tracking is a challenging subject since the objects like ball are quite fast. Detecting and Tracking are working together to achieve this process. Also some of the statistics such as the kilometers covered by a team requires constant tracking of the player. We are using a tracking algorithms that is called "ByteTrack". This is an algorithm that comes with the ultralytics library which provides our model for object detection. That also provides the information how close the two subsystem are to each other. Here is how it basically works:
 - Trained model is used to feed the system with an information
 - ByteTrack algorithm uses the trained model to look over every frame to find out specified objects in the frame.

3. **Image Processing Subsystem:** This subsystem is more like a sub-module when it is compared with the Object Detection and Tracking parts. However, it is highly valuable since it provides the solution to the one of the most important questions of the project. This question is how are the two teams are going to be differentiated by the computer. It is nearly impossible to run a face recognition system from the camera since it's too far away from players. Best way to achieve this task is using color masking operation. Two teams are required to wear jerseys with different colors. That means running a color mask on players will provide the output of the same team players which we want. Here is how this subsystem works;
 - Subsystem takes a frame of the football match.
 - It applies a color masking to the current frame
 - Only objects that can be seen in the screen are grouped.
 - After doing that step for each teams we have the information of which player belongs to the which team.
4. **Authentication Subsystems:** This subsystem feature is used when users login to their accounts. Each accounts have the access to their own team. Using this protection system each teams are able to protect their statistics from other teams. Here is how the subsystem works;
 - Users become a member and creates an account for the first time. An email and a password is set by the user. Verification code is sent to the mail of the user to be sure that user has access to his/her email.
 - When user creates his account, the access of their team are assigned to their account. This both helps to protect the classified information of the current user from different members and the information of the different members from current user.
5. **Web Application Subsystem:** This is the endpoint for the users. Each operation in order to view any kind of statistics of any match data are monitored from this subsystem. Here is the general process of how users are going to interact with the web application it self;
 - User login to their account
 - User can add a new game recording to create statistic by filling necessary parts.
 - User can view any statistics for the previous game recordings that he created.

5 Glossary

Couchbase: an award-winning distributed NoSQL Cloud Database Service that delivers unmatched versatility, performance and scalability. [\[1\]](#)

Hashing: the practice of algorithmically turning a password into ciphertext, or an irreversibly obfuscated version of itself, as a means of blocking against the threat of password breaches. [\[2\]](#)

Salting and Peppering: methods of preventing hashed passwords from being deciphered by hackers using brute force techniques. [\[3\]](#)

JSON Web Token: a compact URL-safe means of representing claims to be transferred between two parties. [\[4\]](#)

6 References

1. *Home*. Couchbase Website. (2023, November 10).
<https://www.couchbase.com/>
2. Team, S. (2023, October 25). *What is password hashing?*
<https://stytch.com/blog/what-is-password-hashing/>
3. Anderson, R. (2022, April 19). *What are password salting and password peppering?*. NetSec.News. <https://www.netsec.news/what-are-password-salting-and-password-peppering/>
4. auth0.com. (n.d.). *Jwt.io*. JSON Web Tokens. <https://jwt.io/>