**IceMetrics – Hockey Analytics – Capstone I**

Jackson Morgan, Tin Duong

Team Vision

Department of Computer Science

University of Missouri

[jkmh3n@umsystem.edu](mailto:jkmh3n@umsystem.edu), [ttdbbc@umsystem.edu](mailto:ttdbbc@umsystem.edu)

**ABSTRACT**

Hockey Analytics is a very important tool for teams, coaches, and players alike to gain a deeper understanding of the dynamics of the game being played. This project aims to develop an automated computer vision web application to analyze different aspects of hockey game footage to generate helpful statistics such as player movement patterns and shot tracking. The application aims to serve a wide audience instead of focusing on the professional side of the sport. Although this product could be used at the professional level, we hope to bring this type of technology to fans and local players alike to analyze themselves, or their favorite teams on their own accord. Our product will provide an attractive user interface for anyone to do just that with the click of a button.

# OUR TEAM

|  |  |  |
| --- | --- | --- |
| **Name** | **Email** | **Phone Number** |
| Jackson Morgan | [jkmh3n@umsystem.edu](mailto:jkmh3n@umsystem.edu) | (314) 371-7636 |
| Tin Duong | [ttdbbc@umsystem.edu](mailto:ttdbbc@umsystem.edu) | (714) 697-8772 |

## 

## MENTOR INFORMATION

|  |  |  |
| --- | --- | --- |
| **Mentor Name** | **Mentor Email** | **Other** |
| Sim Kieu | [simkieu@gmail.com](mailto:simkieu@gmail.com) | <https://www.linkedin.com/in/simkieu/> |

## MISSION STATEMENT

To create both an accessible and user-friendly hockey analytics application that leverages the power of computer vision to bring advanced statistics to the masses. The vision is to provide everyday consumers with the ability to evaluate games, as well as analyze professional teams with ease, by generating heat maps of player locations and shots, tracking total player movement, and much more. By making professional grade insights into the game of hockey accessible to all, this project aims to bridge the gap between amateur and professional technology for analyzing games allowing anyone to appreciate the complex game hockey is.

# TABLE OF CONTENTS

[1. OUR TEAM 1](#_Toc184922328)

[1.1 MENTOR INFORMATION 1](#_Toc184922329)

[1.2 MISSION STATEMENT 2](#_Toc184922330)

[TABLE OF CONTENTS 3](#_Toc184922331)

[2. INTRODUCTION 4](#_Toc184922332)

[2.1 PROBLEM STATEMENT 4](#_Toc184922333)

[2.2 PROBLEM RESOLUTION 4](#_Toc184922334)

[2.3 BENEFITS OF SOLUTION 4](#_Toc184922335)

[3. REQUIREMENTS 6](#_Toc184922336)

[3.1 HARDWARE REQUIREMENTS 6](#_Toc184922337)

[3.2 FUNCTIONAL REQUIREMENTS 6](#_Toc184922338)

[3.3 NON-FUNCTIONAL REQUIREMENTS 8](#_Toc184922339)

[4. RESEARCH 11](#_Toc184922340)

[4.1 RESEARCH INTRODUCTION 11](#_Toc184922341)

[4.2 FRONTEND 11](#_Toc184922342)

[4.3 BACKEND 18](#_Toc184922343)

[4.4 COMPUTER VISION 21](#_Toc184922344)

[8. SYSTEM DESIGN 27](#_Toc184922345)

[8.1 INTRODUCTION 27](#_Toc184922346)

[8.2 METHODOLOGY 27](#_Toc184922347)

[8.3 SYSTEM ARCHITECTURE 29](#_Toc184922348)

[8.4 PSEUDOCODE AND OTHER CODE 31](#_Toc184922349)

[9. PROJECT DEVELOPMENT 39](#_Toc184922350)

[9.1 DEVELOPMENT STRATEGY 39](#_Toc184922351)

[9.2 DEVELOPMENT SCHEDULE 40](#_Toc184922352)

[9.3 WORK DELEGATION 42](#_Toc184922353)

[9.4 POSSIBLE PROBLEMS 43](#_Toc184922354)

[9.5 TESTING PLAN (FOR FUTURE DEVELOPMENT) 43](#_Toc184922355)

# INTRODUCTION

## PROBLEM STATEMENT

Hockey is one of the most fast paced sports with many confusing aspects to the game. The ability to understand game analytics provides a valuable insight for fans and players alike. The professional hockey league known as the NHL (National Hockey League) recently unveiled the implementation of computer vision aided analytics software that provides such advanced insight into professional hockey games. But, as of now, the access to these analytics such as shot tracking and player movement heatmaps is limited to those teams with the technology to pull it off. There are very few alternatives for consumer use that can be applied to amateur and other professional leagues. This has created a barrier for everyday hockey players and fans who want to have the ability to evaluate themselves or their favorite teams.

## PROBLEM RESOLUTION

To address the known lack of accessible advanced hockey analytics software for everyday players and fans, a solution that uses the power of computer vision technology and a user-friendly web application design to make a simple and easy to understand product for analyzing hockey footage. This will allow nonprofessionals and the average consumer to analyze gameplay footage and receive personalized statistics for individual and team performance. This application will fill the void in the consumer space when it comes to advanced hockey video analytics.

## BENEFITS OF SOLUTION

The application will provide a numerous number of benefits to the hockey community, more specifically those that aren't professionals such as high school and club players. By leveraging computer vision technology, the application will bring highly advanced automatic statistics and analysis to all fans and players alike. It will feature player movement tracking, shot and location heatmaps, and a variety of other gameplay statistics, which will enable anyone to enhance their knowledge of the game of hockey with the click of a button. Designed with user friendliness in mind, the application will make it easy for people of all ages and abilities to use our technology. Furthermore, the application will be a cost-effective alternative to the very minimal amount of solution available today. By bridging the gap between consumers and professional grade analytics, this solution will bring hockey knowledge to the masses.

# REQUIREMENTS

## HARDWARE REQUIREMENTS

These specifications are the expected minimum hardware requirements for the application to run and analyze game footage. This could be changed later if needed to ensure the right fit for everyday use.

* **Operating System:** Windows 10 or macOS (latest version available))
* **Processor (CPU):**
  + Minimum: Intel Core i5 or AMD Ryzen 5 (most likely 6 cores or more)
  + Recommended: 8 cores or more for head room (Intel Core i7 or AMD Ryzen 7)
* **RAM:**
  + Minimum: 8GB
  + Recommended: 16GB or more, this could depend on the sizer of the video files being provided
* **Graphics Card (GPU):**
  + Minimum: Integrated Graphics (Intel UHD Graphics, AMD Vega)
  + Recommended: NVIDIA GTX 1650 or AMD RX 570
* **Storage:**
  + Minimum: 256GB SSD or HDD
  + Recommended: 512GB SSD or larger which would provide a faster read and write speed than the HDD alternatives

## FUNCTIONAL REQUIREMENTS

* **Video Uploading and Processing:**
  + Allow users to upload footage of any hockey game in a standardized video format for analyzing.
  + Be able to validate the quality of the video to make sure it is usable with our computer vision algorithms.
  + Use either cloud or local processing of the given footage, depending on how the system will be deployed.
* **Object Detection and Tracking:**
  + Detect objects within the video, such as players, the puck, then net, and any other objects that are essential for the statistical calculations that the program will provide for the user.
  + Ability to assign unique characteristics to each player such as an ID number to track them across multiple instances.
  + Identify events throughout the game footage such as shots and passes, as well as static events such as faceoffs.
* **Heatmap Creation:**
  + Create detailed heatmaps for different statistics such as, player hotspots and shot locations.
  + Ability to display the heatmaps in an attractive and simple UI for easy understanding and analyzing (shown in the context of the entire hockey ice rink)
* **Individual Player and Team Statistics:**
  + Provide a user interface that allows the user to:
    1. Upload videos for analysis
    2. View any statistic and heatmaps for analysis
    3. Filter the data by for the desired analysis
       1. Player
       2. Period of Play
       3. Team

## NON-FUNCTIONAL REQUIREMENTS

## 

These non-functional requirements are here to define system performance requirements and constraints that our application must adhere to. These requirements ensure that the application is functional, and reliable, efficient, and user friendly. These are the requirements specified for the application:

* **Performance**
  + **Response Time**: The application requires user input of video footage from a hockey game. The system should process and analyze this video within a reasonable time frame, this being no longer than 10 minutes as that should be ample time given the recommended hardware and software.
  + **Throughput**: The system should be able to handle the input of multiple different videos concurrently analyzing all of them. The program aims to support at least 5 simultaneous uploads while maintaining high performance and speed.
* **Usability**
  + **User** **Interface** **(UI)**: The application should be both intuitive and aesthetically pleasing for the user, allowing anyone who uses the system to navigate through the different pages in a simple manner. This includes components such as navigation and links to redirect users. Additionally, implementation of redirection can implement and automate certain actions with separate pages making the experience smooth and seamless.
  + **Documentation**: Comprehensive documentation and clear guidelines should be provided to ensure a smooth and transparent process, allowing users to understand how the output data is handled.
* **Security**
  + **Data** **Protection**: The application must ensure the protection of user-provided data, including information from the authentication process and uploaded hockey footage. This data should be securely stored and transferred using supported system protocols.
  + **Authentication**: Implementation of a secure user authentication process, such as multi-factor authentication, as well as role-based authentication for admins (if needed) to provide data integrity and safety.
  + **Regulation**: Compliance with data protection standards to safeguard data and user privacy must be implemented with the system to ensure the application is reliable and safe. This also includes managing the database carefully and heavy unit and integration testing on the system before deployment.
* **Maintainability**
  + **Modularity**: The design of the system should facilitate the creation of updates, enhancements, and other maintenance while not affecting other components of the application.
  + **Code Quality**: Best coding practices should be used, such as commenting as well as proper documentation.
  + **Testing**: Implementation of unit tests and integration tests to make sure that new features, updates, or maintenance does not introduce any bugs. This also ensures that the system can handle the required task properly and provide valid and acceptable results with a high confidence level.
* **Portability**
  + **Cross-Platform**: The web application should be compatible with all the major web browsers such as Chrome, Firefox, Safari, and Edge. It should operate seamlessly across operating systems as well (Windows, macOS, Linux, etc.).
  + **Devices**: The application should be optimized for the usage across a plethora of devices, such as desktops, laptops, tablets, and smartphones. The user experience should be consistent across all platforms.
* **Efficiency**
  + **Resource Usage**: The application should be optimized to make good use of allocated system resources such as the CPU, RAM, and storage. This will ensure the smooth operation of the system during the video processing stage.

# RESEARCH

## RESEARCH INTRODUCTION

This section of the project documentation aims to research and evaluate different aspects of the project that have many variations in the ways to approach them. This involves both the creation of the web application as well and the computer vision algorithms and design. This research will explore and identify effective approaches for developing the application while analyzing the rationale behind decisions of different frameworks and languages for the implementation for the hockey analysis application.

## FRONTEND

### INTRODUCTION

The frontend of a web application is vital to the success of any projects. It acts as a medium between the user and the system, transforming what is a very complex application into user friendly, aesthetically pleasing product for the user. Without a well thought out, structured frontend, even the most advanced technologies with a strong backend will fail, as it would be extremely difficult for the user to interact with the system. In addition, it provides an interface for the user making it less complex and providing additional support to interact with the backend of the system to render and display processed results for the videos.

### PROBLEM STATEMENT

The success of the application relies heavily on the ability to portray the advanced analytics in an intuitive and user-friendly manner. Users will need a streamlined user interface that ensures a smooth and responsive experience across all their devices. Many different frontend frameworks provide unique tools for doing such a thing for the project, but all of them come with different tradeoffs between performance, and customization. The goal is to choose the most compatible and reliable framework that integrates well with the system. Additionally provides tools for the frontend that make the design process as simple as possible.

### RESEARCH

### FRAMEWORKS

Frontend frameworks provide developers with a structure for building the user interface side of an application in a more efficient manner. These frameworks provide pre-made libraries and other functions to create a dynamic application using reusable components. Frameworks also streamline other elements such as routing and event handling, which are essential when trying to make a responsive UI design. Given the wide range of frontend frameworks available, it was important to select one that is modern, efficient, and widely adopted, ensuring access to a broad ecosystem of libraries for creating a highly responsive and user-friendly UI. To achieve this, the search was narrowed down to three of the most popular frameworks: React.js, Vue.js, and Angular.

1. **React.js [1]**
   1. Pros:
      1. Component based modular design optimized for creating reusable and maintainable code.
      2. React utilizes a virtual DOM (Document Object Model), enabling more efficient and responsive UI updates by reducing direct manipulation of the real DOM. This is particularly beneficial when frequent UI updates are required, such as when processing new information from an uploaded video. The virtual DOM enhances performance by optimizing update operations.
      3. Has a large community of developers which offers an extensive ecosystem of resources such as third-party libraries for problems that could arise during development. This makes development simplistic and efficient when searching for solutions.
      4. Highly flexible, allowing new libraries to be integrated seamlessly into the code without disrupting existing functionality, making it essentially plug-and-play.
   2. Cons:
      1. React is more of a library than a full-fledged framework, meaning it relies heavily on integration with additional tools and libraries to achieve optimal functionality. This can make setting up React projects more complex, as not all features are available out of the box.
2. **Vue.js [2]**
   1. Pros:
      1. Lightweight and beginner friendly with a clear and concise syntax.
      2. Vue features a reactive two-way binding system that ensures smooth communication between the UI and the underlying data model. This system ensures that any changes to the data automatically update the UI, and vice versa—modifications in the UI are automatically reflected in the data model. Vue efficiently tracks dependencies between the UI and the data model, updating only the affected parts of the DOM rather than the entire page, improving performance.
      3. Offers comprehensive documentation and is continuously expanding, ensuring ongoing growth in third-party support and library availability.
   2. Cons:
      1. Has a smaller community compared to React and Angular, resulting in fewer third-party plugins and libraries available to support app development.
      2. Not much of a defined pattern for the structure of a project such as components or modules, which makes the creation of large-scale applications harder to accomplish.
3. **Angular [3]**
   1. Pros:
      1. A very complete, comprehensive framework with the tools and libraries needed to build a full application from the ground up. This includes things such as built in handling and routing which other frameworks don't come with without the use of installed libraries.
      2. Much like Vue, Angular uses two-way data binding to simplify UI updates using application state. Meaning that only the specific sections of the application that need to be updated with the new information are changed, leading to a much more efficient application.
      3. Angular will continue to have long term support which ensures the stability of our product. One reason being that the framework is currently backed by google which ensures that it will continue to get updates and support for the foreseeable future.
   2. Cons:
      1. Angular has a much steeper learning curve than both React and Vue, although it has a component system which is a benefit for the React framework, the way Angular handles the same system is complex. This is also due to specific angular syntax that makes it hard to understand.
      2. It is a much heavier framework compared to something like Vue or React, which can lead to worse performance.
4. **Choice**

The frontend framework of choice for the systems use case will be React due to its features, flexibility and simplicity. The goal was to select a framework that is both lightweight and modular, providing full control over the application while benefiting from extensive community support with a wide range of third-party libraries and components to enhance development efficiency. While Vue offered similar advantages, familiarity with JavaScript and React made it a more practical choice for creating the application.

### STYLING

The styling of a web application plays a crucial role in the success of a project. Even the most advanced technologies can fail without an intuitive user interface. Utilizing a framework based on HTML and JavaScript (React), CSS will be used to ensure full control over the application’s design, providing a user-friendly experience for interacting with the hockey analytics system.

While CSS offers complete control over the appearance of an application, managing it can become complex without proper organization or a design system. To simplify the process, a CSS framework like Bootstrap or Tailwind can be used, providing built-in classes and components that allow for faster development of the app’s designs without starting from scratch. Based on prior experience, Tailwind CSS was chosen for this project due to its extensive library of components and classes that are essential for creating a responsive application. Additionally, Tailwind’s customizable configuration files allow for defining custom colors and spacing, ensuring a consistent look across all pages [4].

Finally, Tailwind was chosen for its modern, lightweight approach to styling. It utilizes the production build by removing unused or overwritten styles from the final output, ensuring a more efficient and easier use of the application [4].

### DATA VISUALIZATION

Data Visualization is a key component of this project, as the main objective behind the front end is the ability for the user to read helpful statistics and graphs through clear and engaging visual representations of the analytics. When choosing what tool would be most compatible to make these visualizations, it had to work seamlessly with React without many issues. Here are a few of the most relevant tools for this task:

1. **D3.js [5]**
   1. Pros:
      1. Flexible, a lot of customization for making any visualization from scratch
      2. Complex animations and interactive features which is grade for user engagement
      3. Community support that could be helpful for the creation of graphs for the application
   2. Cons:
      1. Steep learning curve, time consuming to learn and use
      2. Integrating with react is not straightforward
2. **Chart.js [6]**
   1. Pros:
      1. Simple and easy to implement due to its boilerplate charts
      2. Charts are responsive by default without tweaks to the system
   2. Cons:
      1. Limited customization only very standard charts to choose from
3. **Recharts [7]**
   1. Pros:
      1. Recharts is designed specifically for react, allowing integration to be seamless
      2. Similar syntax to React making it easy to implement into custom components
      3. Reusable chart components making it very easy to manage changes
   2. Cons:
      1. Not as flexible as D3.js as there are extensive customization options but it does have a decent range of chart types
      2. Small community support due to it being a newer library
4. **Choice**

The choice of a data visualization library has not been finalized, and there remains the possibility of using multiple libraries if they better suit our needs. Since the specific types of graphs required are not yet determined, keeping options open appears to be the most practical approach for now.

### WORKS CITED

1. React. *React – A JavaScript Library for Building User Interfaces*. Meta. Accessed 12 Dec. 2024. <https://reactjs.org/>.
2. Vue.js. *Vue.js – The Progressive JavaScript Framework*. Evan You. Accessed 12 Dec. 2024. <https://vuejs.org/>.
3. Angular. *Angular – The Modern Web Developer's Platform*. Google. Accessed 12 Dec. 2024. <https://angular.io/>.
4. Tailwind CSS. *Tailwind CSS – A Utility-First CSS Framework*. Tailwind Labs. Accessed 12 Dec. 2024. <https://tailwindcss.com/>.
5. D3.js. *D3.js – Data-Driven Documents*. Mike Bostock. Accessed 12 Dec. 2024. <https://d3js.org/>.
6. Chart.js. *Chart.js – Open Source HTML5 Charts for Your Website*. Accessed 12 Dec. 2024. <https://www.chartjs.org/>.
7. Recharts. *Recharts – Redefined Chart Library Built with React and D3*. Accessed 12 Dec. 2024. <https://recharts.org/>.

## BACKEND

### INTRODUCTION

The backend of a web application is the foundation that ensures the system operates smoothly and efficiently. It handles the core functionality of the application including data processing, functionality of the application, management and communication with other systems. This enables the font end to deliver a seamless experience to the user and allows for intended functionality of the application. A well-structured backend defines the application’s performance and reliability. There are a vast number of frameworks available for backend development and when picking one it generally comes down to which one is more efficient with the current tech stack and the intended use for the application.

### PROBLEM STATEMENT

In modern applications, the backend is almost certainly needed for any dynamic or non-static website and this hockey application is no exception. The backend application plays an important role for system and component integration and communication however, developing a backend that supports both can be challenging due to the complexity of the logic of this system. This can be when dealing with file data passing and computational requests or intensive operations that rely on another system or file. Ensuring a clean modular and reliable backend is essential but poses many challenges for development due to its importance for the systems functionality, reliability and security.

### RESEARCH

### FRAMEWORKS

The backend of a web app serves as its foundation, managing data, logic and communication between the front end and the server. Selecting the right framework is important because it significantly influences the performance, scalability and efficiency. Frameworks like FastAPI are known for its performance and automation and can be ideal for handling image processing tasks and integration with machine learning models [1]. Similarly, Node.js-based frameworks like Express.js are advantageous for visualization and real time analysis and WebSocket integration [4]. The choice for the most optimal backend framework comes down to workflow and system design evaluating what is valued for the software.

### FASTAPI (PYTHON)

FastAPI framework allows for Asynchronous programming and this framework makes python reliable and fast comparable to JS frameworks. It closes the gaps for the slow performance that Python is known for and supports a vast number of libraries that enhances machine learning and computer vision integration. This language also has built in support for OpenCV and its libraries such as TensorFlow and PyTorch, and NumPy which makes using LLMs and computer vision seamless during development. This is also a newer framework that is more simplistic when compared to other Python backends such as Django and Flask which provide more tools for development but may take longer due to its vast libraries and tools to learn from. This framework is also advantageous to Flask and Django due to its speed and efficiency which can increase website efficiency and data processing in a real time environment for the user's experience when using an application built on this framework. A potential downside of using this backend framework could be the data handling from the front end, React uses node as the environment and a potential conflict could be sending and receiving data from the frontend could raise some complexities. This complexity could be solved by either using a middleware or restructuring the data while sending and receiving. Overall, there are not many disadvantages besides minor efficiency differences between using this python framework versus a JS framework. The advantages of having this python backend allows for seamless integration with the OpenCV python library and simplifies communication between the backend API routes and calls and image and data processing of the OpenCV in Python [2].

### EXPRESS.JS (NODE.JS)

This JavaScript framework is lightweight and non-centralized which makes it a great general-purpose framework. It supports asynchronous architecture for real time data streams effectively and suitable for live analysis. It is a framework of choice when integrating cross platform applications due to its support. Its main advantage over other frameworks for the application is that it is efficient and seamless when communicating with React which simplifies the API routing from the front end. This language requires middleware for basic features and requires a third-party application which limits its capabilities and requires more knowledge and research. For simple applications this framework is powerful but for specialized projects that use computer vision which requires communication with a python data processing script this could cause unnecessary complexity within the backend itself and require alternative solutions. For this project efficiency within the backend and speed is valued, this framework has the speed for data communication from the front end but does not integrate well with the Python processing logic. FastAPI seems to provide everything that express does and works more efficiently with the current use case. A potential integration with this framework could be a proxy server on top of FastAPI to receive and process data from React and send to FastAPI API routes and then to the script for processing. This seems to not work as efficiently as using pure FastAPI which simplifies the backend to Python exclusively [4].

### CHOICE

The choice for the backend implementation will be FastAPI due to its support for Python libraries and integration. This will make the Hockey web application easier to implement and maintain. This framework seamlessly integrates with the Python processing script and will simplify language compatibility issues by using Python as the logic and the routing. After researching the pros and cons of the two most popular choices using this framework is more valuable for the current use case and scope.

### WORKS CITED

1. Wikipedia contributors. *FastAPI*. Wikipedia, 12 Dec. 2024, <https://en.wikipedia.org/wiki/FastAPI>. Accessed 12 Dec. 2024.
2. GeeksforGeeks. “Flask vs FastAPI.” *GeeksforGeeks*, <https://www.geeksforgeeks.org/flask-vs-fastapi/>. Accessed 12 Dec. 2024.
3. Mozilla Developer Network. “Introduction to Django.” *MDN Web Docs*, Mozilla, <https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django/Introduction>. Accessed 12 Dec. 2024.
4. Mozilla Developer Network. “Introduction to Express/Node.js.” *MDN Web Docs*, Mozilla, <https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs/Introduction>. Accessed 12 Dec. 2024.

## COMPUTER VISION

### INTRODUCTION

OpenCV can handle computer vision tasks which will simplify the process for analyzing the data. These communications are handled through API routing logic in the backend in which data is manipulated and transferred between the two sides of the application. The library is designed to efficiently handle the complexities of video analysis in hockey; some potential tasks include the movement of players, identifying possession of the puck, and scores. The library also allows for custom trained AI models which could be implemented in alignment with the project goals to provide accurate tracking and detection. The backend focuses on the integration of the front-end application and external systems as well as providing the logic for the core concepts of the system application [2].

### PROBLEM STATEMENT

Analyzing video data is a demanding process which is a tedious time-consuming process which could be exposed to human error if not machine automated. Existing solutions often lack affordability and accessibility with a large overhead for management and organization as well as costs for corporations. The increasing demand for accurate data in the sports industry is due to the rise of its integration into other industries such sports betting, and broadcasting and news which further underscores the importance of automation and computer processing of these tasks which remains consistent for analysis. This also allows for a modular process of each process which is scalable, reusable, upgradable and maintainable.

### RESEARCH

Computer vision is a field of artificial intelligence that enables machines to interpret, analyze, and extract meaningful information from visual inputs such as images and videos and live feed. It works by using machine models as the foundation for its processing which is trained on a large dataset to process videos. Some of its functionality include object detection, image classification, facial recognition and motion detection. The field has seen significant advancements due to the convergence of improved algorithms, increased computational power and the availability of datasets for training. It is embedded in various industries for automation and analysis. There are various libraries in various languages that provide accessibility to these tools that allow for integration of computer vision into different projects [3].

1. **Pattern recognition and deep learning**

Pattern recognition is a foundational aspect of computer vision, involving the identification and categorical of patterns within visual data. In hockey analysis, pattern recognition is used to detect recurring elements such as pattern recognition is used to detect recurring elements such as movement, puck trajectory and gameplay rules such as passing or scoring. Techniques for pattern technology include supervised classification and unsupervised classification. The model is trained using labeled data, where each input is associated with a specific, predefined output label. The algorithm is designed to intake a labeled dataset which it is trained on to make predictive labels for unseen data and inputs. Common algorithms include Decision Trees, Support Vectors, Neural Networks, and k-Nearest Neighbor. These are all examples of deep learning for pattern recognition in computer vision [1].

1. **Object detection and tracking**

Object detection is an important integration within computer vision, combining classification and localization to identify and pinpoint objects in an image or video frame and space. It involves taking the position and boundaries of objects in a frame (which could be image for video) and classifying the image into different categories. It achieves this by using the models trained with the methods above to achieve object tracking and detection. This is important for the hockey analysis web application because it provides tracking of essential components of the game to calculate various statistics of interest [3].

1. **Image processing**

Image processing is an important feature of what computer vision can do, it focuses on the manipulation and analysis of visual data to enhance its quality or extract meaningful information. It serves as a critical preprocessing for applications that utilizes object detection and tracking, facial recognition or sports analytics [2].

### WORKS CITED

1. Viso.ai. “Pattern Recognition in Artificial Intelligence: Definition, Examples, Applications.” *Viso Suite AI*, <https://viso.ai/deep-learning/pattern-recognition/>. Accessed 12 Dec. 2024.
2. OpenCV. “Computer Vision and Image Processing.” *OpenCV Blog*, <https://opencv.org/blog/computer-vision-and-image-processing/>. Accessed 12 Dec. 2024.
3. Papers with Code. “Object Detection.” *Papers with Code*, <https://paperswithcode.com/task/object-detection>. Accessed 12 Dec. 2024.

**4.5 Database**

### INTRODUCTION

Selecting the appropriate database is an important decision in system design as it will directly impact performance, scalability and maintainability. This section will explore the potential database options for the project and evaluate their pros and cons. The goal of this section is to evaluate popular databases and compare it to the goals of this project and pick a suitable option according to the project's needs. The evaluation will compare data structure, query performance and integration with the system's components [1].

### PROBLEM STATEMENT

The Hockey Analysis app requires a database that can handle efficient data storage and retrieval, is scalable, and integrates well with the current tech stack and its components. The database must support structured and semi-structured data to accommodate the system’s functionality and its API operations. These requirements include, managing user information, analysis data storage, and data passing. Additionally, the database must integrate well with OpenCV and FastAPI’s API endpoints [2].

### RESEARCH

**PostgreSQL:**

PostgreSQL is an open-source relational database known for its reliability and features. It supports structured data with complex relationships, it also supports JSON format which makes it highly useful for the Hockey Analysis application system [2]. This database system is useful for applications and systems that require complex querying which is a potential downside of using it. It has a learning curve and requires knowledge to use its capabilities. However, with careful setup and optimization it is great for large datasets or files, it also has strong integration with backend frameworks like FastAPI which makes it a top choice [3].

**MySQL:**

MySQL is one of the most popular databases due to its simplicity and ease of use. It can deliver fast and high performance for different work environments and projects which makes this a well-rounded database [1]. However, MySQL lacks some advanced features that are offered in PostgreSQL, such as text indexing for JSON data. Handling JSON will be an important feature of this project. This database will be less efficient than PostgreSQL. Additionally, it can encounter performance issues in write heavy operations which can impact the performance of the applications API calls [2]. This option is a great all-around option for simple to moderate applications, but it is not a top choice for the current projects teck stack and functionality which aligns with PostgreSQL more [3].

**MongoDB:**

MongoDB is a NoSQL database designed for flexibility and simplicity and ideal for applications with semi structured or changing data models [2]. It allows users to store data in JSON like format which aligns with the project's needs and simplifies the integration with modern APIs. The downside of using MongoDB in the current application is that it lacks a strong relation between data which is important in this web app's functionality. The current Hockey Analysis application uses data stored for each user and well as data associated with each upload for the system to store. This is well along the lines of using this database but using PostgreSQL is still the safer option due to its compatibility and integration with FastAPI and its JSON support [3].

**Choice:**

From research above the decision holds and PostgreSQL is the most integrated for the current application and will be the database for the application. There is a learning curve for this database to fully utilize and depending on development timing, MongoDB is also a great option if the elicitation requirements align with the future goals of development.

### WORKS CITED

1. KeyCDN. “15 Most Popular Databases in 2023.” *KeyCDN Blog*, <https://www.keycdn.com/blog/popular-databases>. Accessed 12 Dec. 2024.
2. Amazon Web Services. “What Is PostgreSQL?” *AWS*, <https://aws.amazon.com/rds/postgresql/what-is-postgresql/>. Accessed 12 Dec. 2024.
3. Sisense Community. “Postgres vs MongoDB for Storing JSON Data: Which Should You Use?” *Sisense Knowledge Base*, <https://community.sisense.com/t5/knowledge-base/postgres-vs-mongodb-for-storing-json-data-which-should-you/ta-p/111>. Accessed 12 Dec. 2024.

# SYSTEM DESIGN

## INTRODUCTION

The design section aims to comprehensively document the methodologies and processes required to build the project. This includes a detailed breakdown of the system's architecture, workflows, and user interactions with our web application. By incorporating flowcharts, diagrams, and pseudocode, this section aims to provide a clear and precise understanding of the system's functionality and interaction between subsystems and its components. The goal is to provide precise and clear insights for developers and clients to ensure the transition from concept to implementation is smooth. Each section is structured sequentially to reflect the logical progression of system design. This section is organized sequentially starting from the high-level system architecture and progressing to detailed workflows and concepts using diagrams and pseudo code. Each component of this section will demonstrate the functional product of the web application when fully implemented.

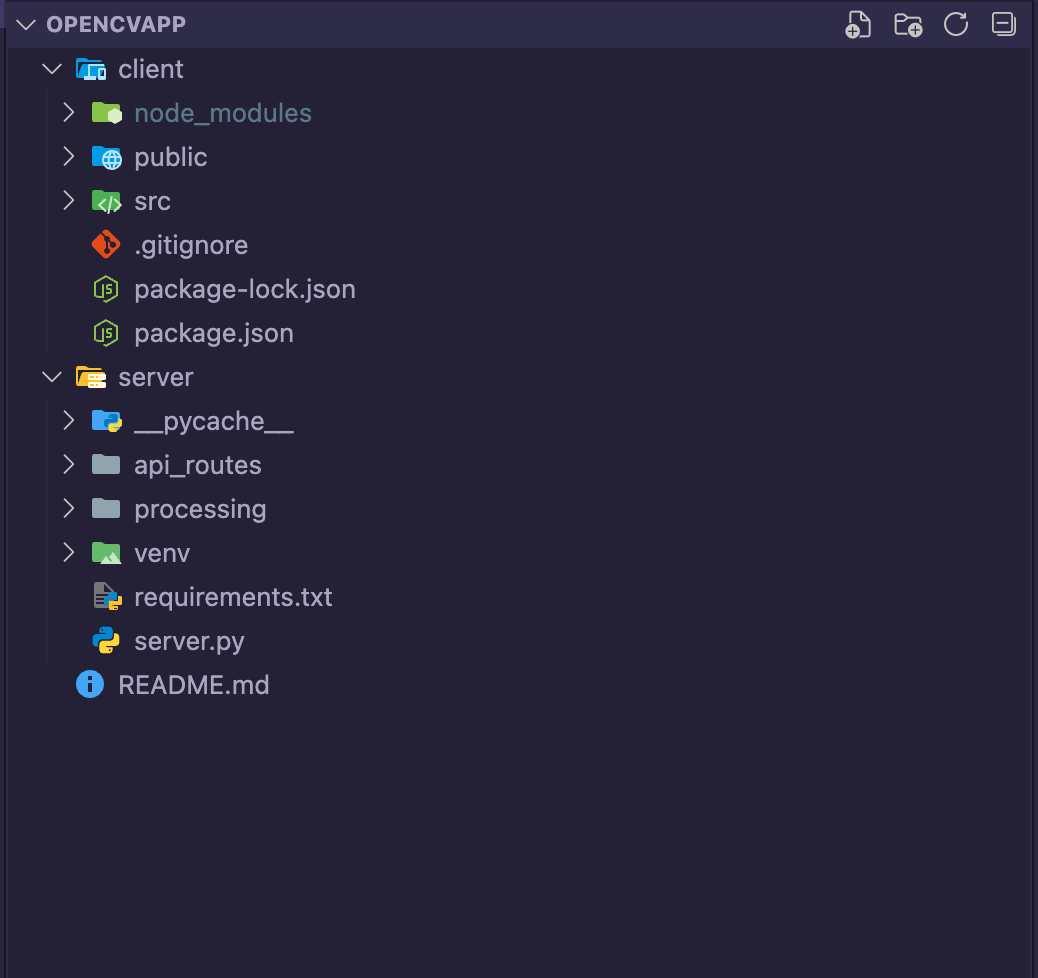
## METHODOLOGY

The software design methodology adopted for this project is system prototyping. System prototyping is the methodology of choice because it allows the web application to be built upon a prototype. The scope of this project is a semester and that gives time for improving the application and its functionality. This time frame also allows the developers to change functionality to fit the requirement elicitation of this project. This project is also a new idea as well and so as requirements change for each iteration of development the software can be tested and changed according to the current goals of the project. This methodology allows for flexibility when compared to a non-iterative approach such as waterfall. With a team of 2 the project is more efficient using an iterative approach according to the scope and time constraints. Having an early prototype allows for the creation of an early deliverable that can be improved upon with each cycle which the team values over a structured approach which can hinder the web application’s outcome.

**Modular Design:**

The system is divided into individual components which allows individual components to have its own functionality in the project. This approach method is efficient when working with the system prototyping approach because it allows for changes to the system without changing every component in the system. This can be achieved by separating the front end from the backend as well as using data structures and object-oriented programming to modularize the system’s code. Modular design also encourages the separations of concerns, and this includes the organization of the project, keeping each functionality in its own folder and or file. Using this approach is essential for the application because it allows for changes and debugging more efficiently and faster. Another characteristic of modular design is the reusability, and readability of components within the system, separating each component also makes the system readable by reusing components.

**Modular System Layout:**

****

## SYSTEM ARCHITECTURE

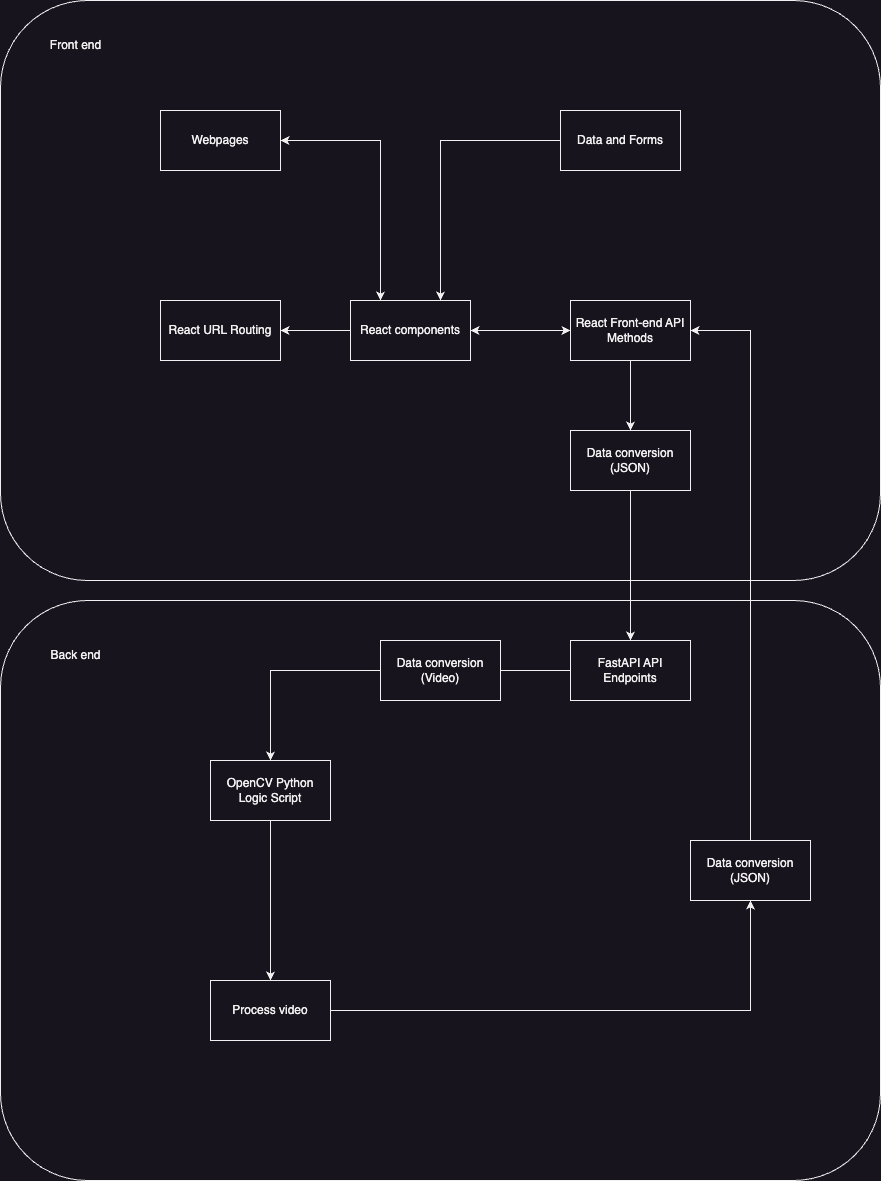
**Overview:**

This section defines the overall structure of the system, demonstrating the high-level design and interactions between different components of the system. The architecture is designed to optimize performance, scalability and maintainability. It includes both software and hardware components of the system and this ensures that the design is modular and scalable.

**Design:**

The front end of the system will be using React as a front-end framework for a responsive and interactive user experience that also allows the system to handle user input form validation and API calls to the backend. The system will also be able to handle real time updates and changes and allow for real time interactions with the system. The backend will be implemented using the FastAPI framework. This allows the system to communicate with the main logic of the system which is the data processing script using OpenCV and other libraries such as NumPy. This will also provide communication via API routes from the front end to the backend allowing data processing to be fast and efficient. The backend will also be responsible for the asynchronous integration which will significantly boost the performance of the application. A large machine model could also be used in the implementation for the application which will cater the experience for object detection and tracking for the application's use case. Below a general system architecture design of the system and how the components interact on a high level.

**System Architecture Diagram:**

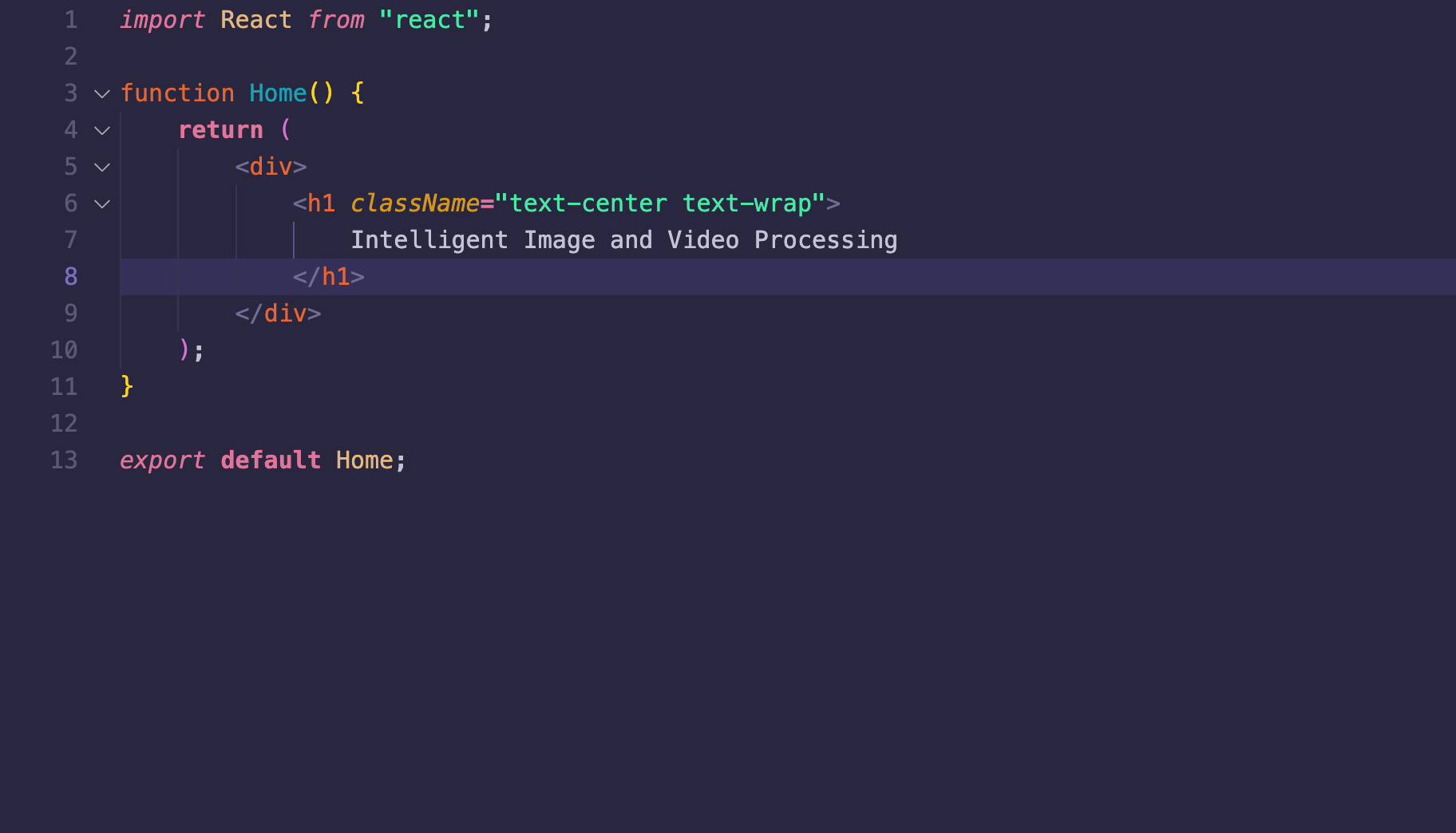
****

## PSEUDOCODE AND OTHER CODE

This section will be showing the pseudo code for the frame of the frontend and backend of the project for basic routing and connectivity. The code is designed to offer a level of understanding of the systems logic and structure. It serves to bridge the gap between conceptual design and implementation. This section will include code for both the front and the backend showing the general layout of each component and how they will connect in the system for the modular design.

* + 1. FRONTEND

**React Component:**

****

**React Routing:**

****

# 

* + 1. BACKEND

**FastAPI routing:**

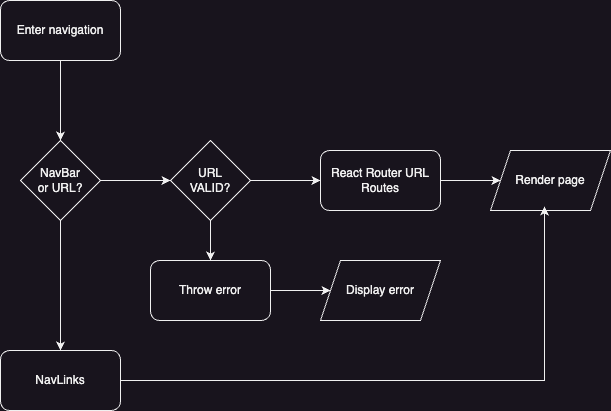
****

* + 1. FLOWCHARTS

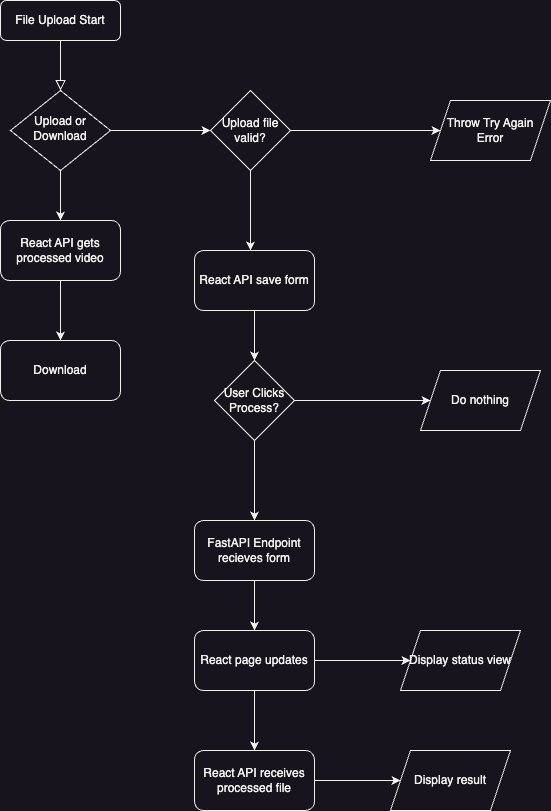
**System Flowchart**

The system flowchart provides a visual representation of the flow of the system with user interactions; it outlines the decision making of the user and how the system responds. These interactions will include basic functionality of the hockey web app such as file upload, page navigation and data feedback. The flowchart in this system will have components such as a start and end block and conditional blocks for different logic that the web app handles. Processing blocks are also used to represent the backend processing of the user input. This section is a bird’s eye view of the flow of the Hockey Analysis app for a client and user.

**Flowchart page navigation:**

****

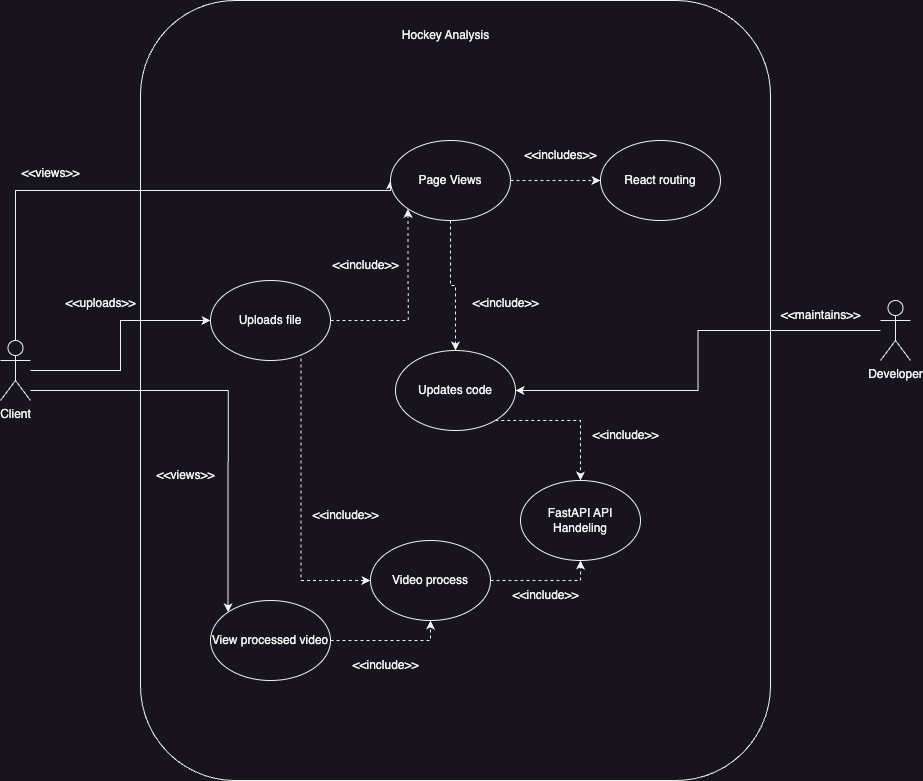
**Flowchart File Upload/Download**

****

* + 1. USE CASE

This section provides an analysis of the system’s design and functionality through a use case diagram. Use case diagrams are useful for mapping out the general functionality between a user and the system and creates a visual representation of how interactions are handled; the outermost rectangle represents the system and the circles inside represent use cases of the system and how they correlate with each other. Stick figures represent the actors and in this case the client and the app developers. Relationships are handled as lines through each use case. The include relationship represents the whole part relationship meaning if one does not exist the other will not either. This shows a strict relationship in the system that might be a potential vulnerability within the system if there are errors. The relationship between the actors is as described and they represent how the user will interact with the system. These relationships are optional and not strict because it is a potential decision that the user will have to make. The diagram below will map the core functionalities and use cases of the user and system as well as the developer.

**Use Case diagram:**

****

# PROJECT DEVELOPMENT

## DEVELOPMENT STRATEGY

In this project, there are many different components that must be seamlessly integrated to achieve the functionality in the application to match the goals. In specifics, the project must combine the computer vision algorithm with a robust backend system for holding necessary data. Simultaneously, the frontend of the application must provide an intuitive and user-friendly interface for interacting with the other components of the system. Coordinating these continuously changing elements requires a development strategy that allows for continuous testing and efficient handling of iterative development between the components. As the project evolves, additional implementations for the hockey analytics may be required, making adaptability a crucial consideration. Currently, adopting a hybrid development methodology (using system prototyping and agile) for our project seems to be the most advantageous strategy. Agile allows for decomposition to the project into manageable sprints enabling focus on certain elements of the project at a time. And system prototyping for early prototyping and iteration techniques to improve it through each cycle.

1. **Flexibility**

Agile and system prototyping allows the team to respond to any changes in project requirements and any challenges that emerge. Due to the complexity of using computer vision algorithms to track in game statistics, the amount of possible data points gathered from the hockey video could change, requiring pivots and adaptability to those changes. Agile and system prototyping’s iterative and decomposition approach ensures that development is more manageable.

1. **Sprint Delivery**

Due to the ability to break down the project into smaller, more manageable chunks, ensure critical elements of the project such as player tracking algorithms and the user interface are developed early in the development cycle. As the project progresses, refinements to each component of the project can be made as each part becomes more complex with more functionality.

## DEVELOPMENT SCHEDULE

To ensure the successful execution of the project in a timely manner, creating a well-structured timeline of the development cycle is vital. The timeline below is an example of how these things can be achieved. It outlines how the agile and system prototyping methodologies will be used, facilitating iterative development throughout each two-week sprint.

**Total Time: One Semester (16-Weeks)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Phase | Duration | Description | Estimated Start/End Date | Team Member Roles |
| Design Finalization Stage | Weeks 1-2 | Set up the development environments such as our react project and node.js server, as well as get familiar with how we will attack the hockey analytics computer vision learning. | Jan 21st - Feb 3rd | **Both -** Setup environment for both backend and frontend development so it is identical for all team members. |
| **Iterative Development Stage** | **Weeks 3-12** | **Execute 1-week and 2-week sprints focusing on our three main components we must develop.** |  |  |
| Frontend / Backend Development (1) | Weeks 3-4 | Design UI mockups, implement simple user interface, set up database schema, implement authentication, and develop the API endpoints. | Feb 4th - Feb 17th | **Jackson Morgan** - UI Development  **Tin Duong** - Initial Backend and Database Development |
| Computer Vision Algorithm (1) | Weeks 5-6 | Start training the model to our specifications. Get player tracking working. | Feb 18th - Mar 2nd | **Jackson Morgan** - Gathering Quality Video Footage and figuring our importing  **Tin Duong** - Initial algorithm creation |
| Frontend / Backend Development (2) | Weeks 7-8 | Integrate Computer vision into API, connect to the user interface and develop a simple beginner statistics dashboard. | Mar 3rd - Mar 16th | **Jackson Morgan -** Create a user dashboard for stats.  **Tin Duong -** Connect backend to frontend with API endpoints. |
| Computer Vision Algorithm (2) | Weeks 9-10 | Finalize the algorithm, optimize the tracking performance, and test on sample data. | Mar 17th - Mar 30th | **Jackson Morgan -** Analyze tracking output to ensure consistency, debug edge cases of the tracking.  **Tin Duong -** Optimize algorithm performance, improve accuracy, and validate performance. |
| Frontend / Backend Development (3) | Weeks 11-12 | Integrate final statistics and analysis into the UI dashboard and complete the full web application, finalize backend connections. | Mar 31st - April 13th | **Jackson Morgan -** Refine user dashboard by integrating the more advanced analytics and visualizations.  **Tin Duong -** Finalize backend data pipelines, making seamless integration between API and frontend. |
| Testing Stage | Weeks 13-14 | Unit Testing, Acceptance Testing, make sure the application functions as we want it to. | Apr 14th - Apr 27th | **Jackson Morgan -** Implement Black Box Testing for overall application functionality.  **Tin Duong -** Implement White Box Testing to test the individual units and integration of those parts of the system. |
| Final Adjustments Stage | Week 15 | Make final adjustments to mainly the frontend to make sure it is as user-friendly and visually appealing as possible. | April 28th - May 4th | **Jackson Morgan -** Enhance UI/UX design making it visually appealing and user friendly.  **Tin Duong -** Conduct a review of the code to ensure it is all clean and maintainable and address any backend problems. |
| Deployment | Week 16 | Deploy the application using a staging environment and make sure it functions as expected. | May 5th - May 11th | **Both - Deploy** |

## WORK DELEGATION

Effective work delegation is important in the development process to avoid duplication of efforts. With a team consisting of two members, overlapping responsibilities are dangerous so carefully handling tasks is crucial, so ensuring that skills are applied to the appropriate areas remains a priority. To optimize this process, tasks will be divided based on each member’s areas of expertise, ensuring efficient use of skills and resources.

1. **Jackson Morgan - Frontend Development**
   1. Setting up the frontend environment and building the responsive UI using the React framework. This involves making it integrated and compatible on most platforms additionally creating a visually appealing user interface.
   2. Setting up routes between the frontend and backend so that the UI can effectively communicate with other sections of the system including data management and data communication.
   3. Develop dashboards that effectively display the analytics, such as heatmaps.
2. **Tin Duong - Backend Development** 
   1. Setting up the backend environment using FastAPI and creating the APIs for user registration and authentication, as well as data manipulation and management with the database.
   2. Configuring and managing the PostgreSQL database ensuring integrity and safety for users.
3. **Both - Computer Vision**
   1. Develop and integrate a computer vision algorithm for player tracking and other analytics using OpenCV
   2. Create efficient ways to process the incoming video footage
   3. Creating algorithms for analysis calculations and handling and formatting the data in an interactive and simple interface such as chart, graph etc.

## POSSIBLE PROBLEMS

1. **Computer Vision**

This technology is new and unfamiliar, making the complexity of analyzing the footage uncertain. During the development process, ongoing testing and iteration is needed to allow adjustments to the level of detail initially planned for the analytics. If the data points extracted from the hockey video are fewer than expected, the scope of the analytics may need to be revised. Certain aspects, such as puck tracking, appear challenging, and the implementation strategy for that component may be subject to change.

1. **Multi-Platform Compatibility**

The goal is to ensure the application is accessible from any operating system and browser by implementing a responsive frontend design. This approach presents challenges, as certain browsers may lack support for specific CSS libraries. As development progresses, it may become clear that some browsers are less compliant with the project, potentially complicating adjustments for these cases.

## TESTING PLAN (FOR FUTURE DEVELOPMENT)