

Module Guide for SFWRENG 4G06 Capstone Design Project

Team #18, InfiniView-AI

Anhao Jiao

Kehao Huang

Qianlin Chen

Qi Shu

Xunzhou Ye

January 19, 2024

1 Revision History

Date	Version	Notes
Jan 13, 2024	1.0	Initial Draft
Jan 14 2024	1.1	Revisit after MIS module decomposition
Jan 17 2024	1.2	Add abbreviations section

2 Reference Material

This section records information for easy reference.

2.1 Abbreviations and Acronyms

symbol	description
AC	Anticipated Change
UC	Unlikely Change
M	Module
SRS	Software Requirements Specification
HA	Hazard Analysis
VnVPlan	Verification and Validation Plan
MIS	Module Interface Specification
MVC	Model-view-controller, a well-known software design pattern
FR	Functional Requirement
LF	Look and Feel Requirement
PR	Performance Requirement
OE	Operational and Environmental Requirements
HS	Health and Safety Requirements
MS	Maintainability and Support Requirement
SR	Security Requirement
CR	Cultural Requirement
LR	Legal Requirement
HTTP	Hypertext Transfer Protocol
OS	Operating System
STUN	Session Traversal Utilities for NAT - a type of server needed for setting up peer-to-peer
RTC	Real-Time Communication
SFU	Selective Forwarding Unit - A software unit that can selectively forward video streams
UI	User Interface
ML	Machine Learning
HLS	HTTP Live Streaming
MPEG-DASH	Dynamic Adaptive Streaming over HTTP
DAG	Directed Acyclic Graph

Contents

1	Revision History	i
2	Reference Material	ii
2.1	Abbreviations and Acronyms	ii
3	Introduction	1
3.1	Overview	1
3.2	Purpose	1
3.3	Design Principles	1
3.4	Design Pattern	1
4	Anticipated and Unlikely Changes	2
4.1	Anticipated Changes	2
4.2	Unlikely Changes	3
5	Module Hierarchy	4
6	Connection Between Requirements and Design	5
7	Module Decomposition	5
7.1	Hardware Hiding Modules	5
7.1.1	Media Control Module (M1)	6
7.2	Behaviour-Hiding Module	6
7.2.1	User Authentication Module (M2)	6
7.2.2	Instructor View Module (M3)	6
7.2.3	Practitioner View Module (M4)	7
7.2.4	Annotation Configuration Module (M5)	7
7.2.5	RTC Control Module (M7)	7
7.3	Software Decision Module	7
7.3.1	APP Module (M8)	8
7.3.2	Video Transform Module (M9)	8
7.3.3	Human Pose Estimation Annotation Module (M10)	8
7.3.4	Center of Mass Module (M11)	8
7.3.5	SFU Server Module (M12)	9
7.3.6	STUN Server Module (M6)	9
8	Traceability Matrix	9
9	Use Hierarchy Between Modules	12
10	Timeline	13

11 Reflection	15
11.1 Qi Shu	15
11.2 Kehao Huang	15
11.3 Anhao Jiao	16
11.4 Qianlin Chen	16
11.5 Xunzhou Ye	17

List of Tables

1	Module Hierarchy	5
2	Trace Between Requirements and Modules	10
3	Trace Between Requirements and Modules	10
4	Trace Between Non-functional Requirements and Modules	11
5	Trace Between Anticipated Changes and Modules	12

List of Figures

1	Use hierarchy among modules	13
---	---------------------------------------	----

3 Introduction

This Module Guide Document serves as a design blueprint for MotionMingle - a WebRTC-based video conferencing application designed to enrich the experience of Tai Chi instruction. It presents a modular perspective of the project, guiding the development team in creating a platform that emphasizes real-time interaction, minimal hardware requirements, and ease of use.

3.1 Overview

This Module Guide Document serves as a design blueprint for MotionMingle - a WebRTC-based video conferencing application designed to enrich the experience of Tai Chi instruction. It presents a modular perspective of the project, guiding the development team in creating a platform that emphasizes real-time interaction, minimal hardware requirements, and ease of use.

The document aligns with standards outlined in various project documents, including the SRS, HA, VnVPlan, SystDes, and MIS.

SRS - /docs/SRS/SRS.pdf

HA - /docs/HazardAnalysis/HazardAnalysis.pdf

VnVPlan - /docs/VnVPlan/VnVPlan.pdf

MIS - /docs/Design/SoftDetailedDes/MIS.pdf

3.2 Purpose

The aim of this Module Guide Document is to detail the architecture of modules, based on selected design principles and patterns, to clarify the project's functionalities and the specific roles of each module.

3.3 Design Principles

The Module Guide Document employs principles like information hiding, high cohesion, low coupling, high fan-in, and low fan-out for breaking down modules. These principles involve identifying and securing anticipated changes.

3.4 Design Pattern

The project utilizes the MVC design pattern, facilitating the division of complex problems into manageable sub tasks across different modules, adhering to the separation of concerns

principle.

4 Anticipated and Unlikely Changes

This section lists possible changes to the system. According to the likeliness of the change, the possible changes are classified into two categories. Anticipated changes are listed in Section 4.1, and unlikely changes are listed in Section 4.2.

4.1 Anticipated Changes

Anticipated changes are the source of the information that is to be hidden inside the modules. Ideally, changing one of the anticipated changes will only require changing the one module that hides the associated decision. The approach adopted here is called design for change.

AC1: Video stream sharing by practitioners might be allowed for better user communications.

AC2: Audio stream sharing by practitioners might be allowed for better user communications.

AC3: The number of annotation types available for users to choose from might change as discovered through the development process.

AC4: The initial release of the application has a limited load capacity, which will need to be expanded as the application's popularity grows to accommodate a growing number of users.

AC5: The exactness of annotations is subject to change, given the lack of definitive measurement standards.

AC6: The application may be extended to offer native client implementation.

AC7: The application may be extended to offer mobile client implementation.

AC8: The application could undergo enhancements with new features based on real-world usage insights.

AC9: The application might incorporate an account login system to verify the identity of instructors.

4.2 Unlikely Changes

The module design should be as general as possible. However, a general system is more complex. Sometimes this complexity is not necessary. Fixing some design decisions at the system architecture stage can simplify the software design. If these decision should later need to be changed, then many parts of the design will potentially need to be modified. Hence, it is not intended that these decisions will be changed.

UC1: The need for specific input/output devices is expected to remain stable, given the application's primary function is to augment the instructor's video stream with annotations.

UC2: The necessity for a stable internet connection should persist, as the application depends on HTTP requests for communication between server and client.

UC3: The application's ease of use is anticipated to remain a constant requirement, catering to its primary user base of elderly individuals.

UC4: The requirement for the application to be easy to learn is expected to stay unchanged, considering its elderly user demographic.

UC5: The demand for server reliability and availability is projected to be consistent, due to its critical role in the application's functionality.

UC6: The system is intended to be compatible with the latest versions of Windows, Linux, and macOS.

UC7: Browser compatibility is a designed feature of the system.

UC8: Unauthorized modifications to the application are to be prevented.

UC9: The fundamental feature of adding annotations to the video stream is not expected to change.

UC10: The application's design, ensuring compliance with relevant laws, is foreseen to remain unaltered.

UC11: The client for this project is not expected to change, as it is designed as a student capstone project with limited market potential.

UC12: The stakeholder for this project is unlikely to change, as it is a student capstone project adhering to specific course guidelines.

UC13: The project's schedule and budget constraints are anticipated to stay fixed, in line with the requirements of the student capstone course.

5 Module Hierarchy

This section provides an overview of the module design. Modules are summarized in a hierarchy decomposed by secrets in Table 1. The modules listed below, which are leaves in the hierarchy tree, are the modules that will actually be implemented.

M1: Media Control Module

M2: User Authentication Module

M3: Instructor View Module

M4: Practitioner View Module

M5: Annotation Configuration Module

M6: STUN Server Module

M7: RTC Control Module

M8: App Module

M9: Video Transform Module

M10: Human Pose Estimation Annotation Module

M11: Center of Mass Annotation Module

M12: SFU Server Module

Level 1	Level 2
Hardware-Hiding Module	M13
	M2
	M3
	M4
Behaviour-Hiding Module	M5
	M7
	M6
Software Decision Module	M8
	M9
	M10
	M11
	M12

Table 1: Module Hierarchy

6 Connection Between Requirements and Design

The design of the system is intended to satisfy the requirements developed in the SRS. In this stage, the system is decomposed into modules. The connection between requirements and modules is listed in Table 3.

7 Module Decomposition

Modules are decomposed according to the principle of “information hiding” proposed by ?. The *Secrets* field in a module decomposition is a brief statement of the design decision hidden by the module. The *Services* field specifies *what* the module will do without documenting *how* to do it. For each module, a suggestion for the implementing software is given under the *Implemented By* title. If the entry is *OS*, this means that the module is provided by the operating system or by standard programming language libraries. *SFWRENG 4G06 Capstone Design Project* means the module will be implemented by the SFWRENG 4G06 Capstone Design Project software.

Only the leaf modules in the hierarchy have to be implemented. If a dash (–) is shown, this means that the module is not a leaf and will not have to be implemented.

7.1 Hardware Hiding Modules

Secrets: The data structure and algorithm used to implement the virtual hardware.

Services: Serves as a virtual hardware used by the rest of the system. This module provides the interface between the hardware and the software. So, the system can use it to display outputs or to accept inputs.

Implemented By: OS

7.1.1 Media Control Module (M1)

Secrets: The implementation for capturing camera and microphone input from clients.

Services: Enables users to turn on/off camera and microphone to upload audio and video.

Implemented By: JavaScript, WebAPI

Type of Module: Abstract Data Type

7.2 Behaviour-Hiding Module

Secrets: The contents of the required behaviours.

Services: Includes programs that provide externally visible behaviour of the system as specified in the software requirements specification (SRS) documents. This module serves as a communication layer between the hardware-hiding module and the software decision module. The programs in this module will need to change if there are changes in the SRS.

Implemented By: –

7.2.1 User Authentication Module (M2)

Secrets: The authentication mechanism and user credentials storage format.

Services: Authenticates users to grant access to the system based on credentials.

Implemented By: JavaScript, React

Type of Module: Abstract Object

7.2.2 Instructor View Module (M3)

Secrets: Instruction delivery and progress tracking mechanisms for instructors.

Services: Provides instructors with tools to manage course streaming, and interact with participants.

Implemented By: JavaScript, React

Type of Module: Abstract Object

7.2.3 Practitioner View Module (M4)

Secrets: User interface customization based on practitioner’s preferences.

Services: Allows practitioners to view available annotations, and select preferred annotations.

Implemented By: JavaScript, React

Type of Module: Abstract Object

7.2.4 Annotation Configuration Module (M5)

Secrets: The configuration options for annotation types and parameters.

Services: Enables users to select and configure various annotation types for personalized course delivery.

Implemented By: JavaScript, React

Type of Module: Abstract Data Type

7.2.5 RTC Control Module (M7)

Secrets: The peer-to-peer connection mechanism and connection between the client and the server.

Services: Provides other modules the ability to manage WebRTC connections.

Implemented By: JavaScript, WebAPI

Type of Module: Abstract Data Type

7.3 Software Decision Module

Secrets: The design decision based on mathematical theorems, physical facts, or programming considerations. The secrets of this module are *not* described in the SRS.

Services: Includes data structure and algorithms used in the system that do not provide direct interaction with the user.

Implemented By: –

7.3.1 APP Module (M8)

Secrets: The business logic that coordinates the flow of data between modules.

Services: Manages application state and serves as the central communication hub for other modules.

Implemented By: JavaScript, React

Type of Module: Abstract Object

7.3.2 Video Transform Module (M9)

Secrets: Algorithms for video encoding, decoding, and transformation.

Services: Process video streams to apply transformations such as scaling, rotating, and overlaying annotations.

Implemented By: Python

Type of Module: Library

7.3.3 Human Pose Estimation Annotation Module (M10)

Secrets: The algorithms and data structures used to calculate and represent the skeletal structure of a human figure in video streams.

Services: Processes video data to overlay a representation of the human skeleton, aiding in the correction and learning of Tai Chi movements.

Implemented By: Python, opencv-python

Type of Module: Abstract Object

7.3.4 Center of Mass Module (M11)

Secrets: The methods used to determine the center of mass in a given pose or series of movements.

Services: Calculates and displays the center of mass in the video feed to help practitioners understand and improve their balance in various Tai Chi poses.

Implemented By: Python, opencv-python

Type of Module: Abstract Object

7.3.5 SFU Server Module (M12)

Secrets: The implementation of the Selective Forwarding Unit (SFU) for handling media streams.

Services: Facilitates the routing of media streams between users to support multi-party video conferencing.

Implemented By: Python, aiortc

Type of Module: Abstract Object

7.3.6 STUN Server Module (M6)

Secrets: Network traversal techniques and session negotiation mechanisms.

Services: Assists in NAT traversal by resolving public IP addresses and enabling peer-to-peer connections.

Implemented By: Implemented externally

Type of Module: Abstract Object

8 Traceability Matrix

This section shows two traceability matrices: between the modules and the requirements and between the modules and the anticipated changes.

Functional Re- quirements	Modules
FR1	M1, M2, M3
FR2	M1, M2, M4
FR3	M1, M2, M3, M4
FR4	M1, M2, M6, M8,
FR5	M1, M2, M6, M8,
FR6	M1, M2, M5
FR7	M1, M9
FR8	M1, M10, M12
FR9	M1, M12
FR10	M1, M12
FR11	M1, M9, M10, M11, M12
FR12	M1, M6, M8, M9, M12
FR13	M1, M6, M8

Table 2: Trace Between Requirements and Modules

Functional Re- quirements	Modules
LF1	M1, M2, M3
UH1	M1, M2, M4
UH2	M1, M2, M3, M4
PR1	M1, M2, M6, M8,
PR2	M1, M2, M6, M8,
PR3	M1, M2, M5
PR4	M1, M9
PR5	M1, M10, M12
PR6	M1, M12
PR7	M1, M12
PR8	M1, M9, M10, M11, M12
PR9	M1, M6, M8, M9, M12
PR10	M1, M6, M8

Table 3: Trace Between Requirements and Modules

Table 4: Trace Between Non-functional Requirements and Modules

Non-functional requirements	Re- Modules
LF1	M3, M4
UH1	M3, M4
UH2	M3, M4
PR1	M3, M4, M8
PR2	M7, M12, M6
PR3	M7, M12, M6
PR4	Everything
PR5	M7
PR6	M8
PR7	M3, M4
PR8	M6
PR9	M1
PR10	M3, M4
PR11	M3, M4
PR12	M5, M9, M10, M11
PR13	M1, M3, M4
PR14	M11, M12
PR15	M3, M4
OE1	M1
OE2	M1
OE3	M1
HS1	M1, M8
HS2	M3, M4
MS1	M1
MS2	M1
SR1	M2
SR2	M2, M3, M4
SR3	M2
SR4	M3, M4

Continued on next page

Table 4 continued from previous page

Non-functional quirements	Re-	Modules
SR5		M3, M4
SR6		M3, M4
CR1		Everything
LR1		Everything
AC1		M3, M4, M9
AC2		M3, M4, M9
AC3		M5, M8
AC4		M8
AC5		M7, M8
AC6		M3, M4
AC7		M3, M4
AC8		M7M
AC9		M2

Anticipated Change	Module
AC1	M3, M4, M9
AC2	M3, M4, M9
AC3	M5, M8
AC4	M8
AC5	M7, M8
AC6	M3, M4
AC7	M3, M4
AC8	M7M
AC9	M2

Table 5: Trace Between Anticipated Changes and Modules

9 Use Hierarchy Between Modules

In this section, the uses hierarchy between modules is provided. ? said of two programs A and B that A *uses* B if correct execution of B may be necessary for A to complete the task described in its specification. That is, A *uses* B if there exist situations in which the correct

functioning of A depends upon the availability of a correct implementation of B. Figure 1 illustrates the use relation between the modules. It can be seen that the graph is a directed acyclic graph (DAG). Each level of the hierarchy offers a testable and usable subset of the system, and modules in the higher level of the hierarchy are essentially simpler because they use modules from the lower levels.

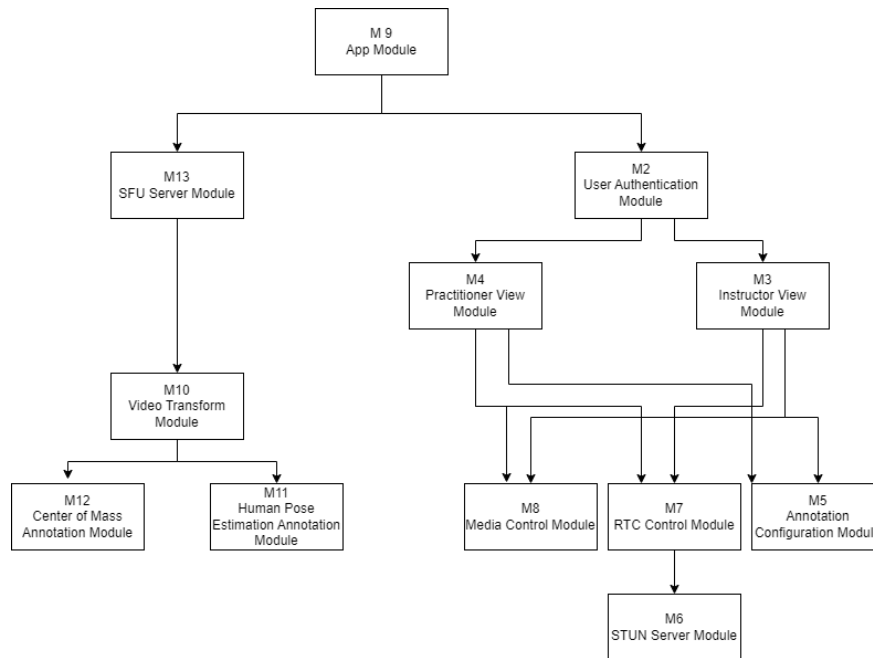


Figure 1: Use hierarchy among modules

10 Timeine

- **Week 1 - 2:**

- M2: User Authentication Module
 - * Research and design (Qi)
 - * Implementation (Qi)
- M9: App Module
 - * Initial setup and framework implementation (Kehao)
 - * Define interfaces for other modules (Kehao)

- **Week 3 - 4:**

- M13: SFU Server Module
 - * Setup and configuration (Xunzhou)
 - * Integration with App Module (Kehao)

- M6: STUN Server Module
 - * Research STUN protocols (Qianlin, Kehao)
 - * Implementation (Kehao, Qianlin)
- **Week 5:**
 - M10: Video Transform Module
 - * Research on video transformation techniques (Qianlin, Xunzhou, Anhao)
 - * Initial implementation (Qianlin, Xunzhou, Anhao)
- **Week 6 - 7:**
 - M3: Instructor View Module
 - * UI design (Qi)
 - * Backend integration (Qianlin, Xunzhou, Anhao)
 - M4: Practitioner View Module
 - * UI design (Qi)
 - * Backend integration (Qianlin, Xunzhou, Anhao)
- **Week 8 - 9:**
 - M7: RTC Control Module
 - * Implement control logic (Kehao, Xunzhou)
 - * Integrate with SFU and STUN modules (Kehao, Xunzhou)
 - M8: Media Control Module
 - * Develop media management logic (Xunzhou, Anhao)
 - * Integration with Video Transform Module (Xunzhou, Anhao)
- **Week 10 - 11:**
 - M11: Human Pose Estimation Annotation Module
 - * Research and select pose estimation technologies (Qianlin, Kehao)
 - * Implement annotation logic (Qianlin, Kehao)
 - M12: Center of Mass Annotation Module
 - * Algorithm design for center of mass calculation (Qianlin, Xunzhou)
 - * Integration with Human Pose Estimation (Qianlin, Xunzhou)
- **Week 12 - 13:**
 - M5: Annotation Configuration Module
 - * Develop configuration options (Qi, Anhao)

- * Implement dynamic changes (Qi, Anhao)
- Testing and Refinement
 - * Integration testing of all modules (Qi, Anhao, Kehao, Qianlin, Xunzhou)
 - * Performance optimization and refinement based on tests (All team members)

11 Reflection

The information in this section will be used to evaluate the team members on the graduate attribute of Problem Analysis and Design. Please answer the following questions:

1. What are the limitations of your solution? Put another way, given unlimited resources, what could you do to make the project better? (LO_ProbSolutions)
2. Give a brief overview of other design solutions you considered. What are the benefits and tradeoffs of those other designs compared with the chosen design? From all the potential options, why did you select documented design? (LO_Explores)

11.1 Qi Shu

I am tasked with the development and design of two client-facing pages, as well as managing the connectivity between the client and server. Our current approach is shaped by budget and time considerations.

One drawback of our current strategy, which involves using Figma for UI design and React for implementation, is that its performance falls short compared to native solutions and lack of experienced designers. With more resources at our disposal, my team and I could develop a high-performance native client application with a responsive UI. This would also allow us to conduct thorough user testing to obtain crucial feedback for enhancing the user experience.

Moreover, we could explore additional design tools and technologies to improve the application's visual appeal and functionality, such as adding animations and interactive features. As for UI implementation, our experience with different UI tools is limited, guiding our current design choices. In terms of client-server interaction, we considered HTTP live streaming as an alternative to WebRTC due to its popularity and established status in streaming services. However, we opted for WebRTC for its ability to facilitate peer-to-peer connections and integrate client video streams, offering greater flexibility.

11.2 Kehao Huang

I was responsible for designing the front-end portion of the system, as well as investigating the integration of machine learning models into the video transform module. Regarding our documented design, our choice to use JavaScript together with React was driven by practicality, yet I am aware that this combination might not yield the same performance

as a native video-conferencing application. With a more generous budget and timeline, I envision developing a native solution that could significantly enhance responsiveness and allow for more intensive user traffic.

Another limitation of our current design is the range of choices for annotations. As of right now, we are planning to provide two types of annotations: center of mass annotation and human pose estimation annotation. Given sufficient time, the system is likely to support a larger variety of annotation types by utilizing more existing machine learning libraries or training our own models.

11.3 Anhao Jiao

In our project integrating a machine learning model with a video stream, the primary limitations included constrained computational resources, which affected model complexity and processing speed, and challenges in model accuracy, scalability, and maintaining data privacy. With unlimited resources, we could enhance computational capabilities for advanced model deployment, develop more tailored and robust models, ensure scalable infrastructure for high-volume traffic, and implement superior data security measures.

Cloud-based processing has benefits such as powerful server-side processing, scalability, and potentially more advanced AI capabilities. The tradeoffs are higher latency, increased dependence on stable internet connectivity, and privacy concerns due to data transmission over the cloud.

The choice of using sockets and focusing on inter-module data efficiency was driven by the specific requirements of the project. The need for real-time processing, coupled with the necessity for different modules to interact seamlessly and use shared data effectively, made this design the most suitable. It offers a balanced approach, optimizing for immediate data processing and action, while also ensuring that computational resources are used efficiently by avoiding redundant data processing across modules.

11.4 Qianlin Chen

I was responsible for developing and designing the SFU module and the video transform module, key components for controlling and transforming video streams in our server system.

One limitation of our current strategy, centered around using Python with libraries like `aiortc` and `OpenCV`, is that it may not be as optimized for performance as other potential solutions. This choice was influenced by our familiarity with these tools and the immediate need to meet project deadlines. Given more resources, I would explore the integration of more performance-intensive languages like C++ for critical processing tasks. This would significantly enhance the efficiency and scalability of our modules, especially under high-load conditions.

Additionally, while our current implementation supports basic video transformations, with additional resources, I would delve into more advanced video processing techniques. This could include AI-driven transformations and augmented reality features, which would greatly enrich the user experience and broaden the application's capabilities. Another area

for improvement is comprehensive testing under varied network conditions and diverse user scenarios. With more resources, implementing a thorough testing framework and conducting extensive user testing would provide invaluable insights, allowing us to fine-tune our modules for better performance and user satisfaction.

Exploring alternative technologies and protocols was also a crucial part of my role. While WebRTC was ultimately chosen for its real-time capabilities and suitability for our project's needs, alternatives like HLS or MPEG-DASH were considered. These protocols offer advantages in compatibility and established use in streaming services but don't match WebRTC's low latency and interactive capabilities.

11.5 Xunzhou Ye

The decision to write the backend server in Python raises concerns about efficiency, latency, scalability, and concurrent usage. Python, while versatile and widely used, may not be the optimal choice for real-time applications requiring low latency and high scalability. In hindsight, using a more performance-oriented language like C++ or Go might have been considered to address these concerns. On the other hand, the choice of Python for the backend is justified by the convenience it offers in integrating the web server with ML processing. Python has a rich ecosystem of machine learning libraries and frameworks, making it a natural fit for incorporating ML models into the platform. This decision prioritizes ease of development and integration, emphasizing the importance of seamless collaboration between the web server and ML components.

Considering alternative options, the exploration of JavaScript with Node.js for the host server introduces its own set of benefits and trade-offs. While Node.js is known for its event-driven, non-blocking architecture, which can enhance scalability, it may fall short when integrating with ML models written in Python. This limitation arises from the lack of native support for certain ML libraries in JavaScript, potentially hindering the seamless integration of ML-generated annotations.