CMPE 246 Computer Engineering Design Studio

Mid-term Proposal

Advanced Project

Project-Group #: ***4***

Project-Name: PiEdge Cloud: A Raspberry Pi Cluster for Real-Time Distributed Video Processing and Web Streaming

Project-Team:

Student Name	**Student Number**	**LAB**
Patrick Agnimel	87760591	L2C
Johan Gonzaga	12977609	L2C
Syed Saad Ali	91863621	L2B
Michael Zhang	80138845	L2A
Algernon Ren	45054053	L2C
Jia Yi Lu	47866033	L2A

The University of British Columbia Okanagan
February 27, 2025

Title: PiEdge Cloud: A Raspberry Pi Cluster for Real-Time Distributed Video Processing and Web Streaming

I. Introduction

In recent years, both industry and academia have migrated their applications to cloud based systems. Cloud computing is a model in which users access computing resources on a pay-as-you-go basis over the internet, with these resources provided by large, remote data centers. While the cloud offers significant benefits—such as increased flexibility, efficiency, and strategic value—the ongoing costs can be prohibitive for small businesses, students, and academic institutions. In contrast, Single Board Computers (SBCs) like the Raspberry Pi offer a low-cost, portable alternative that deliver computing and storage capabilities close to the data source. Although SBCs are not as powerful as traditional servers, they can be clustered to replicate many of the functionalities of conventional data centers, creating innovative and cost-efficient solutions for both cloud and edge computing applications.
In today's digital landscape, the demand for agile and cost-effective computing solutions has grown exponentially in response to the deployment of increasingly advanced Internet of Things (IoT) devices, particularly at the network edge. This proposal introduces an innovative approach that leverages clusters of single board computers—specifically, Raspberry Pi clusters—to create portable and affordable cloud platforms. By exploring the distributed computing capabilities of these clusters, we aim to develop a system that supports advanced edge computing applications alongside robust web services.
The proposal is structured as follows: an introduction outlining the problem and our proposed solution; a motivation section describing the background and relevance of the issue; a project summary outlining our overall objectives; and a project details section covering the architecture and environment, timeline, resources used, design patterns, system design, and personal reflections.

II. Motivation

The rapid proliferation of IoT devices and an increasing demand for edge computing have highlighted the limitations of traditional, centralized data centers. These environments often struggle with bandwidth and latency issues when processing real-time data near its source. Additionally, high costs and infrastructure complexities further hinder the deployment of sophisticated applications in resource-constrained settings. These challenges become even more pronounced during high-traffic events or when transferring large volumes of data. The problem is particularly evident in areas with a high concentration of IoT devices or where network infrastructure is inadequate. Although modern cloud repositories can help alleviate some of these constraints by enabling devices to connect and process tasks alongside the cloud, the overall quality of service is often suboptimal due to network latency and occasional disruptions [1]. Low-cost embedded platforms can support the development of budget sensors that can provide diverse types of data, thereby supporting various projects within the IoT sphere [2].

III. Project Summary

This project aims to develop an innovative, cost-effective cloud platform by leveraging a Raspberry Pi cluster to support real-time distributed computing and video streaming applications. Our objectives include:

· Building a Scalable Cluster:
We will assemble a Raspberry Pi cluster consisting of 8 nodes, each equipped with the necessary bandwidth and networking capabilities to support high-speed data transfer and processing.

· Developing a Wireless Video Streaming System:
A separate system will be built using a Raspberry Pi integrated with a camera module to capture and wirelessly stream video. This standalone setup will serve as the primary source of video data.

· Implementing Distributed Machine Learning:
The Raspberry Pi cluster will process the incoming video stream in real time by employing distributed machine learning techniques. This will demonstrate the cluster's ability to handle complex, time-sensitive computations across multiple nodes.

· Creating a Web Streaming Application:
Building on our experience with real-time streaming protocols, we will develop a web streaming application that offers live video streaming and messaging capabilities. This application will be accessible to users via the internet.

· Ensuring Robust Hosting and Load Balancing:
The final web application will be hosted on the Raspberry Pi cluster. We will implement best practices for dynamic load balancing to ensure optimal performance and reliability, even under varying network conditions and user loads.

IV. Project Details

A. Architecture and Environment

1. System Architecture

This project is divided into three integrated components:

· **Raspberry Pi Cluster:**
A distributed computing environment formed by multiple Raspberry Pi 4 units. One unit serves as the master node, while the others function as compute nodes. These nodes are interconnected via a Gigabit network switch to ensure high-speed communication.

Scalability:
The cluster is designed for easy expansion, allowing additional compute nodes to be added with minimal configuration. This modular design supports flexible scaling as processing demands increase.

Robust Orchestration:
The master node runs a lightweight Kubernetes distribution (K3s) that implements essential control plane functions—such as scheduling, resource management, and health monitoring—to ensure the efficient deployment and management of containerized applications across the worker nodes. Moreover, K3s provides the flexibility to extend or replace the default scheduler with custom scheduling code and algorithms, thereby allowing the scheduling process to be tailored to the specific requirements of the project.

Optimized Environment:
Each node operates on Raspberry Pi OS Lite, a minimal yet stable operating system that maximizes hardware efficiency. The integration of K3s facilitates seamless management of containers (packaged into pods), which are deployed to execute the application workloads.

High-Performance Networking:
The use of a Gigabit network switch guarantees fast, reliable connectivity between nodes, which is critical for maintaining the responsiveness of distributed applications—particularly in real-time processing and edge computing scenarios.

· **Video Capture System + Machine Learning Module:**
A dedicated video capture system using a Raspberry Pi 4 unit paired with a Raspberry Pi Camera Module 3 captures live video. The live video feed is transmitted wirelessly to the cluster using real-time streaming protocols and processed in real time by distributed machine learning algorithms deployed across the cluster.

- **Live Streaming Web Application:**

A web application offering real-time video streaming and messaging services. This application is built using Node.js and Next.js (incorporating React and TypeScript) and is hosted on the Raspberry Pi cluster with dynamic load balancing to maintain optimal performance and reliability.

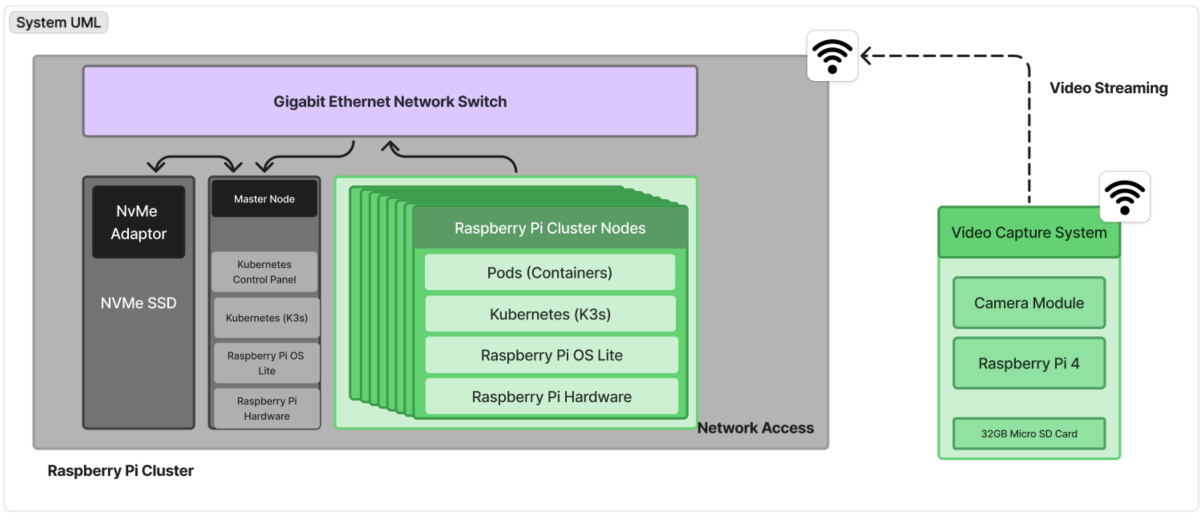

Figure 1: PiEdge Cloud Architecture - Interaction with Video Capture System

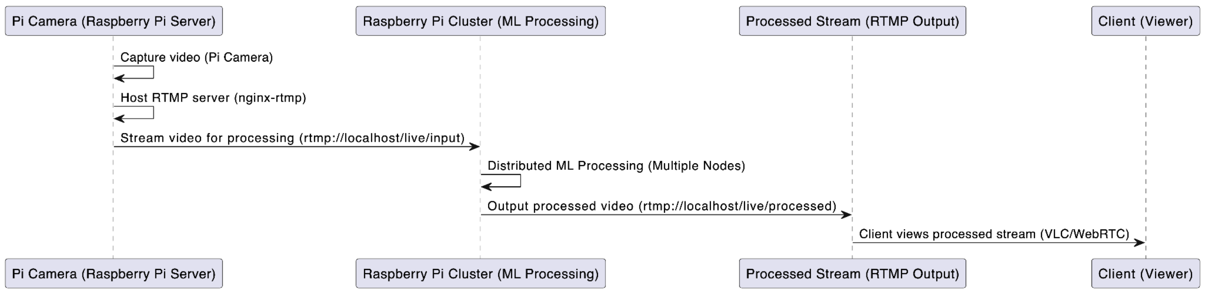
Pi Camera (Raspberry Pi Server)
Raspberry Pi Cluster (ML Processing)
Processed Stream (RTMP Output)
Client (Viewer)
Capture video (Pi Camera)
Host RTMP server (nginx-rtmp)
Stream video for processing (rtmp://localhost/live/input)
Distributed ML Processing (Multiple Nodes)
Output processed video (rtmp://localhost/live/processed)
Client views processed stream (VLC/WebRTC)
Pi Camera (Raspberry Pi Server)
Raspberry Pi Cluster (ML Processing)
Processed Stream (RTMP Output)
Client (Viewer)

Figure 2: Sequence Diagram for Video Capture System + Machine Learning Module

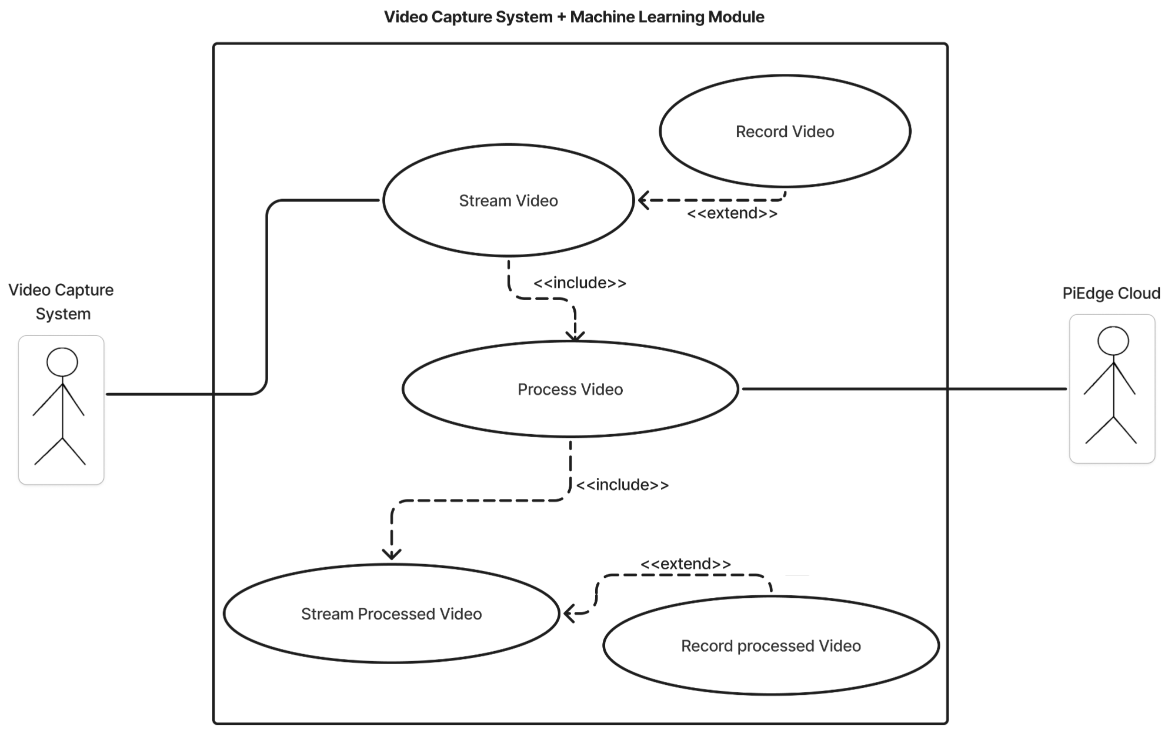

Figure 3: Use Case Diagram for Video Capture System + Machine Learning Module

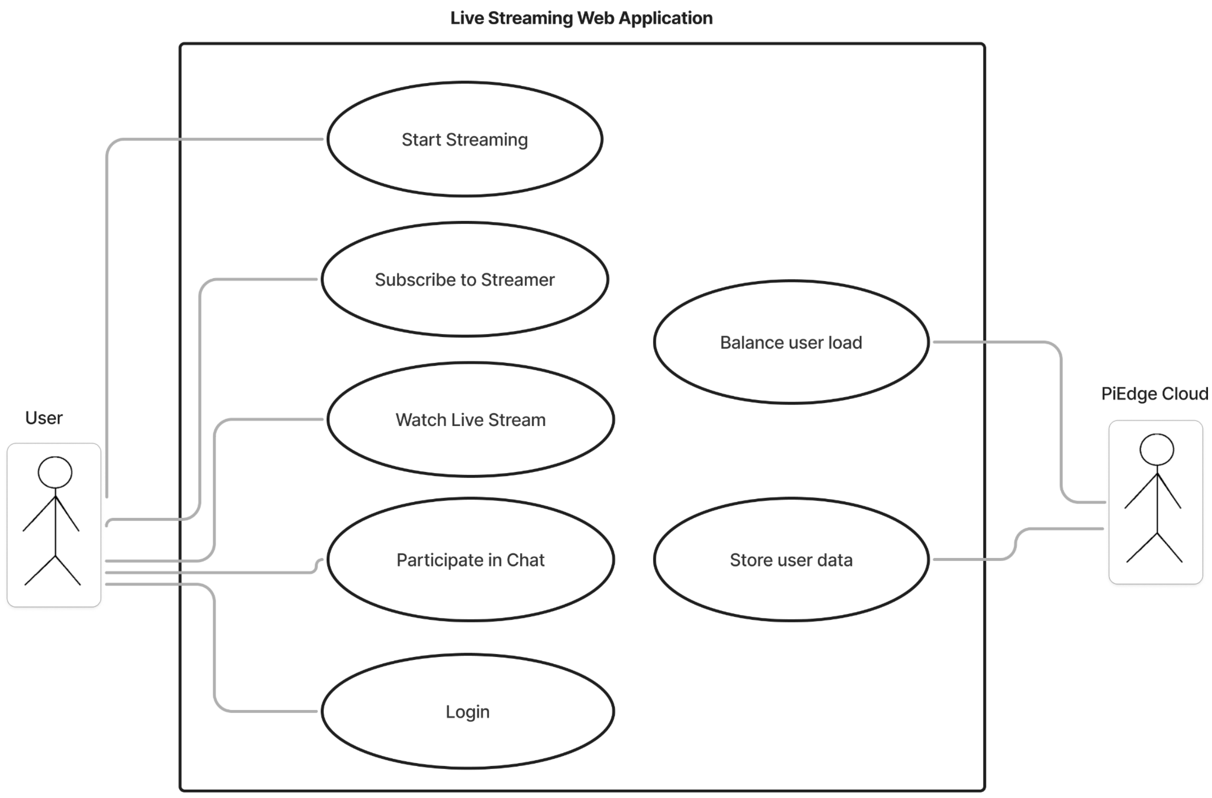

Figure 4: Use Case Diagram for Live Streaming Web Application

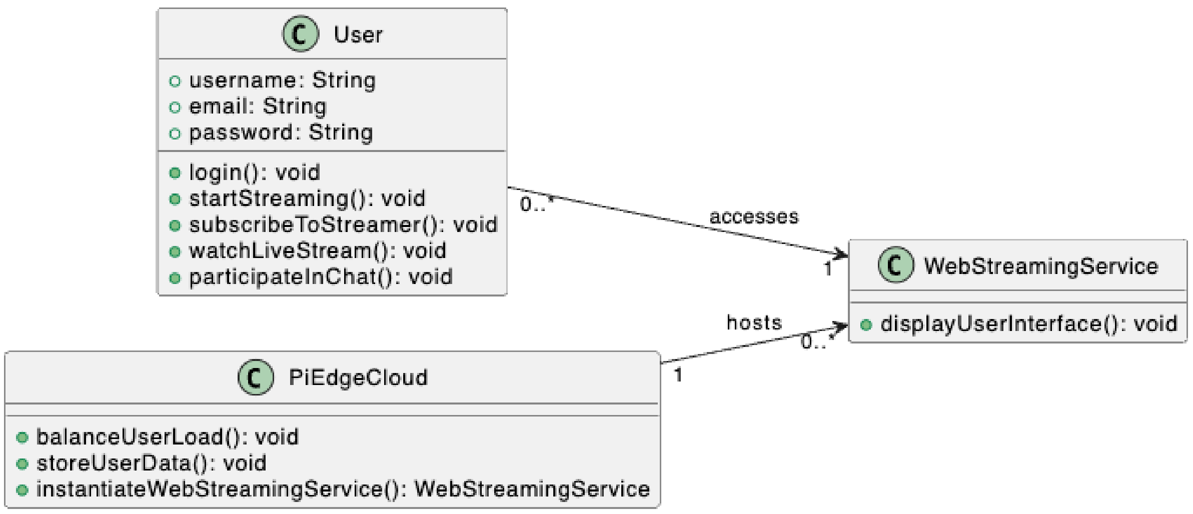

Figure 5: UML Class Design for Live Streaming Web Application

2. Equipment, Resources, Tools and Budget

Hardware:

- Master Node: 1 Raspberry Pi 4 dedicated to cluster management.
- Compute Nodes: 7 Raspberry Pi 4 units.
- Video Capture System: 1 Raspberry Pi 4 unit.
- Raspberry Pi Camera Module 3 for capturing live video.
- Gigabit Network Switch.
- NVMe Adaptor Board
- NVMe SSD drive for the master Raspberry Pi,
- Minimum 16 GB SD cards for each computer unit.
- Raspberry Pi Cluster Cases.
- Multi-socket power supply to support all units.

Budget:

Item	Quantity	Price	Source	Total		
Raspberry Pi 4 (8GB) + power supply + SD card	9	$123.10	PiShop	$1,107.90		
GigaBit Network Switch (8 port)	1	$26.99	Amazon	$26.99		
NVMe Expansion Board	1	$36.99	Amazon	$36.99		
NVMe SSD Drive (M.2 NVME SSD)	1	$59.99	Amazon	$59.99		
Cluster Case	2	$23.99	Amazon	$47.98		
Multi-socket Power Supply	1	$49.99	Amazon	$49.99		
Camera Module Raspberry Pi	1	$29.39	Amazon	$29.39		
Borrowed Raspberry Pi 4 from UBC	3	-$123.10	UBC	-369$		
					Total	$989.93
					Per-Member	$164.99

Software Tools:

- Cluster Configuration

Resource management and allocation: Kubernetes K3
Automation and scripting: Bash

- Machine Learning and Programming

TensorFlow & Python: Training and deployment of distributed machine learning algorithms for processing the live video feed.

- Performance

C++ is utilized where system-level optimization is required, where Python alone may be insufficient.

- Web Development

Frontend & Backend: Next.js (React, Node.js, and TypeScript)
Database: Prisma & MongoDB to handle data management.

Documentation and Tutorials:

- Raspberry Pi Tutorial
- Kubernetes Documentation
- Docker Swarm Documentation
- TensorFlow Documentation
- Installing Docker Swarm on 4 Raspberry Pis with Portainer Integration - Lab Series
- Guide for Writing Project Proposals
- Sample Project Proposal

B. Design Pattern

Our project leverages a suite of design patterns to ensure that each component is robust, flexible, and easy to maintain. By thoughtfully applying these patterns, we can seamlessly integrate diverse functionalities-from cluster management to real-time machine learning updates and web development.

1. Cluster Configuration and Container Orchestration

Singleton Pattern:
In the Raspberry Pi cluster, the master node serves as the unique controller for resource management. Implementation of the Singleton pattern guarantees that only one master instance is active, thereby preventing conflicts and ensuring centralized decision-making for resource allocation, task scheduling, and node coordination.

Factory Method & Abstract Factory Patterns:
As the cluster scales, dynamic instantiation of compute nodes becomes critical. The Factory Method pattern enables on-demand creation of individual compute node services, while the Abstract Factory pattern provides a unified interface for producing families of related nodes. This dual approach facilitates the flexible integration of new nodes into the cluster without necessitating extensive reconfiguration, ensuring scalability as processing demands increase.

2. Machine Learning Module

Strategy Pattern:
The machine learning component, responsible for tasks such as face recognition, object tracking, and object recognition, employs the Strategy pattern. This pattern encapsulates each algorithm as a separate strategy, permitting easy substitution or upgrade without altering the overall architecture. Such modularity facilitates rapid experimentation and adaptation to varying performance requirements.

Observer Pattern:
In a distributed processing environment, real-time communication between microservices is essential. The Observer pattern is utilized to broadcast updates from individual machine learning modules to other components, such as the web streaming application. For instance, when an algorithm detects an object or face, it publishes an event to a message broker. The web application, acting as an observer, subscribes to these events and overlays detection results—such as bounding boxes or labels—onto the live video feed. This decoupled communication mechanism enhances responsiveness and enables real-time data fusion from multiple algorithms.

3. Web Development

Model-View-Controller (MVC) Pattern:
The web application is architected using Next.js and Node.js, adhering to the MVC pattern. In this architecture, the Model component manages data and business logic, the View component renders the user interface, and the Controller processes incoming requests and orchestrates responses. This clear separation of concerns simplifies development and debugging, and supports future enhancements without disrupting existing functionality.

Adapter Pattern:
Integration of various internal modules or legacy systems is often necessary within a cloud infrastructure. The Adapter pattern is employed to create intermediary modules that translate external or legacy data formats into the internal representations expected by the application. For example, if a video streaming module outputs data in a different format or protocol than the modern web streaming platform, the adapter converts stream metadata—adjusting coordinate systems, timestamps, or data structures—to ensure compatibility. This isolation ensures that modifications to legacy outputs require changes only to the adapter, preserving the stability and integrity of the core platform.

4. Prototype Optimization

Preliminary prototypes have demonstrated that the application of these design patterns results in significant improvements in development efficiency and system performance. For instance, using the Abstract Factory pattern to standardize service component creation has reduced redundancy and streamlined the process of scaling the cluster. This modular approach facilitates the introduction of new features or upgrades to existing components with minimal disruption, ensuring that the system can evolve to meet growing user demands and technological advancements.

C. Team Composition and Project Management

Team Name: **Project π**

Team Members:

Patrick Agnimel
Johan Gonzaga
Syed Saad Ali
Michael Zhang
Algernon Ren
Jia Yi Lu

Roles and Responsibilities:

Patrick Agnimel – Quality Assurance and Documentation Lead. Responsible for maintaining comprehensive project documentation, coordinating research efforts, and overseeing quality assurance processes including module integration testing.

Johan Gonzaga – Lead for Machine Learning Module Development. Oversees the integration of TensorFlow and Python-based distributed algorithms for real-time video processing.

Syed Saad Ali – Project Manager and Lead for System Architecture and Cluster Configuration. Manages overall project planning, milestone tracking, and coordination of hardware and software integration.

Michael Zhang – Web Development Lead. Focused on developing the frontend and backend using Next.js, Node.js, and TypeScript, and integrating the database with Prisma and MongoDB.

Algernon Ren – Infrastructure and Hardware Assembly Specialist. Handles the setup of the Raspberry Pi cluster, networking components, and overall physical integration.

Jia Yi Lu – Operations and Maintenance Coordinator. Manages version control, CI/CD pipeline setup with GitHub Actions, issue tracking via the GitHub Project Kanban board, and documentation upkeep.

Communication and Collaboration

Primary Communication Tools:

- Discord chat group for day-to-day communication.
- Email for formal or long-form communications.
- Discord Video Calls for online meetings.

Document Collaboration:

- Google Docs for real-time collaborative work on reports and meeting notes.
- GitHub Repository for documentations.

Version Control and Task Tracking:

- Git and GitHub are used for version control.
- GitHub Actions facilitate CI/CD and automation.
- Kanban board and GitHub Projects are employed for task tracking and progress monitoring.

Meeting Schedule:

- Regular daily scrum meetings (time TBD) to discuss progress and challenges.
- Weekly sprint meetings every Sunday from 1 PM to 3 PM (in person).
- Weekly sub meetings every week during CMPE 246 Lab.

D. Timeline and Milestones

For this 6-week project starting on February 27, 2025, we will adopt an Agile methodology with weekly sprints. Below is the comprehensive roadmap detailing key phases, milestones, and expected outcomes for each sprint.

Sprint 1: February 27 – March 5, 2025
Cluster Hardware Assembly and Initial Environment Setup
Milestones: Procure and assemble all hardware components (Raspberry Pi 4 units, Raspberry Pi Camera Module 3, Gigabit Network Switch, Cluster case, storage devices, Ethernet cables, etc.). Install operating systems on the Raspberry Pis and conduct basic connectivity tests.
Expected Outcome: Fully assembled hardware with confirmed network connectivity and readiness for software deployment.

Sprint 2: March 6 – March 12, 2025
Cluster Configuration and Software Deployment
Milestones: Install and configure Kubernetes (using the K3s distribution optimized for Raspberry Pi). Develop and run basic container deployment tests to ensure proper resource management and automation using Bash scripting.
Expected Outcome: A robust cluster environment with container orchestration and resource management fully operational.

Sprint 3: March 13 – March 19, 2025
Development Video Capture System and Machine Learning Module
Milestones: Integrate a dedicated Raspberry Pi 4 unit with a Raspberry Pi Camera Module 3 to capture live video. Configure the system to transmit the video feed wirelessly to the cluster using a real-time streaming protocol. Deploy TensorFlow with Python to develop and test distributed machine learning algorithms that process the incoming video feed in real time, including tasks such as face recognition, object tracking, and object recognition.
Expected Outcome: A working prototype of the machine learning module that effectively captures, transmits, and processes live video feeds across the cluster.

Sprint 4: March 20 – March 26, 2025
Web Streaming Application Development
Milestones: Develop the web application’s backend and frontend using Next.js (incorporating React, Node.js, and TypeScript). Integrate Prisma with MongoDB for database management and implement dynamic load balancing on the cluster.
Expected Outcome: A functional web streaming application that provides real-time video streaming and messaging capabilities, with a stable backend and optimized performance.

Sprint 5: March 27 – April 2, 2025

Integration and System Testing

Milestones: Integrate the cluster configuration, machine learning module, and web application. Conduct comprehensive system-level testing, including dynamic load balancing tests and performance optimization (utilizing C++ where necessary).

Expected Outcome: A fully integrated system demonstrating seamless interaction between all modules, validated for performance and reliability.

Sprint 6: April 3 – April 8, 2025

Final Optimization, Documentation, and Presentation Preparation

Milestones: Perform final debugging and performance fine-tuning. Complete comprehensive documentation detailing system architecture, design patterns, and the project workflow. Prepare the final project presentation and report for submission.

Expected Outcome: A finalized, well-documented project that meets all objectives and is ready for evaluation.

E. Personal Reflection and Team Feedback

Individual Reflections:

Patrick Agnimel: Reflecting on the upcoming project, I am both excited to contribute to such an innovative endeavor at the intersection of AI, IoT, and cloud computing. Preparing for this project has deepened my appreciation for meticulous documentation and rigorous quality assurance processes. I am committed to ensuring that every component of the project is thoroughly tested and well-documented, thereby facilitating smooth integration and robust performance as the system evolves. Drawing on my past experiences, I understand the importance of iterative testing and clear, consistent communication among team members. As we embark on this journey, I intend to apply my expertise in quality assurance to establish comprehensive testing protocols and maintain detailed documentation that captures our design decisions, challenges, and learnings. Although the project has not yet started, I anticipate that open-ended challenges will arise, and I am prepared to adapt my strategies to address these effectively while upholding the highest standards of quality. I also recognize the value of personal accountability and continuous improvement. By actively seeking feedback from my teammates and reflecting on our progress, I aim to contribute to a dynamic and responsive project environment. Ultimately, this project not only represents a significant technical challenge but also serves as an opportunity for personal and professional growth. I look forward to collaborating with my dedicated team and achieving success together.

Syed Saad Ali: Brainstorming ideas for this project has left me in awe of the flexibility and potential of the Raspberry Pi. Our team ultimately decided on the Raspberry Pi cluster supercomputer project, integrating a unique blend of AI, IoT, cloud computing, and more. I see this project as a great opportunity to apply my knowledge from previous courses, such as Software Engineering and Introduction to Databases, and to leverage insights gained from their respective projects in this one. To excel in this project, I recognize the importance of maintaining efficient and frequent communication to ensure a shared understanding among team members. Furthermore, I am excited to work on such an innovative and creative endeavor alongside my highly dedicated teammates, who are making incremental progress every day. It is an honor to collaborate with such exceptional individuals, and I am committed to giving my very best.

Jia Yi Lu: I am enthusiastic about working on such a unique and creative project with talented individuals. The capabilities of the Raspberry Pi have always intrigued me, and this project will give me the opportunities to explore them. Ideally, our Raspberry Pi cluster will be able to perform object detection using an AI model integrated with computer vision peripherals, and be able to connect to the cloud so it can be easily accessible by anyone. Since AI is at the forefront of technology right now, it is exciting to engage with it to create a computer system that has potential real world benefits. To this end, I will strive to do my very best, and be personally accountable and responsible in the face of project challenges.

Johan Gonzaga: I think our team, even with the challenges, will be able to finish this project on time. Everyone has their own strengths and roles, but the project is still tough, especially with the higher-level software design and machine learning aspects. The big question is how efficient our implementation will be and what we can improve—whether that's setting up Kubernetes on our PiEdge cluster or refining the machine learning algorithm. That said, any concerns I have are outweighed by my determination to get it done. Overall, despite the challenges posed against us as a team, I am thrilled with the opportunity to develop my skills in machine learning and to develop something I can be proud of.

Algernon Ren: I'm proud of the team's synergy and dedication during the beginning stages of this project. In my role as the Infrastructure and Hardware Assembly Specialist, I am responsible for the assembly of our Raspberry Pi cluster and integrating the networking components, ensuring a robust and scalable physical framework. It was inspiring to see our hands-on efforts come together, reinforcing the strong foundation of our proposal. Working with such a collaborative group has prefaced every challenge into an opportunity for growth and innovation.

Michael Zhang: I'm really passionate about this project as I can dive into the role of Web Development Lead, allowing me to harness the full potential of Next.js, Node.js, and TypeScript to create a seamless and efficient web experience. Integrating the backend with Prisma and MongoDB will deepen my appreciation for modern database management and the power of type-safe queries. This role provides an incredible opportunity to apply my knowledge of full-stack development while refining my problem-solving skills in real-world applications. I am thrilled to contribute to this innovative project that pushes my technical boundaries, and I am grateful to work alongside such talented and driven individuals.

Academic Articles

[1] Y. Wu, L. Zhang, Z. Gu, H. Lu, and S. Wan, "Edge-AI-driven framework with efficient mobile network design for facial expression recognition," ACM Trans. Embed. Comput. Syst., vol. 22, no. 3, Art. 57, May 2023, pp. 1-17. [Online]. Available: https://doi.org/10.1145/3587038.

[2] G. E. Farrel, W. Yahya, A. Basuki, K. Amron, and R. A. Siregar, "Scalable edge computing cluster using a set of Raspberry Pi: A framework," in *Proc. 8th Int. Conf. Sustainable Inf. Eng. Technol. (SIET '23)*, New York, NY, USA, 2023, pp. 287–296. [Online]. Available: https://doi.org/10.1145/3626641.3626936

[3] D. G. Costa, "On the development of visual sensors with Raspberry Pi," in *Proc. 24th Brazilian Symp. Multimedia Web (WebMedia '18)*, New York, NY, USA, 2018, pp. 19–22. [Online]. Available: https://doi.org/10.1145/3243082.3264607.

AI chatbot tools, such as ChatGPT, will be used as a guide when we encounter difficulties in coding or determining how to proceed with the project. They should only be used as a last resort and will not replace human verification. All software engineering processes, including pull request reviews, must be conducted by lab members. Chatbots may, however, be used to enhance the writing quality of project documentation, such as README.md files.