Monash University Information Technology
FIT3162 - Computer Science Project 2

Final Report

Stream Processing Architecture using Apache Kafka for Analytics and Visualisation On A Dashboard

Khai Fung, Lim
29297311
klim0022@student.monash.edu

Yi Ping, Ho
29352258
yhoo0007@student.monash.edu

Chiu Gin, Chong
28842022
ccho0024@student.monash.edu

Fernando Ng
28737377
fern0002@student.monash.edu

Semester 1, 2020

Team 1C
Supervisor: Dr. Vishnu Monn Baskaran
WORD COUNT: 10579

Contents

1 Introduction

Technological advancements in recent years have enabled extensive real-time, large-scale data processing to be done in a cost effective manner. The idea of Big Data has created a new era of possibilities to tackle the world's leading problems especially in public security. Governments and private enterprises are starting exploit the potential of Big Data Analytics to aid them in accomplishing their interests. The security and surveillance sector is still in the early days of employing Big Data technologies in their products and the effectiveness of such frameworks are still in question.

Major cities in Malaysia have been facing numerous criminal cases and according to a study by NUMBEO, the capital of Malaysia, Kuala Lumpur has a crime index of 65.35 and is the sixth highest among cities in Asia (NUMBEO, 2020) along with four other cities in Malaysia in the top 20 most dangerous cities. These statistics have prove that traditional systems cannot be utilized adequately in this modern era. The role of a traditional surveillance system in deterring crime has always been passive (Singh, Aggarwal, & Kaur, 2015). Therefore, a modernized video surveillance system is essential to control the widespread increase of crime in Malaysia.

A system can be built in hopes of tackling crime in major cities in Malaysia in an effective manner. It aims to reduce the costs of maintaining and running a video surveillance system that traditionally requires a huge sum of money to build the required facilities and employ the minimum manpower to ensure the system is running optimally. Instead, it aims to automate the entire detection side of the system where no human is needed to supervise all video feeds in front of monitors 24/7. Furthermore, a fully automated system avoids the proneness of human error in its surveillance task. Everything will be recorded and detected according to tried and tested algorithms which will reduce the chances of missing out crime. All statistics, sensor data and feed can be directly stored as evidence to be analysed by relevant authorities.

Most importantly, this project aims to prove the feasibility of employing such open sourced and secure big data technologies to fight crime effectively. To put it in a detailed context, this project develops and deploys an active video surveillance system that performs real-time video processing and analysis from input streams and archives the processed videos into a database. The video frames from input streams are analysed using a face detection program to count the number of people present in the current frame. In addition to that, the system displays the processed videos and its data analytic findings onto a user interface dashboard for users to interact with the system and perform queries from the database. This system architecture is consisting of Apache Kafka, a cluster computing framework which is the backbone of our system, OpenCV, a highly optimized library with video analysis tools for video processing and MongoDB, an open-sourced database network used for high volume data storage.

2 Background

2.1 Introduction

An active video surveillance system differs from a traditional surveillance system in the sense that it gives continuous real-time analytics even in an excessively dynamic environment (Collins, Lipton, & Kanade, 2000). The system effectively screens the incoming input video streams for irregularities in real-time and takes relevant actions like notifying the administration control about the irregular activity. Traditional surveillance systems such as closed-circuit television (CCTV) on the other hand only allows video analysis to be done after a crime has already been committed.

One advantage of using active video surveillance over traditional systems is that various video processing functionalities can be incorporated into the framework to draw out information from the processed video in real-time (Wu, Ci, Luo, Ye, & Wang, 2011). The information can be portrayed onto a user interface to provide more details and statistical data about the input video stream. Engebretson (2010) features the information on a screen which is commonly referred to as a security dashboard. The information can additionally be stored in a database to be queried later for other analytical reasons. In short, active video surveillance systems are an improvement over traditional video surveillance systems that can't display real-time analytical data and require constant monitoring by an operator.

Questions about how effective CCTVs are against fighting crime has been asked for many years. A result from the study by Lawson, Rogerson, and Barnacle (2018) has even shown that street lights are more effective at deterring crime than CCTV surveillance cameras. This has caused an increase in demand for a surveillance system that can perform data analysis automatically on incoming video streams as well as run on a larger scale with more cameras. The advancement of image processing technologies using artificial neural networks in the form of deep learning has allowed video analysis such as detecting and recognising objects in video to be processed. Computer vision systems like these with a decentralised intelligent system outperforms the traditional approach. Some researches have done similar work that uses video surveillance footage to perform analysis and data extraction.

Data streaming systems are also showing drastic improvements over the decade. Cluster computing oriented systems like Apache Kafka and Apache Flink are commonly used in various industries. Open-source software like these that are readily available has eliminated the need for high-performance computing clusters which significantly require more time and effort (Syafrudin et al., 2017). Many public area of many countries has already deployed active video surveillance systems with the combination of data streaming and video processing technologies to acquire demographic information and reduce crime as well as terrorist attacks (Soehnchen, 2016).

2.2 Related Works

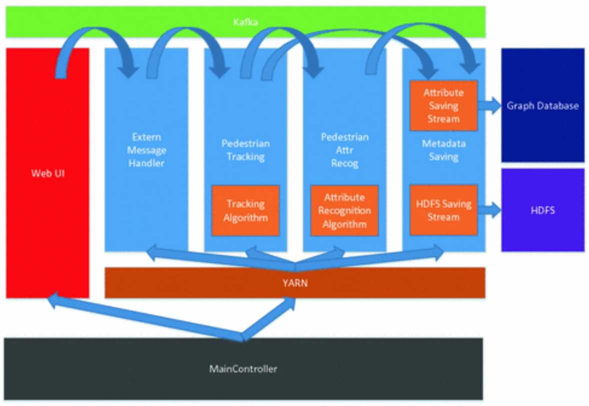


Figure 1: Diagram of the LaS-VPE Platform

Similar work has been done by many researchers over the years. An intelligent visual surveillance system referred to as Large Scale Video Parsing and Evaluation Platform (LaS-VPE Platform) was designed by Yu, Zhou, Li, Zhang, and Huang (2016) using big data processing framework like Apache Kafka, Hadoop Distributed File System (HDFS) and Spark Streaming. The design of the system is intended for adaptability in integrating different types of data processing for future framework upgrades. The LaS-VPE Platform performs transferring of messages between processors using Apache Kafka. From multiple viewpoints, the system built in the article has similarities with the active video surveillance system of this project with the fundamental contrast being the system uses web User Interface to interact with users instead of a surveillance dashboard.

Another proposed framework that involves three major parts including video stream collector and data stream buffer that sends data to be processed in the video stream processor is presented by Pandey and Singh (2018). The system uses Apache Kafka with OpenCV for incorporating a facial recognition system to detect pedestrians in the video streams. The captured video frames are converted and transferred in the system as JSON messages which is similar to what we have implemented in our project.

A different approach done by Zhang, Yan, and Kou (2016) uses KVM, a unified message data structure that handles video frames in Kafka. The system promises a scalable online video processing system that can be adapted to allow distributed computing. The system uses Xuggler, an open-source codec from Java to compress and decompress the video frames during processing and transferring. The framework of our system is similar to the proposed system in feeding live data from IP cameras to be transferred to the Kafka cluster and processed to be stored in a database.

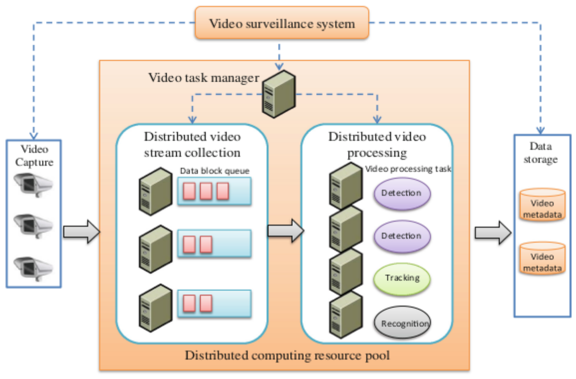


Figure 2: System for fine-grained online video processing

A framework that uses Apache Kafka and Spark is developed by Ichinose, Takefusa, Nakada, and Oguchi (2017) to produce a high throughput video processor from multiple video sources. A deep neural network is implemented in the system for digit recognition on images. Apache Kafka's performance in transferring data is thoroughly analysed in the article. The results from the article shows that Apache Kafka has high capability in scalability when the size of the computing cluster increases with up to a transfer throughput of 200,000 images per second.

Another study was conducted by Kreps, Narkhede, and Rao (2011) to test the performance of different distributed streaming software such as Kafka, RabbitMQ and ActiveMQ. The producer performance in publishing frames to consumer and the consumer performance in consuming messages are tested for all three software. In both tests, results show Kafka winning triumphantly at producing and consuming messages 4 times greater than its competitors.

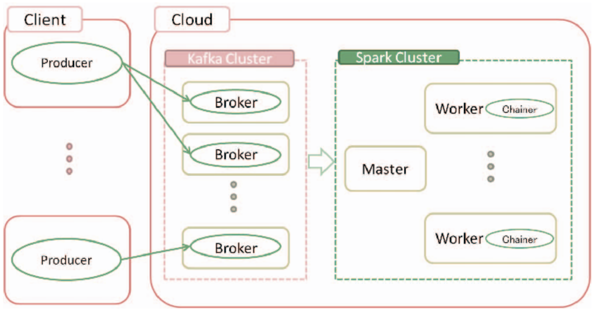


Figure 3: Stream Processing Framework Configuration

In conclusion, we have stated a number of research papers related to our project done in the past have helped us greatly as useful reference during the implementation of our system. The articles have presented some really well-designed systems for an active video surveillance system. Our system emphasizes less on testing the scalability of our system but more on building a functionally complete big data architecture.

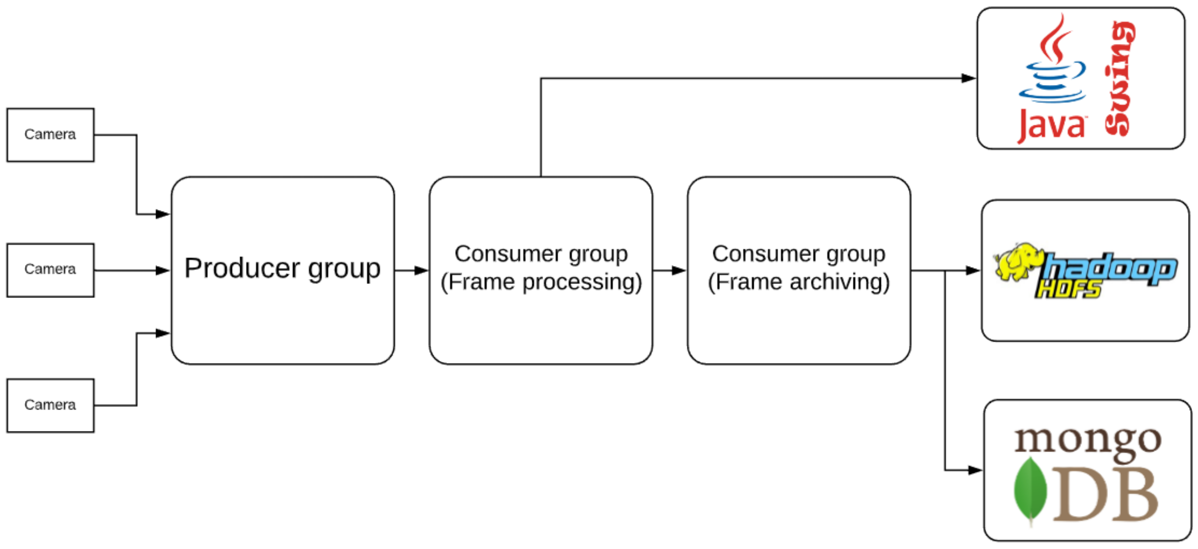


Figure 4: High-level system architecture

3 Methodology

3.1 System Architecture

Figure 4 shows the high-level architecture of the system. The source of new data into the system is the Kafka producer group (Producers). Nodes in this cluster connect to their assigned video sources and read frames into the system. The frames are then consumed by the processing group which performs arbitrary processing on the video frames. The processed frames are then consumed by both the dashboard and the archiving group. The dashboard which utilizes the Java Swing framework displays the video frames along with the extracted data from the frames and some miscellaneous information such as latency. On the other hand the archiving group accumulates the video frames and batches them into predefined intervals. The frames and their extracted data are then stored into a Hadoop Distributed File System (HDFS) cluster and a MongoDB database respectively.

Figure 5 shows the system at a lower-level. The core of the system is a Kafka cluster which serves as the central store for all video frames produced or reproduced in the system. This core can consist of several Kafka nodes that acts as servers that are managed by Zookeeper. The rest of the system is comprised of modules that interact with this central cluster to perform their respective tasks namely the Kafka producer group and the Kafka consumer groups. The following subsections will explore each of these components in more detail.

3.2 Video Stream Collectors

An active video surveillance would use IP cameras as input for the architecture. However, to mimic IP camera inputs, it is possible to use video files such as MP4 files to replace such IP camera video streams. Both video inputs are interchangeable and supported by the OpenCV library. Despite the video stream collector is not considered part of the system, it is required to connect and run IP cameras and video files through a property file.

The Video Stream Collector first reads a property file and gets the number of inputs that are to be sent to the framework. Then, it will start an event generator that is a Kafka Producer (Video Stream Producer).

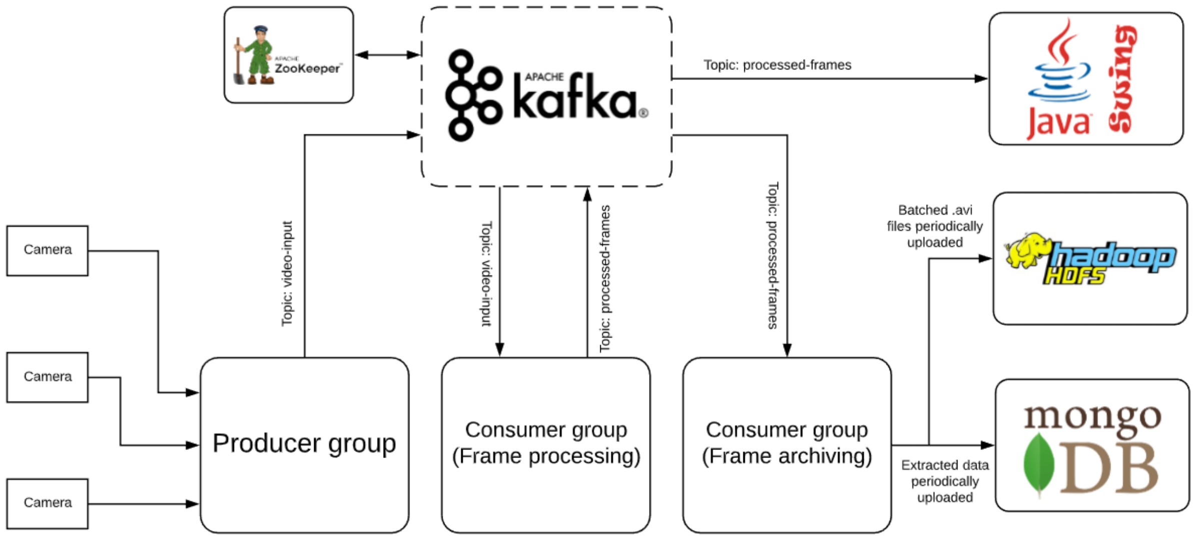


Figure 5: Low-level System Architecture

3.3 Video Stream Producers

Algorithm 1: Video Stream Producer Algorithm

```
pprops = getProducerProperty();
producer = new Producer(pprop);
properties =
  getCameraProperties(cameraId);
camera = new VideoCapture();
while camera.read() do
    image.resize();
    JSONobj = new JSON();
    JSONobj.addProperty(properties);

    data =
      serializeJSON(JSONobj);
    producer.send(ProducerRecord(
      topic, cameraId, data);
end
camera.release();
```

The first component of the system is a Kafka producer group which serves as the entry point for video frames into the system. As the name implies, nodes in this group read raw frames from their assigned video sources and packages them into JSON messages along with some relevant metadata. These frames are then published under the Kafka topic 'video-input'. Each video source registered with the system should be given a unique numerical identification number. It is important that this number be unique and numerical as the system's custom Kafka partitioner uses the ID directly as the message key.

Kafka clusters only guarantees ordering at the partition level. Therefore, in order to preserve the ordering of the video frames, each video source must be allocated its own partition within the Kafka cluster. By using a unique numerical ID number for each video source, the system can guarantee that all frames from a particular source will be kept in the same partition (preserving order) and will not share its partition with frames from other sources. This also allows for a higher number of partitions as compared to the alternative method of preserving message ordering in Kafka, by publishing all messages to one partition and stream filtering at the application level.

3.4 Video Stream Processors

After the frames have been introduced into the system, they are consumed by the frame processing group which are processing nodes with the Kafka group ID 'frame-processing'. In the current implementation, the processing group performs face detection using a Haar cascade classifier and draws bounding boxes over the detected faces. The number of faces detected is also saved along with the frame which is then republished to the Kafka cluster under the topic 'processed-frames'. The same key and partitioner is reused for this topic so as to mirror the original partition configuration. This allows the ID numbers to act as a consistent identifier for their corresponding video source across the system.

Algorithm 2: Video Stream Processor Algorithm

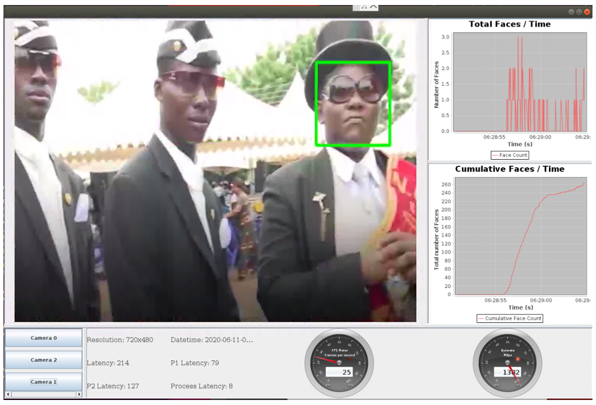
```
cprop = getConsumerProperties();
consumer = new Consumer(cprop);
consumer.subscribe(cprop.topic);
pprop = getProducerProperties();
producer = new Producer(pprop);
json = new JSON();
cascade = HaarCascade();
while true do
    crecords = consumer.poll();
    for record : crecords do
        cameraId = record.key();
        JSONobj = fromJson(
          record.value());
        properties = get(
          "properties");
        byteArray = get("frame");
        matrix = byteArray-
          toMat(byteArray);

        numFaces =
          cascade.detect(matrix);
        sendJSON = new JSON();
        sendJSON.addProperties(
          "properties");
        data =
          serializeJSON(sendJSON);
        producer.send(ProducerRecord(
          topic, cameraId, data));
    end
end
```

Another byproduct of having multiple partitions under the 'video-input' topic is that the system can rely on the Kafka cluster to automatically distribute the partitions among available processor nodes. This allows for improved performance as different partitions can be read in parallel.

The processors are in essence, Kafka consumers and producers in one application. They simply consume from one topic and republish their results to another. This means that the processor nodes can be easily replaced with counterparts that perform different types of frame processing as long as they adhere to publishing and consuming to and from the correct Kafka topics.

3.5 Video Stream Dashboard

The system dashboard is also implemented as Kafka consumer that reads video frames from all partitions of the topic 'processed-frames'. The dashboard also queries the Kafka cluster on startup for the number of partitions. It then updates the user interface to accommodate the correct number of video sources. As the number of video sources cannot be changed during run-time due to Kafka's limitation on live updating the number of partitions, a single query on startup is sufficient to ensure that the dashboard displays the correct number of partitions/video sources. Figure 6 shows the currently implemented dashboard.



Resolution: 720x480
Datetime: 2020-06-11-0...
Latency: 214
P1 Latency: 79
P2 Latency: 127
Process Latency: 8

Figure 6: System Dashboard

Due to the way Kafka offsets work, the dashboard must read all partitions in order to keep the Kafka offsets for each partition up to date with the latest messages. Otherwise, the video feeds that are not currently being displayed by the dashboard will freeze until the dashboard begins consuming messages for them again. Thus, all video feeds are consumed by the dashboard and it is the responsibility of the dashboard to filter them as per user selection based on the id of input.

Algorithm 3: Video Stream Dashboard Algorithm

```
initUIdesign();
addListeners();
initPlots();
cprop = getConsumerProperties();
consumer = new Consumer(cprop);
consumer.subscribe(cprop.topic);
json = new JSON();
while true do
    crecords = consumer.poll();
    for record : crecords do
        cameraId = record.key();
        JSONobj = fromJson(
          record.value());
        properties = get(
          "properties");
        if cameraId == selectedId
          then
            byteArray =
              decode("frame");
            matrix = byteArray-
              toMatrix(byteArray);

            matrix.resize();
            Img =
              matrixtoImg(matrix);
            displayImg(Img);
            updateCharts();
            updateStatistics();
        end
    end
end
```

The dashboard also displays analytical data obtained during the processing stage. As mentioned in the subsection 3.4, this information is bundled together with the frame as messages. The dashboard can simply retrieve this information from the message and display it. For example, the current implementation of face counting includes the number of faces detected in each frame in the frame message itself. The dashboard contains two plots for face detected against time, and cumulative number of faces detected of all feeds respectively. The number of faces detected in each frame can be used to update the two plots such that the charts appear to update in real time.

Aside from the analytical data, the dashboard also calculates the latency of the message, and the current number of frames per second, and the byte rate. The latency is measured by calculating the difference between the frame's time of creation against the time at which the dashboard receives the frame. Therefore, this latency represents the time elapsed since the frame was created to when the frame is displayed to the user. On the other hand, the frame and byte rates are simply calculated by counting the number of frames and bytes received for the particular partition/video source in one second. The latency calculated is split to four parts of Latency, P1, Process and P2 for further analysis. Latency is the total latency, P1 is the latency of Video Stream Producer to Processor, Process is the latency of running the image processing algorithm and P2 is the latency of Processor to Dashboard display.

3.6 Video Stream Archivers

Another consumer of the topic 'processed-frames' is the video stream archiving group. Nodes in this group accumulate processed frames and batch them into video files based on a user-specified period. The files are pushed from memory directly into the HDFS cluster via the Hadoop Java API which removes the need for any intermediary storage. The video files are given names based on a timestamp and their ID number.

Algorithm 4: Video Stream Archiver Algorithm

```
cprop = getConsumerProperties();
consumer = new Consumer(cprop);
consumer.subscribe(cprop.topic);
cameraIds = getCameraIds(consumer);
for camId : cameraIds do
    thread = new Thread(new VideoStreamWriter(camId, cprop));
end
```

Algorithm 5: Video Stream Writer Algorithm

```
aprop = getArchiverProperties();
consumer = new Consumer(aprop);
consumer.subscribe(aprop.topic);
connectMongoDB();
connectHDFS();
writing = true;
while true do
    crecords = consumer.pol();
    for record : crecords do
        frame = getJSON(record.value());
        if !writing then
            getWriters();
            setFilePaths();
        end
        if frameCount $\geq$ batchSize then
            sendHDFS(filename);
            inserttoMongo(data);
            updateFilePaths();
            writing = false;
        end
        matrix = byteArraytoMatrix(frame);
        write(matrix);
        frameCount++;
    end
end
```

Simultaneously, the archiver also aggregates the analytical data from the frames. The aggregated data is then pushed to the MongoDB database at the same periodicity as the video files. For example in the current implementation, the system aggregates both the total number of faces of the average number of face in each video file. As the video file is sent to the HDFS cluster to be stored, its corresponding analytical data is also sent to the MongoDB database keyed by the video file name. This way, previous video footage and its corresponding processed data can both be retrieved by video source ID and timestamp.

3.7 Supporting Classes

3.7.1 Util and Property File Reader

After refactoring the code base, it is found that there were several functions that repeated throughout the different class files that were stated before. Therefore, to follow the Do Not Repeat Yourselves rule, functions such as getProperties can be imported by other class files from other class files.

Meanwhile, any parameters can be added and changed directly from the property file. Therefore, this makes the framework dynamic and allows users to change settings and parameters easily without changing it in the code base. Further details of the codes of all classes can be found in the inline comments and documentation.

3.7.2 Partitioner and PublishCallback

The partitioner is a simple hashing function that can be modified to suit the needs of the camera Ids. Since our project is assuming a maximum small number of inputs, the hashing function will follow the integer of the camera Id. This also acts as the key for partitions used by Kafka.

Since sending in Kafka is asynchronous, there must be a completion message sent back to the Kafka producer to catch any exceptions. This is to complete the code and acknowledgments that is required for asynchronous communications.

4 Project Management

4.1 Project Overview

The project was an experiment to determine the feasibility of big data frameworks such as Apache Kafka to process huge amounts of video data in a streamlined and efficient manner. Therefore, to complete this project, proper project management should be undertaken to plan and carry out the project professionally. A six phased waterfall model will be used that includes phases of Requirements, Design, Proposal, Implementation, Testing, and Closure. Project management software and tools would be used to help manage the tasks and schedules of the team. Apart from utilizing project management tools, communications guidelines, risk assessment, and schedule for completing tasks will be used and handled by the project manager as well. Software developers will design and code the software of the project. Meanwhile, Quality Assurance will provide a comprehensive test plan to test all of the user acceptance criteria that has been initially agreed with the Project Supervisor as well as the code base of the project. Project milestones marks the progress of the team and allows a checkpoint for the team to review the progress.

4.2 Project Scope

4.2.1 Project Deliverables

There are two different types of deliverables that are to be submitted along witht his project. Firstly, Product Deliverables include deliverables that are related to the project which includes the software and all of the relevant reports. Meanwhile, project deliverables are deliverables that are related to the project management aspect of the project. The full list of deliverables is listed below:

Product Deliverables

- Code Base
- Code Report
- Code Documentation
- Test Report
- End User Manual
- Technical User Manual
- Methodology
- Final Report

Project Deliverables

- Scope Statement
- Risk Register
- Gantt Chart
- Test Plan
- Communications Matrix
- Team Management Report
- Meeting Minutes
- Work Breakdown Structure
- Organisation Chart
- Responsibilities Matrix
- Project Presentation

4.2.2 Project Requirements

The project requirements have been identified after several meetings with the project supervisor. The project requirements include both non-functional and functional requirements that are to be fulfilled by the active surveillance system. However, due to the COVID-19 outbreak and Movement Control Order (MCO) enforced by the Malaysian Government, we were not able to enter campus that has the necessary equipment to fulfill such requirements. To explain our requirements, functional requirements are the functions of the surveillance architecture that allows users and owners to successfully use the system as a video surveillance system whereas non-functional requirements include functionalities of the system which are not crucial to the operation of the system such as security, usability, and performance.

Table 10 lists the updated non-functional requirements while table 11 lists updated the functional requirements that is suited for the team to perform testing on the project remotely. The functional requirements have been captured as user stories from the perspectives of the three main actors involved with the system: the remote surveillance cameras, the user(s) of the system, and the surveillance system itself.

4.2.3 Product User Acceptance Criteria

User acceptance criteria for each of the user stories have been identified, updated and listed in table 12. These represent the requirements that our system must fulfill in order to satisfy the project description where the team accesses the project framework remotely. The product user acceptance criteria are requested from the project supervisor and updated when Monash has been forced to closedown from government policies.

4.3 Project Organisation

Figure 7 shows the structure of the project team that includes the name and the role of each member in the team. There is a project manager, two software developers, and a member in charge of quality assurance. Table 1 shows the directory of the team where the contact information of each team member is shown.

Name	**Email**
Lim Khai Fung	klim0022@student.monash.edu
Ho Yi Ping	yhoo0007@student.monash.edu
Fernando Ng	fern0002@student.monash.edu
Chong Chiu Gin	ccho0024@student.monash.edu

Table 1: Project Team Directory

4.3.1 Process Model

Development of this project will follow the waterfall model which is comprised of different major phases. This is because the requirements have been clearly defined and understood by the project team. Despite having a sudden change in project requirements, the changes were minor and are mostly on the conceptual level. Thus, it did not affect a large number of phases in the project lifecycle but mainly on Project Implementation and Project Testing and Verification phase. This shows that the waterfall model defined with the phases is suitable for our project. The said waterfall model is shown in Figure 8.

As usual, our project started off with the gathering of the requirements of our project from studying research papers and discussion with project supervisor. Through interviewing the project supervisor, who will provide the requirements and expectations for our project, a written document containing a brief description and expectations of the project can be created. After the requirements have been identified and understood, the project enters the design and prototype phase where a proof of concept is produced based on published research journals. The architecture of our system is drafted during this phase based on the proof of concept.

After the acceptance of our project proposal, the project moves into the implementation phase. This phase is where the working system is developed. To accommodate with the risks involved and the unprecedented COVID-19 pandemic, the implementation phase will include a project requirements review and update session. When all of the components are up and running, system testing and verification will be performed in accordance to the test plan. A test report will be also be produced in this phase summarising the results obtained from the testing process.

The final stage of the project starts when the system has been implemented and tested. A final report documenting the entire project development life cycle is produced and submitted along with the final product that is mainly the code base. All documents produced that are listed above for the project and product requirements will be archived in a team Google Drive folder while the code base will be stored on

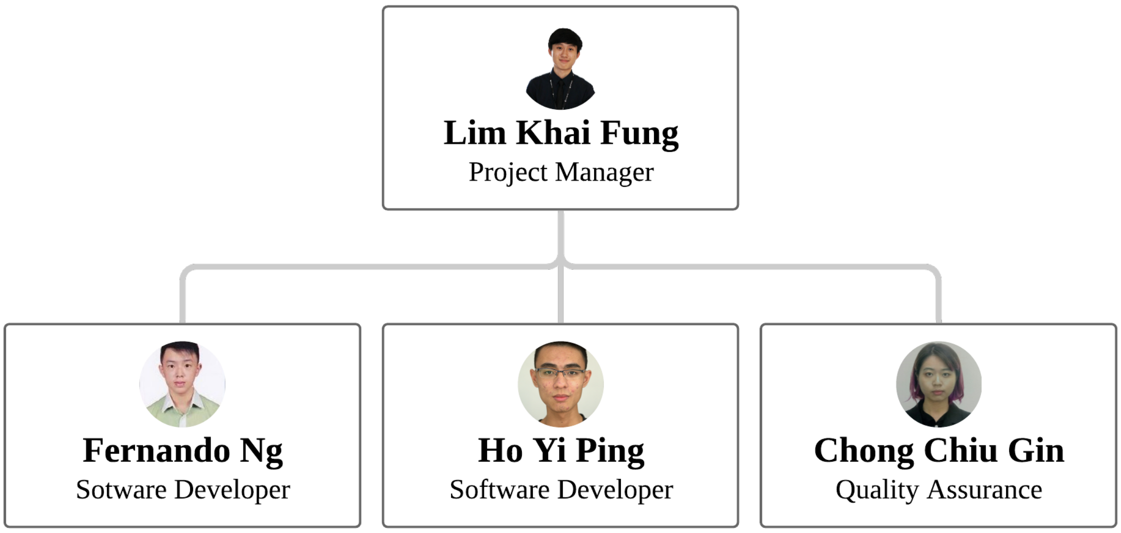


Figure 7: Project Organisation

the team's GitHub repository. Since there are currently no plans to provide long-term support for the system after handing it over to the clients, our development plan does not contain a maintenance phase after the project closure phase as opposed to conventional waterfall models.

Since the waterfall model is often too rigid for the detailed operations of the project, we will be adopting some flexibility into our development life cycle by using dynamic waterfall models, especially on the product generation processes to quicken decision making. An example of the dynamic waterfall model in action is shown in Figure 9.

From Figure 9, we can see that the process of getting a task done (in our case is to complete the software and hardware identification process) involves a lot of parties and communication where following the waterfall model entirely would slow down any decisions being made. Therefore, everyone's roles in the team are fluid and will have the power and trust to make their own decisions.

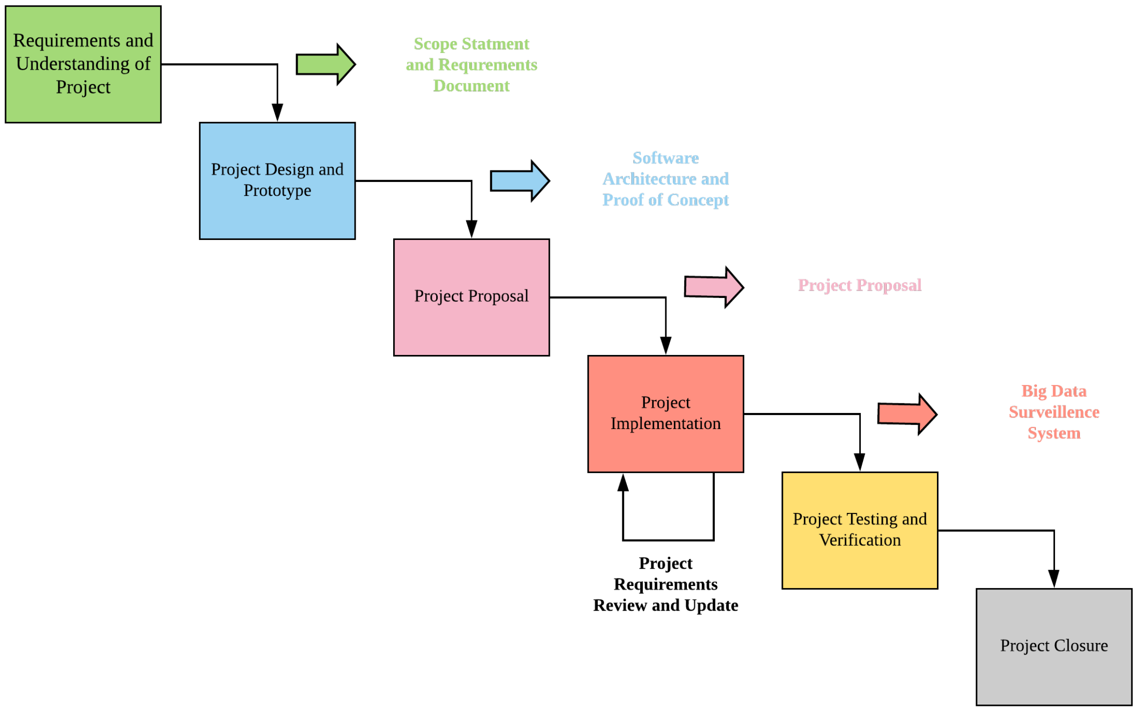
4.3.2 Project Responsibilities

To ensure that each individual member's responsibilities are clear and well defined to avoid disputes, we are using a RACI (Responsible, Accountable, Consulted, Informed) matrix to clearly define the responsibilities of each member for each task. The matrix is shown in Table 13 in the appendix. All main tasks that have been identified during both the planning phase and the project requirements review phase have been included in the RACI matrix.

4.4 Management Process

4.4.1 Risk Management

A risk register that captures information such as rank of the risk in the risk register, the description and category of the risk, root cause of the risk and what events will trigger the risk to cause damage, the potential responses facing the risk and the ways to prevent it from happening, the risk owners, the probability and the impact of the risk, and finally the rationale of the probability and impact of the risk has been created. The risk register will be constantly



Requirements and Understanding of Project
Scope Statment and Requrements Document
Project Design and Prototype
Software Architecture and Proof of Concept
Project Proposal
Project Proposal
Project Implementation
Big Data Surveillence System
Project Requirements Review and Update
Project Testing and Verification
Project Closure

Figure 8: Waterfall Model

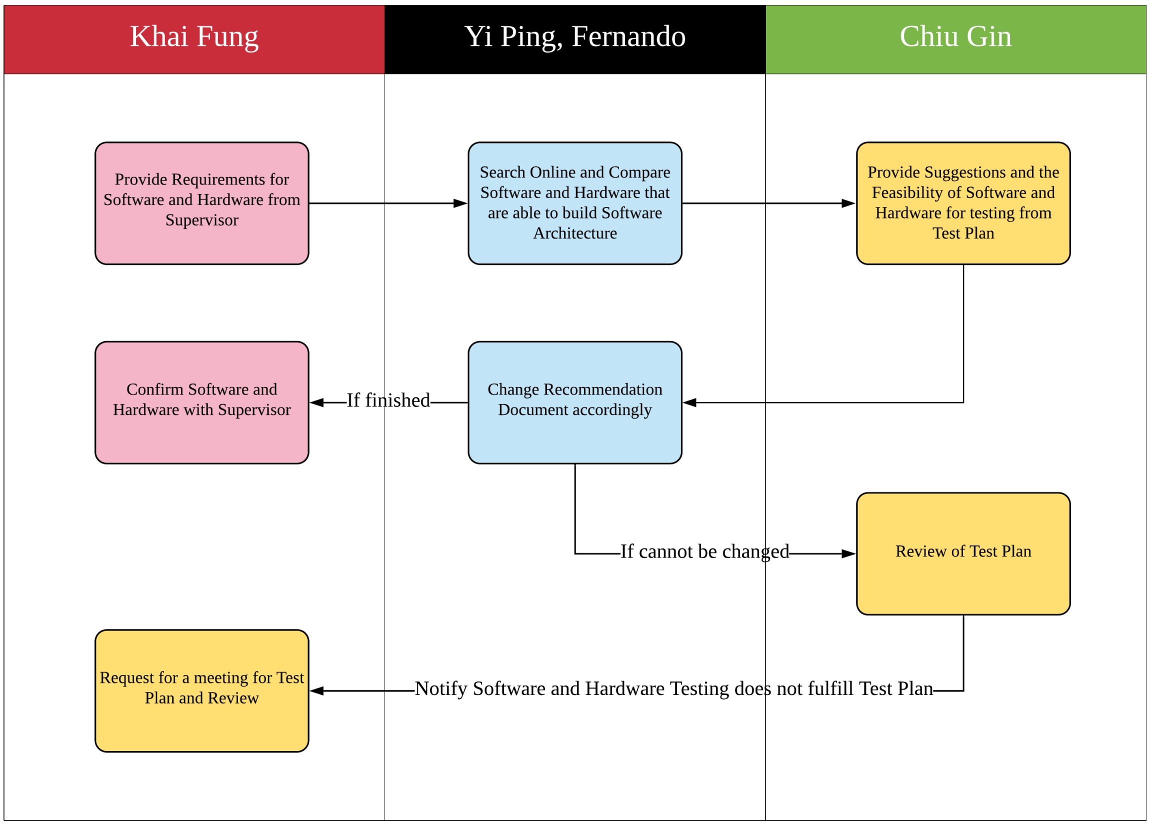
monitored and updated as more risks manifest themselves over time. Tables 14 and 15 in the appendix shows the updated and final risk register of the project. Whenever a problem or risk is encountered, the project team will first refer to the risk register if the risk is present to determine the next course of actions. If the risk is not present in the risk register, the risk register will then be updated.

Semester 1, 2020 was marked with the COVID-19 outbreak which caught the whole world by surprise and forced the government too lockdown the country and restrict all movements of its citizens. This caused a lot of problems onto how our project is planned to run and forced us to adapt and update all project requirements to suit the need to avoid breaking any law or imposing additional risks onto the safety and health of team members. Despite this unprecedented event is only captured under Natural disasters of our risk register, there were plenty of subsequent risks that have popped up from the change of method on the project operations. Following the movement restriction and fully remote implementation of the project, a lot of risks are identified and updated on the risk registers. The mitigation plans are discussed by the team members including project supervisor to ensure every involving member is clear on the risks involved. Further explanation will be on the risk registers shown in the appendix.

4.4.2 Monitoring and Controlling Mechanisms

To monitor all of the stakeholders, we have prepared a stakeholder engagement plan shown in Table 16 to provide a guideline on how to engage each stakeholder involved in the project either online or in person.

Moreover, to ensure consistency in our communications and meetings, we also have set up general guidelines for meetings as



Khai Fung
Yi Ping, Fernando
Chiu Gin
Provide Requirements for Software and Hardware from Supervisor
Search Online and Compare Software and Hardware that are able to build Software Architecture
Provide Suggestions and the Feasibility of Software and Hardware for testing from Test Plan
Confirm Software and Hardware with Supervisor
If finished
Change Recommendation Document accordingly
If cannot be changed
Review of Test Plan
Request for a meeting for Test Plan and Review
Notify Software and Hardware Testing does not fulfill Test Plan

Figure 9: Dynamic Waterfall Model

well as communication standards for all members to follow. This is to reduce the chances of miscommunication among members and to facilitate formal documentation of all project related communications (formal and informal) that have taken place. The communication standards define proper channels of communication among team members. Both of the documents are shown in Table 17 and 18 respectively in the appendix. A sample meeting minute is also shown in Figure 27 in the appendix. All meeting minutes throughout the implementation phase of the project is provided in the Team Management Report.

4.4.3 Communications Matrix

As a sample to show the possible communication events that will happen throughout the project for all stakeholders involved, we have prepared a communications matrix that encompasses the communications of between all stakeholders into Table 19 in the appendix.

The communication matrix includes the main communication types that the team will be using the communicate among members and the project supervisor. It also shows the purpose of each communication type and the medium used. The frequency of communicating with the audience is also stated as a guideline to follow for all team members who would like to update other members or the project supervisor. Each communication channel requires deliverables to be prepared to fully utilise communication effectiveness and time. The communication matrix also serves as the main reference document for team members who do not know what channels they should have their concerns resolved.

4.4.4 Video Conferencing, Recording and Tasks Management

Due to MCO and the closedown of campus, team members were not able to meet face to face for the second half of our project lifecycle that starts at the project implementation phase. To adapt to these dire changes, the project team has utilized online video conferencing software, Zoom for remote meetings between members of the Project Team and the Project Supervisor. Zoom is also used as the recording medium to record all of the important meetings or presentations that are to be submitted as part of the project deliverables. To replace communication between project members and supervisor, the team uses the Slack channel created by the project supervisor to report, consult and communicate with the project supervisor.

For Tasks Management, Trello, a free online Kanban Board is used to track and mark all progress and status of the tasks that are completed, ongoing, or to be started. Meanwhile, without in person access to the computers on campus, the team also uses Teamviewer to remotely access and carry out code, testing and analysis. These online collaboration software has made it possible to complete the project at each of our homes without meeting face to face.

4.4.5 Review and Audit Mechanism

One of the ways to escalate emergencies is through the proper communication channels. Each arising problem is grouped into four priorities which will have their decision makers and the time frame for the problem to be resolved. Therefore, team members can refer Table 20 in the appendix to be able to find the correct person to resolve the situation.

For the communication that do not belong to the Communication Escalation Process, team members can follow the communication flow chart as a general guideline for communicate with the team and the supervisor. In short, the communication flow chart separates the external and internal communication when the team resolves a problem requiring communication among stakeholders. The communication matrices are updated in accordance to the massive change of communication style of the team and are reflected on subsection 4.4.4 Video Conferencing, Recording and Tasks Management.

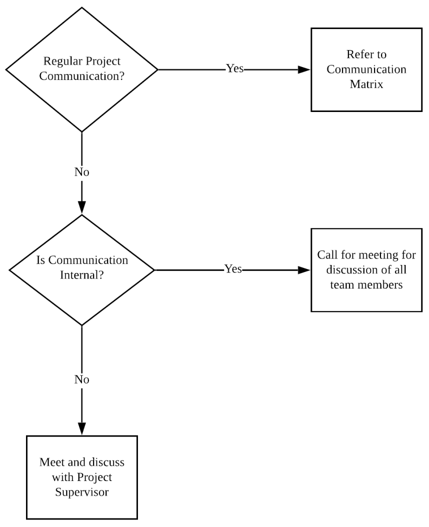

Regular Project Communication?
Yes
Refer to Communication Matrix
No
Is Communication Internal?
Yes
Call for meeting for discussion of all team members
No
Meet and discuss with Project Supervisor

Figure 10: Communication Flowchart

On the other hand, for our code implementation, we would be using GitHub as a version control tool. The public repository has been setup and each team member who wishes to contribute their code can push their changes to the code onto the main branch or individual branches which are maintained in Git by versions. Therefore, every change and historical version of the code is clearly archived and documented in Git. Our public repository for our project in Git can be accessed through `https://github.com/yipingho/FIT3161-Team-1C`. To work on the projects independently at our homes, the team used both Teamviewer to access the computer set up on campus for testing and reporting purposes while VirtualBox is used to mimic a virtual Ubuntu

Linux environment for code development.

For quality assurance on the code, we will be doing pair programming which reduces the probability of spaghetti code for our coding on the system architecture. Pair programming that is programming with two people on a program can be achieved since there are two software developers in the team.

All documentation and documents will be uploaded and updated in Google Drive and similar changes will be reflected on Overleaf that we will use to write our final reports. All of the team members will be able to visit the team's shared Drive and Overleaf shared papers to access the latest or historical documents of our project.

4.5 Schedule and Resource Requirements

To plan for the schedule of the project, the team has discussed on the tasks that are to be delegated to each team member. We would be using a Gantt Chart to record all of the tasks and show the timeline for each task. By using TeamGantt, we can easily have all team members use the Gantt chart directly by reporting the status of progress for their delegated task. All team members are also notified about their upcoming task deadlines via email. This is helpful in ensuring that each task can be done on time. Furthermore, the Gantt chart shows the dependencies of all of the task where we can clearly evaluate the flow of project by completing this task. Prerequisites for each tasks can be clearly identified and completed before taking on a new set of tasks.

To further help micromanage and track small but important tasks that are ongoing, completed or to be initiated, the team also uses Trello to track all tasks especially when developing the code base in the project implementation phase. This is where our process model differs from the conventional waterfall model where only deadlines are met. Trello helps the team to carry out code sprints of our own and track everything on a minute level instead of a high level only captured in the Gantt charts.

To ensure all of the tasks are defined clearly, we will use a Work Breakdown Structure (WBS) to list all of the items that are to be completed at each phase to finish the project. The Trello tasks will mimic the WBS but is a more detailed task board where each WBS item is further subdivided into sub-items that are newfound during coding. Therefore, there will be no ambiguity in all of the components for our project and everyone can keep track of the to-do-items for our project.

4.5.1 Gantt Chart

We have prepared a Gantt Chart schedule that will span the entire length of the project from the start of the team selection phase to the end of the project closure. The Gantt chart is split into two parts shown in Figures 28 and 29 where all of the tasks that we have identified will have their own schedule. By looking at both of the figures in the appendix, we can see that the Gantt Chart follows the Waterfall Model in a way that there are observable phases of the project separated into several non-overlapping timelines.

Figure 28 shows the timeline of our project since the finalisation of the project team. The actual project starts at the "Pre-project Planning" Phase that is equivalent to the Requirements Gathering phase that is stated in our Waterfall Model. Then, all of the tasks that involve designing the architecture and providing a logical proof of concept is done in the Project Design phase that is represented in light blue. Since most of our work before implementation would be to prepare a project proposal, we can see that the Project Proposal section takes in most tasks where each project management deliverable is included.

After the submission for our project proposal that is marked by a milestone, we will have a break to prepare for our exams. After that, the project team will start the implementation phase during our summer break. There will be a small break in between the end of December 2019 and January 2020 to allow the team to enjoy our winter holidays. The implementation for each component is split into 2 weeks per component to allow the team members to carry on on their external work or family matters mainly during Chinese New Year. After that, we will have all of our reviews on the project management plan, code and test plan to ensure that we have implemented the architecture that fulfils the requirements.

Finally, when everyone is back into the university for Semester 1, 2020, we have originally planned to utilise the Data Science Lab in the university to test our architecture. But due to unforeseen circumstances, we were forced to update our project requirements and thus our schedule. Therefore, after the project review that is between the project implementation and project testing phases, we will be reviewing and implementing the code updates that are required to cater the new project requirements updates. All reports that needs to be submitted at the end of the project will be further reviewed along with the test results and the outcomes analysis. Most importantly, to ensure that everything can be tested thoroughly, there will be two rounds of testing to fully ensure that the architecture is fully capable to perform its promised features as well as fulfilling the requirements. Then, the project will end with a post-mortem to have a final reflection and archiving of all of the documents.

4.5.2 Work Breakdown Structure

To ensure that all of the tasks and deliverables are listed clearly in a well defined structure, we have prepared a Work Breakdown Structure (WBS) to have a full list of tasks and deliverables that are required to be completed throughout the whole project. The WBS shown in Figure 30 in the appendix can be split into different parts that are equivalent to each of the phases of our project following the Waterfall Model. The tasks and deliverables for each of the phases are listed as the child of the branch of the phase in the WBS. For example, all of the work that is to be completed and included in the project proposal is listed as a child under the project proposal node. This is to ease the tracking of all tasks that are required to be done for a phase.

4.5.3 Product Backlog

A product backlog that includes the user stories, estimation of time and the priority has been included to fully estimate our project schedule in Table 2 in the appendix.

User Story ID	Estimation	Priority
1	1	3
2	2	4
3	10	10
4	5	7
5	6	8
6	4	9
7	7	8
8	3	5

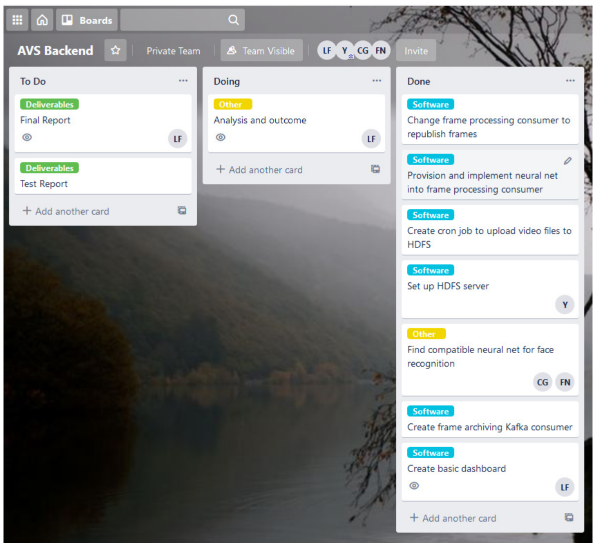
Table 2: Product Backlog

4.5.4 Project Budget Plan

Due to the changes in project requirements forced by the risks we have faced, the project has now gone full digital where there would be no IP cameras required and we will be replace those inputs with video files. Therefore, there will be no need of purchasing IP cameras. Furthermore, no testing will be required through AWS because testing, analysis and outcomes will be completed on a single local computer server. Therefore, there will be no financial expenses for the project.

4.5.5 Trello

Trello is an online collaborative Kanban board where it is very easy and intuitive to manage the tasks at hand and our to-do-list. We are not fully utilizing Trello to set all of the tasks and labels properly but we are using it as a marker for all to-do tasks, ongoing tasks and completed tasks to help the team remember the important tasks and not to leave out anything. This allows swift tracking of our project tasks and deadlines and can help us identify anything that has been left out. Moreover, updating the tasks are very simple and dynamic. A sample Trello board that we have used at one point of the project is shown in Figure 11.



To Do
- Deliverables: Final Report (LF)
- Deliverables: Test Report
+ Add another card

Doing
- Other: Analysis and outcome (LF)
+ Add another card

Done
- Software: Change frame processing consumer to republish frames
- Software: Provision and implement neural net into frame processing consumer
- Software: Create cron job to upload video files to HDFS
- Software: Set up HDFS server (Y)
- Other: Find compatible neural net for face recognition (CG FN)
- Software: Create frame archiving Kafka consumer
- Software: Create basic dashboard (LF)
+ Add another card

Figure 11: Trello Board

4.6 Limitations

As we have repeatedly emphasized on the effects of the COVID-19 pandemic and the MCO forced upon us, we would like to reiterate its impact on the project once again. The closedown of the campus and the country had severe impacts on the project team where one member was forced to go back home to Indonesia. Every member were not able to work together in person face to face and relied on online collaboration software to work on the project. Adding to this, the sudden change of project requirements forced a revision of the implemented code at a short time that was not planned in the Waterfall model.

The Waterfall model was not suitable for dynamic and quick changes and depends on the consistency of the project requirements. Therefore, the team was required to explore Agile methods and perform code sprints in short bursts to complete the code for the newly updated requirements. The distance between members did not contribute to the effectiveness of carrying out project tasks as members were not able to meet face to face and quickly resolve and answer any questions as well as clear confusions immediately. Members were forced to keep their questions up until the weekly meetings to be resolved which caused a slowdown in the process of implementation and testing.

Furthermore, explanations through online conferencing was vague and hard to convey to both team members and the project supervisor. Complex concepts could not be clearly explained verbally and presenting the concepts are limited by the software themselves. Moreover, the constant shutdown of the remote computer due to external reasons on campus did not help the situation. The team heavily relies on the computer because its results are key for the team to perform outcome analysis as accurately as possible since it meets the minimum requirements of hardware to run the entire project simultaneously where our virtual machines could not.

Overall, working at home significantly decreased the communication between members and project supervisor that could have been aplenty when everyone was able to meet face to face. Lastly, performing sprints remotely through accessing a remote computer with slow network connection was a painful experience and very ineffective.

5 Discussion/Outcomes

5.1 Kafka at Heart

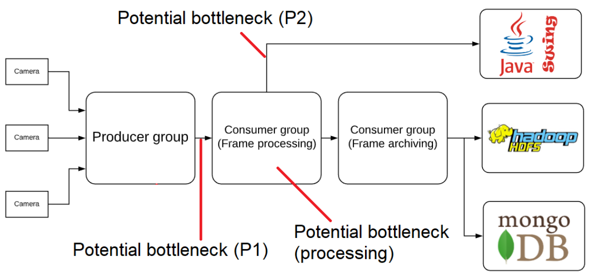
As discussed in the methodology section, the heart of the system is a Kafka cluster which serves as the central repository for all video frames represented as Kafka messages. Thus, the performance characteristics of the system as a whole is closely related to the performance of the Kafka framework. Although Apache Kafka was originally designed for relatively small logging messages, Confluent (original creators and current maintainers of Apache Kafka) and independent third parties have conducted studies that show that Kafka is capable of handling large, image scale messages (in the megabyte range) (Yu et al., 2016; Pandey & Singh, 2018).

The biggest issue of the system is the ever-increasing end-to-end latency (also known as unbounded catch-up time), which occurs if the video stream processors are not able to keep up with the rate at which new frames are being produced. There are many different possible approaches proposed by Confluent (Byzek, 2019) that one might take in order to tackle such a problem. For this system, we have decided to employ and test three such techniques in different combinations:

- Applying a small artificial delay to throttle the producer(s)
- Increasing the producer's batching factor
- Increasing the number of processor daemon

5.2 Profiling the System

Before we can start tweaking the system to improve performance, the system must first be profiled in order to identify all potential bottlenecks. Although this system is meant for use with IP cameras, we have substituted test video files instead for convenience and improved consistency in a performance testing environment. The videos used had a resolution of 720x480 and 3, 8-bit colour channels. By including timestamps within the frame messages, we were able to calculate and measure the time elapsed between the various components of the system. Figure 12 shows the three major potential bottlenecks found in the system.



Potential bottleneck (P2)
Camera
Camera
Camera
Producer group
Consumer group (Frame processing)
Consumer group (Frame archiving)
Java
Swing
hadoop HDFS
mongo DB
Potential bottleneck (P1)
Potential bottleneck (processing)

Figure 12: Potential bottlenecks

Confluent defines five major components of end-to-end latency in Kafka as:

- Produce time: processing the record and batching with other records in the internal Kafka producer
- Publish time: sending the record from Kafka producer to broker and appending the record to the leader replica log
- Commit time: replicating the record to follower replicas for fault tolerance
- Catch-up time: catching up to the record's offset in the log
- Fetch time: fetching the record from the broker

According to Confluent's definition of end-to-end latency, the identified potential bottlenecks of the system can be labelled as:

- P1: End-to-end latency between the producer group and the processing group
- P2: End-to-end latency between the processing group and the dashboard

- Processing: Latency induced by processing on the frame

Figure 13 illustrates the sections which are covered by each potential bottleneck. *init*, *postProc*, and *recv* are the timestamps that the system records. These timestamps are then used to calculate the values for $P1$, $P2$, and Pr.

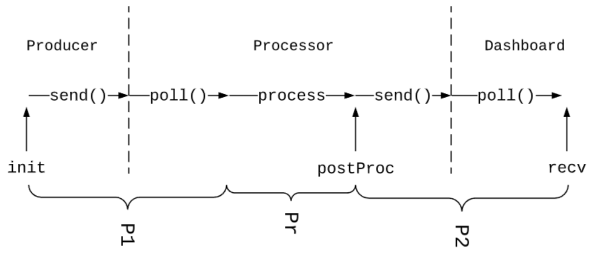


Figure 13: Sections of the system profiled

5.3 The Bottleneck

The system was first tested with one video source processed by one processor daemon without any modifications such as artificially added delays. Table 3 shows the latency induced by each of the potential bottlenecks in milliseconds against time elapsed since system start.

Time(s)	**P1**	**P2**	**Pr**	FPS
1	956	34	6	29
10	5462	27	9	32
20	12197	26	8	39
30	19108	22	17	32
40	25787	24	16	38
50	32801	27	14	35

Table 3: Induced latency in ms, 1 video, 1 processor, no delay

The values in Table 3 were retrieved by calculating using the following set of equations where the variables are named in Figure 13:

$$P1 = postProc - proc - init \qquad (1)$$

$$P2 = recv - postProc \qquad (2)$$

P1 is the apparent source of the increasing end-to-end latency while P2 and frame processing induces little to no latency in comparison. Figure 14 shows a plot of the overall latency of the system against time. From Figure 14, we can infer from the straight line that the rate of frame processing is a constant factor slower than the rate of frame production. Combined with the result that P1 is the latency bottleneck in the system suggests that the producers are producing at an overwhelming rate.

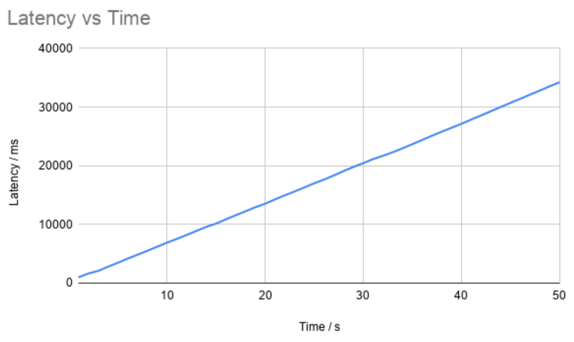


Figure 14: Overall latency of 1 video, 1 processor, 0 delay

The maximum frame rate a single processor can handle can also be calculated from the values in Table 3. Since the processor has an unlimited supply of video frames, the frame rate measurement obtained in this test represents the upper bound of the capabilities of the single processor daemon. By averaging the frame rate measurements from the test, a single video stream processor is estimated to be able to process 36.62 frames per second.

5.4 Throttling the Producer

As a result of using video files in place of IP cameras, the producers are not bounded by the reading rate of any physical image sensor. Therefore, the producers were reading and producing frames at a rate of ≈103 frames per second, which is significantly higher than any real IP camera (which produces at ≈24 frames per second). On top of this, Kafka clusters are implemented such that all writes and reads to and from a partition must be done through the 'partition leader', which can be any one particular node in the cluster. Combine this with the fact that each camera must publish to its specifically assigned partition means that the excessively high message production rate can quickly overwhelm the partition leader they are sent to, which in turn causes it to be unable to serve those frames to the other components of the system. The unbounded latency increase in P1 is the evidence of this phenomenon.

In order to throttle the producer to a more realistic rate, an artificial delay must be added into the reading loop of the producer such that the message production rate drops from the original 103 frames per second to 24 frames per second. The duration for this delay can be calculated using the following equation 3.

$$delay\_\mu s = \left(\frac{1}{fps_{desired}} - \frac{1}{fps_{in}}\right) \times 10^6 \quad (3)$$

Substituting $fps_{desired} = 24$ into (3):

$$fps_{in} = 103\ (measured) \quad (4)$$

$$delay\_\mu s = \left(\frac{1}{24} - \frac{1}{fps_{in}}\right) \times 10^6 \quad (5)$$

$$delay\_\mu s \approx 31957 \quad (6)$$

This results in a delay of approximately $32,000\mu s$ or $32ms$ which should be introduced in between frame reads in the producer should cause the producer to produce frames at roughly 24 frames per second instead of the previous 103. Table 4 shows the induced latency values obtained after the addition this delay.

Time(s)	P1	P2	Pr	FPS
1	34	27	13	29
10	21	27	22	24
20	26	25	8	25
30	26	31	17	24
40	36	33	18	21
50	26	22	14	25

Table 4: Induced latency in ms, 1 video, 1 processor, 32ms delay

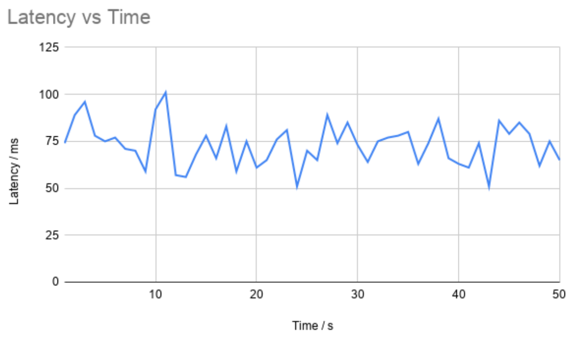


Figure 15: Overall latency of 1 video, 1 processor, 32ms delay

The addition of the artificial delay puts an upper bound on the P1 latency values, and consequently the overall latency of the system. The frame rate is also stable around the expected 24 frames per second, indicating that both the calculated delay was accurate, and that a single processor daemon can indeed cope with a more realistic frame rate. This is a drastic improvement over the original unbounded latency as the processing and display of frames in the dashboard can now keep up to date with live video reading with minimal latency. As shown in Figure 15, the overall latency of the system hovers around 70ms, with small spikes up to 101ms and down to 51ms.

To further verify the estimated capabilities of the processor, we tested a configuration of 3 video streams into 1 processor. In theory, this means a maximum total frame rate of ≤ 36.62 across all video streams is estimated to be processed, or ≈ 12.206 frames per second per video stream. In this configuration, the single processor will have to read from all three partitions and perform processing while keeping up with the production of new frames. By use of the previous equations, the artificial delay that must be introduced in between frame reads across all producers is $\approx 72,260\mu s$, which will result in a frame rate of $\approx$ 12.206. The results of this test are shown in Table 5 and Figure 16.

Time(s)	P1	P2	Pr	FPS
1	21	26	8	12
10	48	32	20	12
20	39	37	21	12
30	47	33	21	12
40	35	21	15	12
50	46	30	15	12

Table 5: Induced latency in ms, 3 videos, 1 processor, 72ms delay

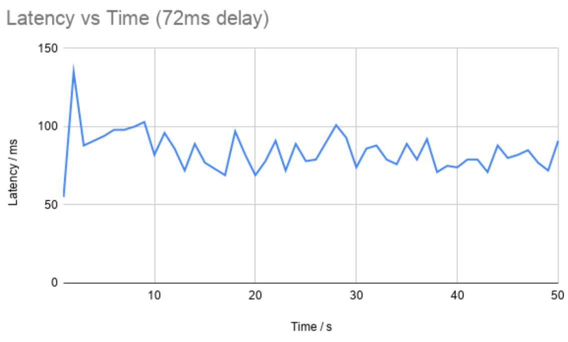


Figure 16: Overall latency of 3 videos, 1 processor, 72ms delay

As expected, the single processor is capable of handling three video streams at the calculated frame rate while keeping the overall latency from spiralling out of control. The latency values are nearly identical to those of the single video test but the system is now able to handle 3 video streams at 12 frames per second as opposed to 1 video stream at 24 frames per second.

5.5 Increasing Processors

Another method of scaling the system is to increase the number of processing daemons. As previously shown, a single processor daemon is capable of handling 1 video stream at a maximum of ≈36.62 frames per second or 3 video streams at ≈12.2 frames per second. By extension, 3 processors should then be able to handle 3 video streams at the 'full' 24 frames per second while keeping overall latency in check.

In order to test this, two additional processor daemons can be started alongside the original and the Kafka cluster will automatically perform load balancing and assign one partition to each daemon, thus evenly distributing the workload. The results for running 3 video streams at 24 frames per second with 3 processor daemons are shown in Table 6 and Figure 17.

Time(s)	P1	P2	Pr	FPS
1	64	32	5	32
10	25	54	9	24
20	59	25	11	25
30	27	22	8	24
40	23	24	6	26
50	31	32	12	20

Table 6: Induced latency in ms, 3 videos, 3 processor, 32ms delay

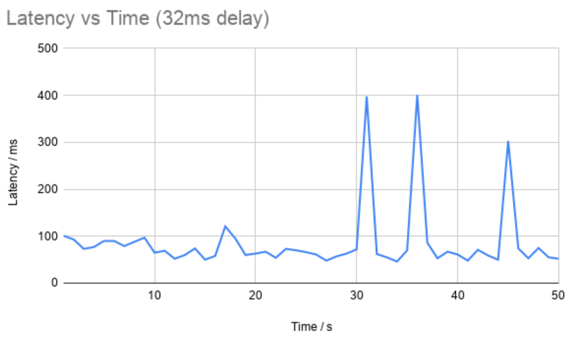


Figure 17: Overall latency of 3 videos, 3 processor, 32ms delay

The results are similar to those from a

single processor and video stream, with the exception of occasional large latency spikes. For example, three large spikes of up to 400ms are shown in figure 17 in the range $Time = 30$ to $Time = 50$. This is known as 'tail-latency' as they are at the tail ends of the latency spectrum (Povzner & Hendricks, 2020). A potential cause might be that the Kafka node is being flooded with requests due to the increased number of clients which it has to serve 3 processors and 3 producers, each of which makes at least one request to the node per video frame as every individual video frame is sent as its own request. The latency spikes may be a result of the Kafka node being hit by a burst of requests from all of its clients making requests to it simultaneously.

5.6 Producer Batching

One method to reduce the load on the Kafka node is to batch multiple frames into one request. Kafka provides parameters that can be tweaked to adjust the batching behaviour of the producers. For our system we have decided to use a time-based batching method due to the regularity of message production and consumption intervals. Adding a batching interval means that messages within the interval will be batched together under one request to the Kafka node.

Without message batching, every frame produced by either the producer or the processor must send at least one request to the corresponding Kafka node. There is also a fixed number of miscellaneous requests between each consumer and the Kafka node (e.g. consumer heartbeat, offset commits, etc) that is always taking place regardless of the messages being transferred, we shall refer to this as the communication overhead. Therefore, the number of requests sent by a producer or processor per second can be calculated with the following equation where the frame rate directly contributes to the number of requests sent to the Kafka node.

$$n_{requests} = fps + overhead \tag{7}$$

With the addition of message batching, the term $number\_of\_frames$ can be divided by some predetermined factor to lower the total number of requests sent to the Kafka node for the price of increased average latency. The following equations calculate the dividing factor and the resultant number of requests sent per second.

$$factor = batch\_interval \times fps \tag{8}$$

$$n_{requests} = \frac{fps}{factor} + overhead \tag{9}$$

Which is equivalent to

$$n_{requests} = \frac{1}{batch\_interval} + overhead \tag{10}$$

As shown in equation 10, the message rate is now determined by the batch interval instead of the frame rate (assuming batch buffer size is not a limiting factor). The trade off is that each message size becomes larger, as they now contain multiple messages instead of one, and the average latency increases due to the producers holding messages back for the specified amount of time to accumulate them before sending.

Table 7 and Figure 19 shows the results of adding a batching interval of 50ms to both producers and processors. With a frame rate of 24, the time to send a frame is approximately $1/24 = 0.0416$s or 41.666ms which is very close to the chosen batch interval of 50ms. This simply means that batching will not occur for every request, instead, equation 11 calculates the average number of frames per request.

$$avg_{frames} = batch \times 24 \tag{11}$$

Substituting $batch = 0.05$:

$$avg_{frames} = 0.05 \times 24 \tag{12}$$

$$avg_{frames} = 1.2 \tag{13}$$

This translates to every fifth request to the Kafka node containing two frames batched together instead of the usual one frame per message as illustrated in figure 18. It can also be observed that the value selected for the batching interval must be $> \frac{1}{fps}$ in order for any actual batching to occur, else the producer will never wait for a long enough period to accumulate more than one frame to be sent in every single message. By substituting $batch\_interval = 0.05$ into equation 10 (and assuming an overhead value of 0), we can estimate that this batch interval value will reduce the rate of messages sent from 24 to 20.

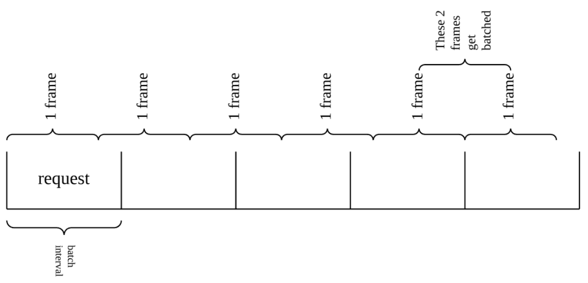


Figure 18: Effect of message batching on frame production

Although it may not seem like much batching actually took place, this batch interval value still made a difference to the overall performance of the system. As seen in Figure 19, the number of latency spikes have indeed reduced at the cost of a higher average latency. This acknowledges our assumption that the Kafka node was being overwhelmed by requests was the cause of the latency spikes.

Note that because the 50ms batching interval was applied to both the producer and processor, both P1 latency and P2 latency values increased by an average of 50ms each (recall from the system architecture that processors are also producers). As a result, the overall system latency increased by approximately 100ms (from average of 86.5ms without batching, to 192.16ms with 50ms batching), which is approximately the sum of the two batching intervals. This is because the producers will batch messages for 50ms after a frame read from the video source, and the processors will also batch their processed frames for an additional 50ms before republishing them to the Kafka cluster.

Time(s)	P1	P2	Pr	FPS
1	87	86	7	32
10	93	157	20	22
20	99	89	14	22
30	80	96	13	24
40	94	97	10	24
50	86	117	12	24

Table 7: Induced latency in ms, 3 videos, 3 processor, 32ms delay, 50ms batch interval

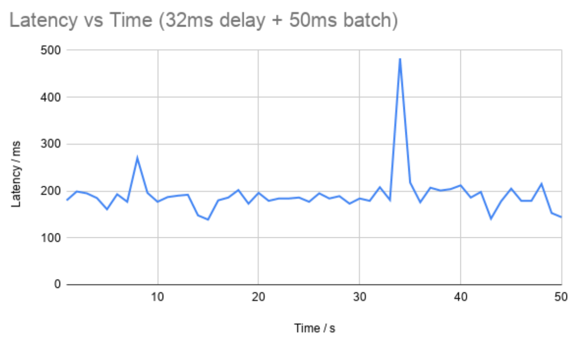


Figure 19: Overall latency of 3 videos, 3 processor, 32ms delay, 50ms batch interval

In an attempt to further reduce the latency spikes, another test with a batch interval of 100ms was carried out. Substituting $batch = 0.1$ in to equation 11 yields 2.4 average frames per message. This time batching is more aggressive as each request is guaranteed to contain either 2 or 3 frames. The results for this test are shown in Table 8 and Figure 20.

Again average latency increased by approximately 100ms to 310.98ms (up from 192.16ms at 50ms batching interval) that was increased due to a 50ms increase in producer and 50ms increase in consumer. The latency spikes have also reduced in magnitude and frequency compared to the previous test. Figure 21 shows a comparison of the 50^{th} and 99^{th} percentile latency across the three batch tests.

Time(s)	P1	P2	Pr	FPS
1	110	256	6	28
10	124	135	10	25
20	142	148	8	24
30	136	139	11	26
40	108	226	8	25
50	190	160	4	24

Table 8: Induced latency in ms, 3 videos, 3 processor, 32ms delay, 100ms batch interval

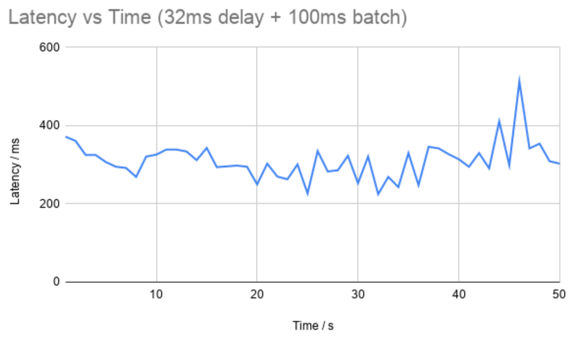


Figure 20: Overall latency of 3 videos, 3 processor, 32ms delay, 100 batch

By increasing the batching interval, the tail latencies can be reduced relative to the average latency, which is reflected in the decreasing difference between the 50^{th} and 99^{th} percentile latency values as the batching interval increases. This results in smoother latency values across the system, which in turn also results in smoother video playback on the dashboard.

There is of course a point of diminishing returns where increasing the batch interval value will no longer result in smoother latency values. This may occur when the source of latency is no longer from the saturation of the Kafka node(s) to handle requests, or when the producers' buffers are no longer large enough to hold all the messages accumulated in a single batch (at which point the request will be sent regardless of whether the batch interval has elapsed). Therefore, the appropriate value for the batch interval parameter is a matter striking a balance between acceptable mean latency and the occurrence of latency spikes.

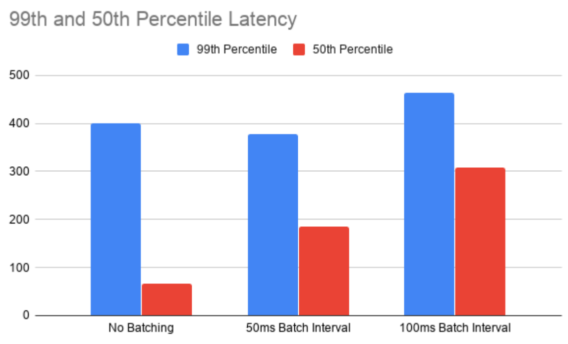


Figure 21: 99th and 50th percentile latency comparison between different batch intervals

It is interesting to observe that larger latency spikes tend to only start occurring at $Time = 30s$ onward. This is likely due to the memory buffers in the Kafka nodes filling up and them consequently having to perform disk IO operations to clear their buffers. There is no proper solution besides configuring Kafka to only retain logs for an incredibly short amount of time. However, doing so might risk Kafka purging messages before they have been properly consumed and processed, resulting in unwarranted loss of data.

5.7 Conclusion

The main novelty of this system is the use of an Apache Kafka cluster as a central repository for all video frames. This has allowed the system to adopt Kafka's highly scalable, modular, and flexible nature, as shown by the various tuning methods studied above. It shows that with proper tuning, a high end computer will be able to run the framework with encouraging results and is open to horizontal scaling. However, all results were done in a local network without the inclusion of network latency in a clustered environment.

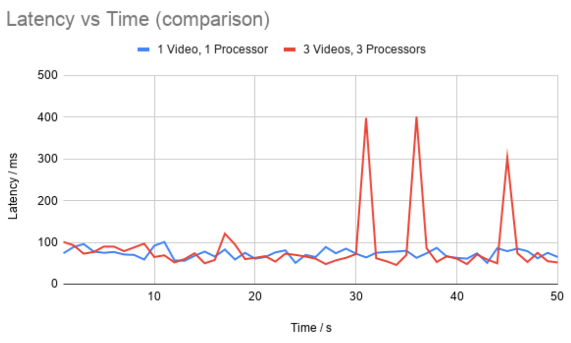


Figure 22: Latency values for different number of processors

Aggregated results for all tests that were carried out are shown in Figures 22 to 24. It is evident that different configurations would result in varying performances for different inputs. The ideal outcome would be to determine a single, fixed configuration which leads to the best performance overall. However, life is not without compromises and so is our system. Tweaking different parameters and using different system configurations each has their own strengths and weaknesses.

The best configuration is highly dependent on the production environment and product requirements. For example, if powerful machines are available or the video source have low frame rates, the system can be configured such that each processor daemon can handle more than one video feed and still has minimal latencies. Another example where a compromise must be found would be the balance between mean latency and tail latency values.

Depending on use case and system purpose, a high but stable latency may be more desirable than low but spiky latency or vice versa. For example, in active video surveillance, some environments would prefer a higher throughput with high but stable latency for cases such as recording and logging. Whereas, some environments would allow spiky data but prioritizes minimal latency for emergency response purposes such as detecting and fighting crime. The batch interval values can then be estimated using the methods suggested above but ultimately the proper tuning must be determined for its intended purposes.

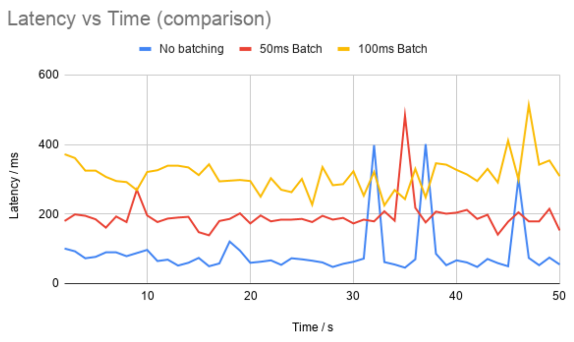


Figure 23: Latency values for different batch intervals

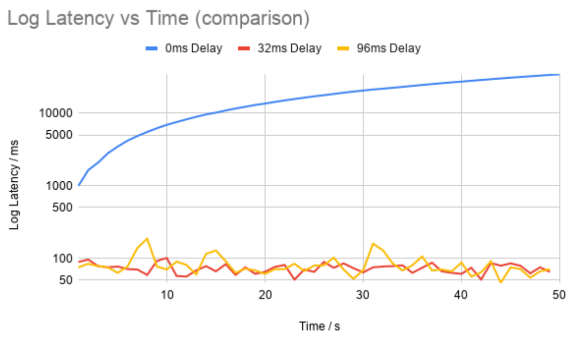


Figure 24: Log latency values for different producer delay values (recall that 0ms = 103FPS, 32ms = 24FPS, 72ms = 12FPS)

Furthermore, if a system does not require the strict ordering of video frames for an input, the system can be adjusted such that the absolute ordering of video frames can be traded off for increased throughput and latency. By disregarding the absolute ordering of the video frames, the system will be able to better distribute the frames to more than one processors or partitions as any processor can process frames from any video source at any given point in time which allows further parallelization for a single input. In contrast, the current implementation is limited where one video input can only have a maximum of one partition to maintain absolute ordering. By having to preserve perfect frame ordering throughout the system, it is inefficient to split any one video source across multiple processors as reassembly of the video is cumbersome and requires complex synchronization that will likely negate any performance gained from splitting it in the first place.

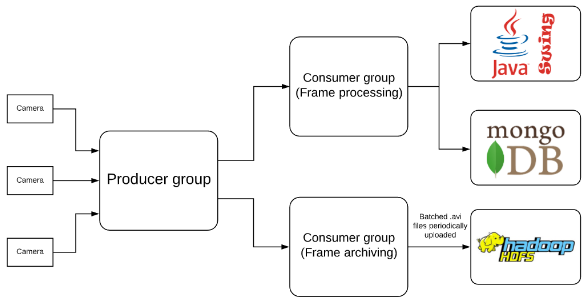

Camera
Camera
Camera
Producer group
Consumer group (Frame processing)
Consumer group (Frame archiving)
Java
Swing
mongo DB
Batched .avi files periodically uploaded
hadoop HDFS

Figure 25: Possible architecture if absolute ordering is not required

It can be argued that only reason to preserve perfect frame ordering throughout the system is so that they can be displayed to the dashboard in a coherent manner. However, depending on use case, a live video feed may not even be necessary if the desired monitoring and analysis can be automated and implemented as processor modules, hence eliminating the need for any human monitoring of the live video feeds. This would allow for a system architecture resembling the one shown in figure 25, which has improved load balancing and parallelism between the producer group and the processing group at the cost of perfect message ordering and data consistency.

For our project, we are assuming the system is for use in an environment where:

- A human monitor is required
- The system is ran on commodity hardware
- Moderate but stable latency is desirable

Based on our test results, we recommend that the following steps be taken to get the best performance from the system:

- Limit the number of daemons running on any single machine based on available memory, and logical cores
- Assign each video stream its own Kafka partition
- Assign one video stream to each processor daemon
- Set moderate batching interval values for both producers and processors

In conclusion, our tests show that such a system, which uses Apache Kafka for live video processing, is a viable way of implementing an active video surveillance system especially with the recent advancements in both hardware and software for real time video analysis. As shown above, the latency induced from the actual processing of frames is consistently low. This means that there is still a lot of room in the system for more complex video processing methods to be implemented while keeping the performance within acceptable thresholds.

5.8 Future Work

Due to unfortunate circumstances during the development of this system, there were a number of important factors which we were unable to test. The most important of which being deployment of the system across multiple machines in a network

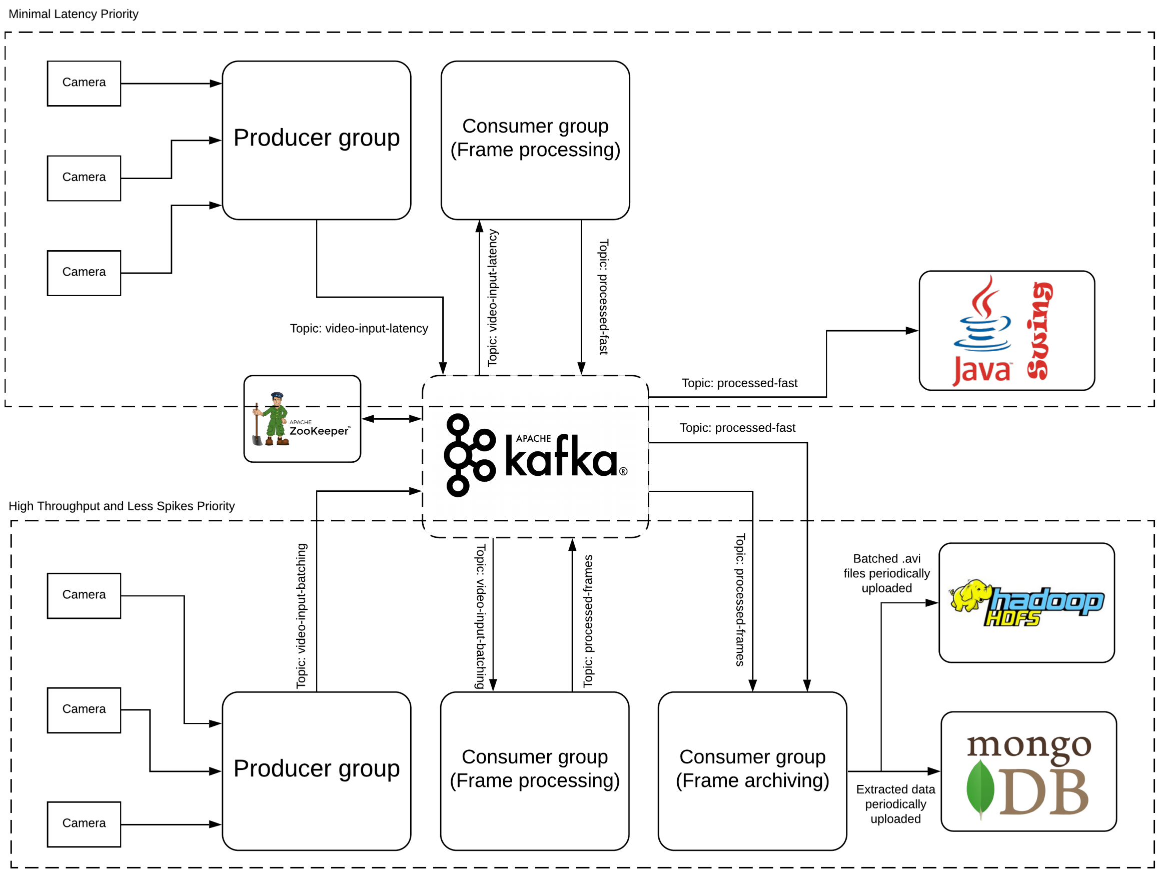

Minimal Latency Priority
Camera
Camera
Camera
Producer group
Consumer group (Frame processing)
Topic: video-input-latency
Topic: video-input-latency
Topic: processed-fast
Topic: processed-fast
Topic: processed-fast
Java
Swing
ZooKeeper
kafka
High Throughput and Less Spikes Priority
Camera
Camera
Camera
Topic: video-input-batching
Topic: video-input-batching
Topic: processed-frames
Topic: processed-frames
Producer group
Consumer group (Frame processing)
Consumer group (Frame archiving)
Batched .avi files periodically uploaded
hadoop HDFS
Extracted data periodically uploaded
mongo DB

Figure 26: Hybrid System Architecture

and its effects on performance. For example we were not able to test the scalability of running more than three to five processors due to hardware limitations. We were also unable to properly study the effects of increasing the size of our Kafka cluster and the effects of network latency on both Kafka and the system. Hence our test conclusions may make certain assumptions based on previous work done on these two subjects by other parties.

Finally, we have made an observation that since our system is modular and highly scalable through the additions of nodes onto the cluster, pure batching of a set interval would infinitely increase the average latency if we were to scale infinitely times horizontally by adding more processor and consumer layers in between input and output. Then, the system will be able to take in large throughput at the cost of very high latency. As we have said that all systems configuration depends on the use case, we can explore the idea of a hybrid system architecture, where different video inputs can pass their data through the framework on different topics, producer settings and consumer settings.

This allows a full coverage and potential use of the Kafka cluster where inputs are allowed to be configured according to their use case. For example, the architecture shown in Figure 26 can be split between a group of inputs and nodes which prioritizes minimal latency for display purposes and another group which prioritizes throughput and consistent data sending for data storage and logging purposes through the use of different Kafka topics, consumer and producer groups.

References

Byzek, Y. (2019). *Optimizing your apache kafka deployment.* Confluent Inc. Retrieved from https://assets.confluent.io/m/6b6d4f8910691700/original/20190626-WP-Optimizing_Your_Apache_Kafka_Deployment.pdf

Collins, R. T., Lipton, A. J., & Kanade, T. (2000, Aug). Introduction to the special section on video surveillance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *22*(8), 745-746. doi: 10.1109/TPAMI.2000.868676

Engebretson, J. (2010, 03). What's a security dashboard? *SDM*, *40*(3), 77-78,80. Retrieved from https://search-proquest-com.ezproxy.lib.monash.edu.au/docview/228467931?accountid=12528 (Copyright - Copyright BNP Media Mar 2010; Document feature - Illustrations; Last updated - 2014-05-25; SubjectsTermNotLitGenreText - United States–US)

Ichinose, A., Takefusa, A., Nakada, H., & Oguchi, M. (2017, Dec). A study of a video analysis framework using kafka and spark streaming. In *2017 ieee international conference on big data (big data)* (p. 2396-2401). doi: 10.1109/BigData.2017.8258195

Kreps, J., Narkhede, N., & Rao, J. (2011). Kafka: A distributed messaging system for log processing. In *Proceedings of the netdb* (pp. 1–7).

Lawson, T., Rogerson, R., & Barnacle, M. (2018). A comparison between the cost effectiveness of cctv and improved street lighting as a means of crime reduction. *Computers, Environment and Urban Systems*, *68*, 17 - 25. Retrieved from http://www.sciencedirect.com/science/article/pii/S0198971516304240 doi: https://doi.org/10.1016/j.compenvurbsys.2017.09.008

NUMBEO. (2020, June). *Crime index by city in asia 2020.* Retrieved from https://www.numbeo.com/crime/region_rankings.jsp?title=2020&region=142

Pandey, A., & Singh, H. (2018, 02). Face recognition of pedestrians from live video stream using apache spark streaming and kafka. *International Journal of Innovative Technology and Exploring Engineering*, *7*, 4-10.

Povzner, A., & Hendricks, S. (2020, February 25). *99th percentile latency at scale with apache kafka.* Confluent Inc. Retrieved from https://www.confluent.io/blog/configure-kafka-to-minimize-latency/#:~:text=If%20your%20total%20throughput%20is,may%20actually%20improve%20overall%20latency

Singh, S. K., Aggarwal, A., & Kaur, K. (2015, Mar). Evaluation & trends of surveillance system network in ubiquitous computing environment. *International Journal of Advanced Networking and Applications*, *6*(5), 2487-2494. Retrieved from https://search-proquest-com.ezproxy.lib.monash.edu.au/docview/1686107001?accountid=12528 (Copyright - Copyright Eswar Publications Mar/Apr 2015; Document feature - Diagrams; ; Tables; Last updated - 2015-06-05)

Soehnchen, A. (2016, 03). Video surveillance in public transportation. *Mass Transit*, *42*(2), 26-28. Retrieved from https://search-proquest-com.ezproxy.lib.monash.edu.au/docview/1770839657?accountid=12528

(Copyright - Copyright Cygnus Publishing Mar 2016; Document feature - Charts; Graphs; Last updated - 2016-03-07; SubjectsTermNotLitGenreText - United States–US)

Syafrudin, M., Fitriyani, N., Li, D., Alfian, G., Rhee, J., & Kang, Y.-S. (2017, Nov). An open source-based real-time data processing architecture framework for manufacturing sustainability. *Sustainability*, *9*(11), 2139. Retrieved from http://dx.doi.org/10.3390/su9112139 doi: 10.3390/su9112139

Wu, D., Ci, S., Luo, H., Ye, Y., & Wang, H. (2011, Oct). Video surveillance over wireless sensor and actuator networks using active cameras. *IEEE Transactions on Automatic Control*, *56*(10), 2467-2472. doi: 10.1109/TAC.2011.2164034

Yu, K., Zhou, Y., Li, D., Zhang, Z., & Huang, K. (2016). A large-scale distributed video parsing and evaluation platform. In Z. Zhang & K. Huang (Eds.), *Intelligent visual surveillance* (pp. 37–43). Singapore: Springer Singapore.

Zhang, H., Yan, J., & Kou, Y. (2016). Efficient online surveillance video processing based on spark framework. In Y. Wang, G. Yu, Y. Zhang, Z. Han, & G. Wang (Eds.), *Big data computing and communications* (pp. 309–318). Cham: Springer International Publishing.

Appendix

Component	**Link**
Communications Matrix	https://drive.google.com/open?id=19VunoelWGNY2XdNqmp10vGJPpZLhFikS
Gantt Chart	https://drive.google.com/open?id=1TF58YVimFrE2QP1idtDLRy__L-OtNfQr
Low Level Architecture	https://drive.google.com/open?id=1ldY9JBJ_WI-KF-lU7UDBkQvr63e_hXxf
Risk Register	https://drive.google.com/open?id=1LeZUMoSON0gaSmqxPlpal2ttSXlgAL9Q
System Dashboard	https://drive.google.com/open?id=1L1TWmbFphQQ-bnjzJ8Ts0ZsZNjWec2pM
Stakeholder Engagement Plan	https://drive.google.com/open?id=19VunoelWGNY2XdNqmp10vGJPpZLhFikS
Work Breakdown Structure	https://drive.google.com/open?id=1ylBUn2tpm84cflwslLQVMTbe5fjc96WK

Table 9: High Definition Links to Large Diagrams/Tables

ID	Requirement Description	Category
1	Multiple concurrent video streams within 2 sec delay (2 sec latency)	Performance
2	Video streams will at least have 720p resolution (1280 x 720)	Quality
3	Database to query in less than 1 sec (statistics of video)	Performance
4	Clean overlay with buttons for user to click	Usability

Table 10: Non-Functional Requirements

ID	User Story
1	As a video file, I would like to upload the my video files as frames to the system for processing
2	As a user, I would like to view video streams from the different video inputs
3	As a user, I would like to view data analytics done by the system on the online dashboard
4	As the system, I would like to receive incoming video streams from all video files provided in the file path
5	As the system, I would like to be able to perform data analysis tasks on the incoming video file
6	As the system, I would like to be able to produce a summary of the data analysis on an online dashboard
7	As the system, I would like to stream the received video streams onto the online dashboard
8	As the system, I would like to store the processed data in a database for future retrieval

Table 11: Functional Requirements as User Stories

ID	User Story ID	Requirement Description
1.1 - 1.2	1	-System can play video files -Video played in local files can be read to the system at a reasonable rate
2.1 - 2.2	2	-System must be able to publish the video streams captured from all video file sources to the online dashboard with reasonable latency -Dashboard must have a way of allowing users to view video feed of all played videos
3.1 - 3.2	3	-System must post results of data analysis to the dashboard -Dashboard must clearly display the analysis results in accordance to the selected video feed
4.1 - 4.2	4	-System must be able to connect to video files -System must be able to receive video feed from all video file paths
5.1	5	-System must be able to perform data analysis tasks in real-time on incoming data
6.1	6	-Database can show the summary of analysis output
7.1 - 7.2	7	-System must stream its data into a database. -Database must be able to serve the stored data on request

Table 12: User Acceptance Criteria

Task	Project Manager	Software Developer	Quality Assurance	Project Supervisor
Requirements	R	A	I	C
Software and Hardware	A	R	I	C
Architectural Design	A	R	I	C
Test Cases	A	I	R	C
Proof of Concept	A	R	I	C
Design/Prototype Report	I	R	A	I
Literature Review	I	R	A	C
Management Plan	R	I	I	I
Project Model	R	I	I	I
Project Organisation	R	I	I	I
Project Responsibilities	R	I	I	I
Deliverables	R	A	I	C
WBS	R	I	I	I
Project Schedule	R	C	C	I
Project Review	R	A	C	C
Version Control	I	R	I	I
Risk Register	R	I	A	C
Meeting Outline	R	I	I	I
Communications Matrix	R	I	I	C
Methodology	I	R	I	C
User Acceptance Criteria	R	I	A	C
Test Plan	A	I	R	C
Presentation	R	A	C	I
Proposal	R	R	R	C
Software Installation	I	R	I	-
Implementation	A	I	I	C
Code Review	I	R	A	I
Documentation	I	R	I	-
Setup	A	R	I	I
Demo	R	R	A	C
Test Report	I	A	R	I
Outcomes Analysis	R	R	R	I
Team Management Report	R	C	C	I
Code Report	A	R	C	I
Task Management	R	C	C	I
Post Mortem	R	R	R	I

Table 13: RACI Matrix

No.	Rank	Risk	Description	Category	Root Cause	Triggers	Potential Responses	Prevention Methods	Risk Owner	Probability	Impact	Status	Score	Score Rationale
1	23	Financial attraction of project to investors	This project does not intend to attract any financial incentives from outsiders	Positive	Project idea is suitable to be deployed as a product in the industry	Project is pitched to external parties	Discuss with project supervisor and the university	No prevention methods needed	Project Team, Supervisor	1	2	Open	2	Since the project will not be pitched to the public, it is highly improbable to be known by external parties. Financial investors will not affect the operation of the project
2	8	Server failure	Kafka and Zookeeper servers are interrupted and crashed	Technical	Data servers are shared by the university and the project may not be allowed to have constant up time	Computers running the server are used by other university students	Go to the computer that is running the server and check if processes are interrupted	Put up notices on the computer that it is currently in use	Project Team	3	6	Open	18	It will be improbable for the computer server to be interrupted if a notice is set but if the server fails everything will stop running
3	20	No permitted area for installation of webcams	Permission may not be given for webcams to be installed on the area	External	Privacy issues are the main consideration for the university	University does not allow any person to install video surveillance in the university	Display the project surveillance in terms of public presentation in the university and asks for participants' consent	Collaborate with security department on which area can be used	Project Team	2	3	Open	6	It is highly improbable for no areas in the university to conduct our project. Even if there are no areas, we can do it on other places
4	12	Video file corrupted	The video file that is streamed is corrupted and are unable to be accessed	Technical	Typing and spelling errors when inserting the directories of communication	When the directory or filenames of the data is incorrectly saved	Change the directory and check that everything is correctly named	Set up a standard for naming files and directories	Project Team	3	4	Open	12	Everything will need to be written correctly to provide the correct output during testing. Can be fixed very quickly once noticed
5	5	Project ignored by members in internship and external work	Project members will be in their internships locally and overseas for a few months at the end of the year	Schedule	University requires students to undergo internships to graduate	Members prioritise their internship work instead of the project tasks	Discuss and provide a flexible schedule for the team member and redistribute the workload	Set up a schedule that minimises work during internship periods	Project Manager	4	6	Open	24	Team members have committed and are responsible enough to fulfill their tasks. A slacking team member will cause others to do more work
6	15	Team members need to take time off to celebrate various festivities.	Various festivities such as Chinese New Year, Christmas, etc that are celebrated by team members.	Schedule	Malaysia celebrates too many festivities.	Arrival of festive seasons	Schedule activities around the festive seasons and prepare team members in case overtime is needed	No prevention methods needed	Project Team	2	5	Open	10	Team members can still do their work remotely during festivities. The schedule can consider the festivities but it will affect the progression
7	12	Supervisor busy to meet	Our project supervisor may not be able to meet with the team when needed.	Schedule	Supervisor is a busy man	Monash School of IT is understaffed	Find alternative communication methods as an alternative to face-to-face meetings	Arrange meetings ahead of time	Project Manager, Supervisor	4	3	Open	12	Project team just updates the supervisor, this can be done through other communication channels. Supervisor can always find time to respond via email
8	11	Project is delayed due to change in project requirements	Our client may decide to change project requirements midway through development.	Scope	Project scope is not clearly defined at the beginning of development	Client changes project requirements during development	Negotiate with the client on whether the changes are necessary and inform the clients of the impacts on the project if these changes are taken into account	Fix the project requirements before development starts	Project team, Supervisor	2	7	Open	14	All of the requirements have been clearly agreed so there will be less change. But a change will force a lot of new things to be done and redone
9	4	Failure of network to establish connections between our server and remote cameras	Network fails to connect server and remote cameras.	Technical	Network is not appropriately configured/not suited to handling the system	Network failure	Switch system over to backup network	Ensure network is correctly configured to handle the system and implement backups	Software Developers, Quality Assurance	4	7	Open	28	It is highly probable that the Internet will be disconnected in some time that the process is running. This will cause no data to be streamed and architecture will halt
10	15	Traffic accidents/Natural disasters	Occurrence of natural disasters or accidents that affect our system or team. E.g. floods, fire, road accidents, etc.	External	Acts of God	Unknown	Restore system from backups and find replacement team members if needed	Provide backup of data for our project	Project Team	1	10	Open	10	There are very small chances for accidents to happen to the team but if there is any serious accidents, it will have a high impact on the ability and morale of the team
11	3	Campus closedown	Campus is closed down and team members cannot enter the campus	External	Government policies to force closedown	Natural disasters such as pandemics	Utilize remote access software and update project requirements	Select project requirements that are flexible	Project Team, Supervisor	3	10	Open	30	There are chances like demonstration, natural disasters, pandemic and government policies that forces a closedown of campus which does not allow our project to have a live hardware setup
12	20	Database accessed by non authorised personnel	all data stored in database accessed by unauthorised user that it leaks private information	Security	the database that is used to store the data is leaked by unauthorised users	there is security gaps in accessing the data	Find other database provider that can provide more confidentiality	Encrypt every data before storing in the database	Project Team	1	6	Open	6	Database are fairly secured but if accidentally leaked to the public will cause a public outcry to the victims in the stream

Table 14: Risk Register part 1

No.	Rank	Risk	Description	Category	Root Cause	Triggers	Potential Responses	Prevention Methods	Risk Owner	Probability	Impact	Status	Score	Score Rationale
13	8	Software updates	when there is software updates, it may not support other related software. E.g. after updating Apache Kafka, it will not support our database provider that we should find another database provider	Technical	Incompatible support systems between multiple software	New versions of relevant software are released	Modify code in order to accommodate the new updates or rollback software to previous versions.	Delay updates until the system is ready	Software Developers	3	6	Open	18	There are always constant updates in open source software that will remove or add new functions. This might cause our current architecture to fail with deprecated functions
14	10	Overoptimistic assessment of workload	Team members has underestimated the amount of workload tasked to each member and this may cause problems in time management as well as completing the project before due date.	Schedule	Lack of understanding in project requirement scope and time management skills among team members.	Project manager misinterpret the abilities of the team members	Adjust the schedule and inform the client about the changes	Use a pragmatic approach when designing the schedule	Project Manager	3	5	Open	15	Project team members does not know the true abilities of each other and will always give more workload to them and cause incomplete tasks every deadline. This will be a slight impact to the project
15	6	Inadequate testing	Test plan for project is unreliable and software produces errors and bugs.	Technical	Fault in quality assurance	Incompetent test plan and test coverage	Refactor test cases and test plan to fix errors and bugs	Plan a high test coverage at multiple stages of the project to catch any errors early on	Quality Assurance	3	7	Open	21	The test cases are reviewed and tested before deployment and reduce error with fault tolerable software. However, an uncaught error will cause the architecture to fail
16	18	Legal actions	Legal actions can be taken by parties involved in the CCTV surveillance used in our project due to violation of their privacy.	Legal	People feel their privacy is violated	Installation of CCTV at questionable locations	Make official apology statement to parties involve	Get approval of permission from respective parties if necessary	University, Project Team, Supervisor	1	8	Open	8	People will be asked for their consent or told that they are being surveilled. If anyone is offended, legal courses might ensue that costs time and money
17	15	Theft of intellectual property	Outside parties plagiarising our hard work or stealing our content will results in theft of intellectual property(IP).	Legal	Dishonesty of outside parties	The credibility of our work is competitive	Legal actions taken involving violating rights of intellectual property	Content of our project should remain confidential among team members and clients until project finishes	University, Project Team, Supervisor	2	7	Open	10	Since the project are mostly using open source and free software, they will be published in an open source manner. However, the University property licenses will be considered
18	6	Inadequate performance of our software	The software we used did not perform as expected and doesn't match the outcome of the project.	Technological	Fault in software development team	Lack of understanding between software development team	Find alternatives to the software package	Ensure the software development team knows the software they're handling well	Software Developer	3	7	Open	21	Since the project is still in proposal phase, nothing has been built and tested. There might be a lack of understanding on the implementation of the project. This will cause a significant change in project progression if this is the case
19	19	Reduction of project scope	Clients reduces the workload of the project by removing part of the requirement scope.	Positive	Clients change their mind about the project requirement	Change of requirements from client	Discuss and update the project requirement scope with client accordingly	No prevention method needed	Project Team	1	7	Open	7	The project scope is clearly communicated by the team and the supervisor, it will not change very easily. The team will change their plans to accommodate for scope changes
20	12	Similar open source code found	Useful references of the software required by our project can be found in software development platform like GitHub.	Positive	Plenty of projects and tutorials online creating the architectures with same software	Tutorial is found online that works on the same problem	Reference the tutorial and better understand the capabilities of the architecture	No prevention method needed	Software Developers	4	3	Open	12	There are plenty of tutorials that are ready online. It will not impact that much as we will build the architectures ourselves
21	20	Software used remains competitive	Software used in our project remains relevant in the near future and is proof to be useful in this competitive age.	Positive	Well-designed and developed software	Competitors cannot developed better software	Maintain the quality of the software by fixing bugs	No prevention method needed	Software Developers	3	2	Open	6	The software used are competitive and the architecture is still new among the industry. It will not have much impact for this project
22	1	Restriction of movement of team members	Team members forced to move back to their homes which involves overseas	External	Government enforces a stay at home policy and movement restriction	Natural disasters or disease outbreak	Prepare remote access and update project requirements	Cannot be prevented	Project Team, Supervisor	4	9	Open	36	It is likely that some natural disasters would cause team members to go back to their hometowns that include overseas and outstation from the university campus. This forces the project to be done remotely
23	1	Power Outage and Hardware Inaccessibility	Remote access will require constant power and Internet access on the target computer on campus	External	Circuit and network maintenance on campus	Unstable power and inconsistent network receivers	Prepare instances that can be run at home computers	Cannot be prevented	Project Team, Supervisor	4	9	Open	36	It is likely that Monash will carry out maintenance by closing the power and network temporarily on campus. This will affect the accessibility and progress of the project when the team is working on important parts of the code

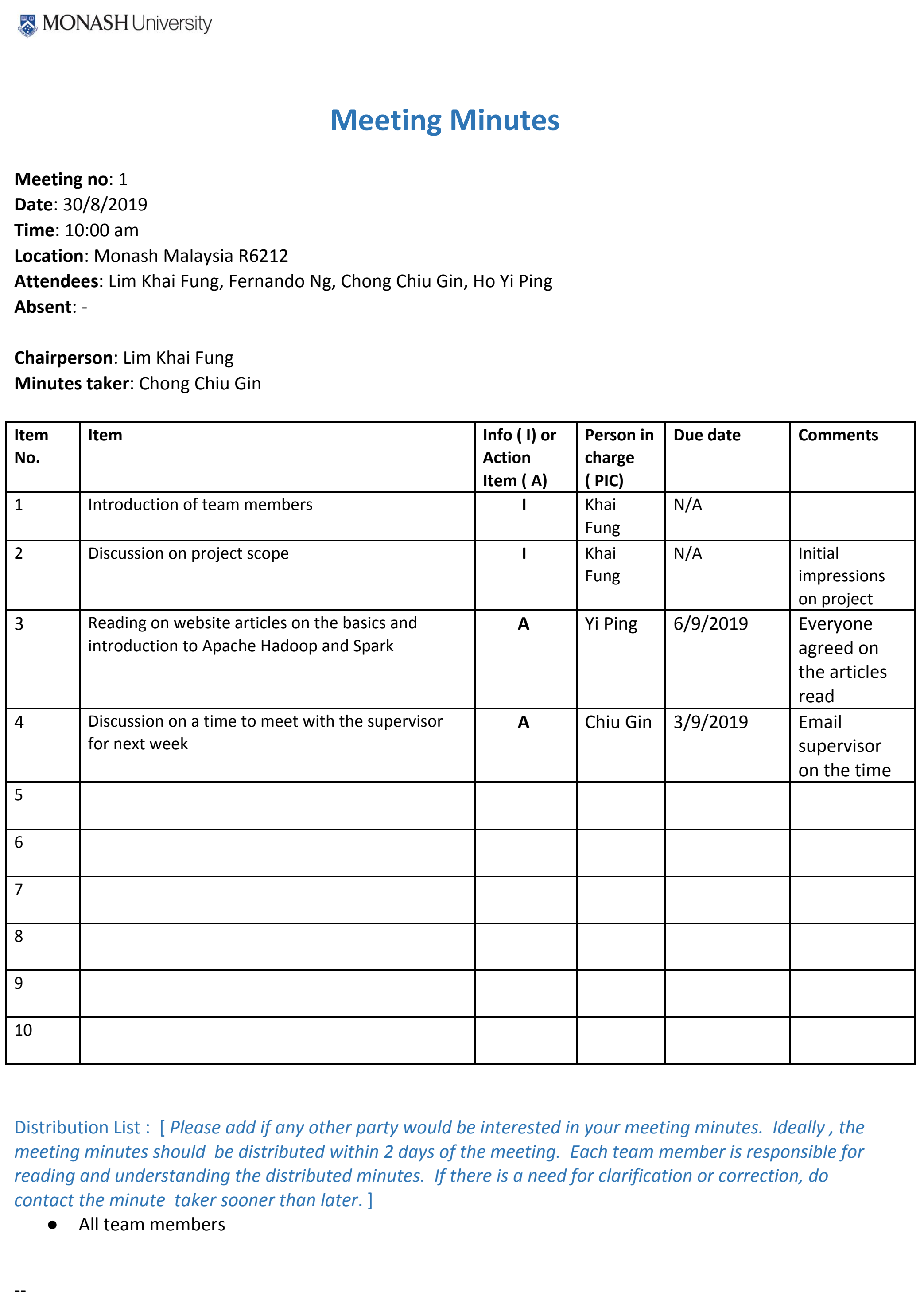
Table 15: Risk Register part 2

Stakeholder/Roles	**Interests**	**Impact/Influence**	**Contributions**	**Risks**	**Management Strategy**	**Responsibility**
Client	Product to be delivered on time	High	-Provide product requirements -Test product and provide feedback	-Client does not know what they really want -Client changes requirements drastically -Client adds many new requirements without extension	Constant communication and updates will be delivered to the client	Project Manager
Project Manager	-Project runs smoothly -Clear plan of execution	High	-Coordinates the project team -Manages the schedule of the project -Provide clear communication	-Overthinking and performing irrelevant tasks -May be unsympathetic in some cases	-Requires the support of the team members -Always communicate with client and team to stay on track	Project Manager
Lead Software Developer	-Product is well designed -Product runs efficiently	High	-Software design of the product -Code lead -Outcome analysis	-Communication difficulties between software developers -Unfamiliar software packages	-Constant communication and updates among team members -Provide extra time learning the required software packages	Project Manager
Software Developer	-Product is implemented efficiently -Product runs in a user interactive environment	Medium	-Coding and testing on product -Code report write up	Requires more time to understand certain topics and do their tasks	Good schedule management and work breakdown with explanation	Lead Software Developer
Quality Assurance	-Product has high fault tolerance -Product is bug free with error prevention measures	High	-Provide a test plan and report -Provide proof of concept of plan	-Quality of tests unreliable -Not all test cases can be thought of and considered	-Clear communication and reminder of the testing -Constant testing of different cases at every stage of the project	Project Manager

Table 16: Stakeholder Engagement Plan

Components	**Description**
Meeting Agenda	Meeting Agenda will be distributed 3 business days in advance of the meeting. The Agenda should signify each topic to be discussed. The first item in the agenda should be a review of action items from the previous meeting recorded in the previous meeting minute
Meeting Minutes	Meeting minutes will be distributed within 3 business days following the meeting. Meeting minutes will include the status of all items from the agenda and topics discussed along with new action items that are to be completed
Action Items	Action Items are recorded in both the meeting agenda and meeting minutes. Action items will include the action item, description of the action item and the owner of the action item. Meetings will start with a review of the status of all action items from previous meetings listed in the previous meeting minute and end with a review of all new action items resulting from the current meeting
Meeting Chairperson	The Chair Person is responsible for distributing the meeting agenda, facilitating the meeting and distributing the meeting minutes. The Chair Person will ensure that the meeting to start and end on time and the completion of all action items
Note Taker	The Note Taker is responsible for documenting the status and discussion of all meeting items and taking notes on important items during the meeting. The Note Taker will give a copy of their notes to the Chair Person within 1 business day after the meeting
Time Keeper	The Time Keeper is responsible for taking time to adhere to the time schedule given in the Meeting Agenda. The Time Keeper will alert the presenter when approaching the time limit. A board with the minutes left will be shown to the presenter

Table 17: Meeting Guidelines



MONASH University

Meeting Minutes

Meeting no: 1
Date: 30/8/2019
Time: 10:00 am
Location: Monash Malaysia R6212
Attendees: Lim Khai Fung, Fernando Ng, Chong Chiu Gin, Ho Yi Ping
Absent: -

Chairperson: Lim Khai Fung
Minutes taker: Chong Chiu Gin

Item No.	**Item**	**Info (I) or Action Item (A)**	**Person in charge (PIC)**	**Due date**	**Comments**
1	Introduction of team members	**I**	Khai Fung	N/A	
2	Discussion on project scope	**I**	Khai Fung	N/A	Initial impressions on project
3	Reading on website articles on the basics and introduction to Apache Hadoop and Spark	**A**	Yi Ping	6/9/2019	Everyone agreed on the articles read
4	Discussion on a time to meet with the supervisor for next week	**A**	Chiu Gin	3/9/2019	Email supervisor on the time
5					
6					
7					
8					
9					
10					

Distribution List : [*Please add if any other party would be interested in your meeting minutes. Ideally , the meeting minutes should be distributed within 2 days of the meeting. Each team member is responsible for reading and understanding the distributed minutes. If there is a need for clarification or correction, do contact the minute taker sooner than later.*]

- All team members

--

Figure 27: Sample Meeting Minutes

Components	**Description**
Kickoff Meeting	Project team will utilise Meeting Minutes and Meeting Agenda templates for meeting agenda and meeting minutes that is provided in Moodle
Project Team Meetings	Project team will utilise Meeting Minutes and Meeting Agenda templates for meeting agenda and meeting minutes. No additional slides are required in meetings
Project Report	Project team will follow a loose reference on IEEE Standard Format. Additionally all project documents and materials used will be in the shared drive.
Ad Hoc Formal	Project communications and project meetings has no standard template or format that must be used. Will be carried out either in person or through Zoom
Ad Hoc Informal	Communications through WhatsApp
Code Documentation	Documentation follow the best practices of JavaDoc
Task Delegation	Task delegated through Gantt Chart in TeamGantt and Trello

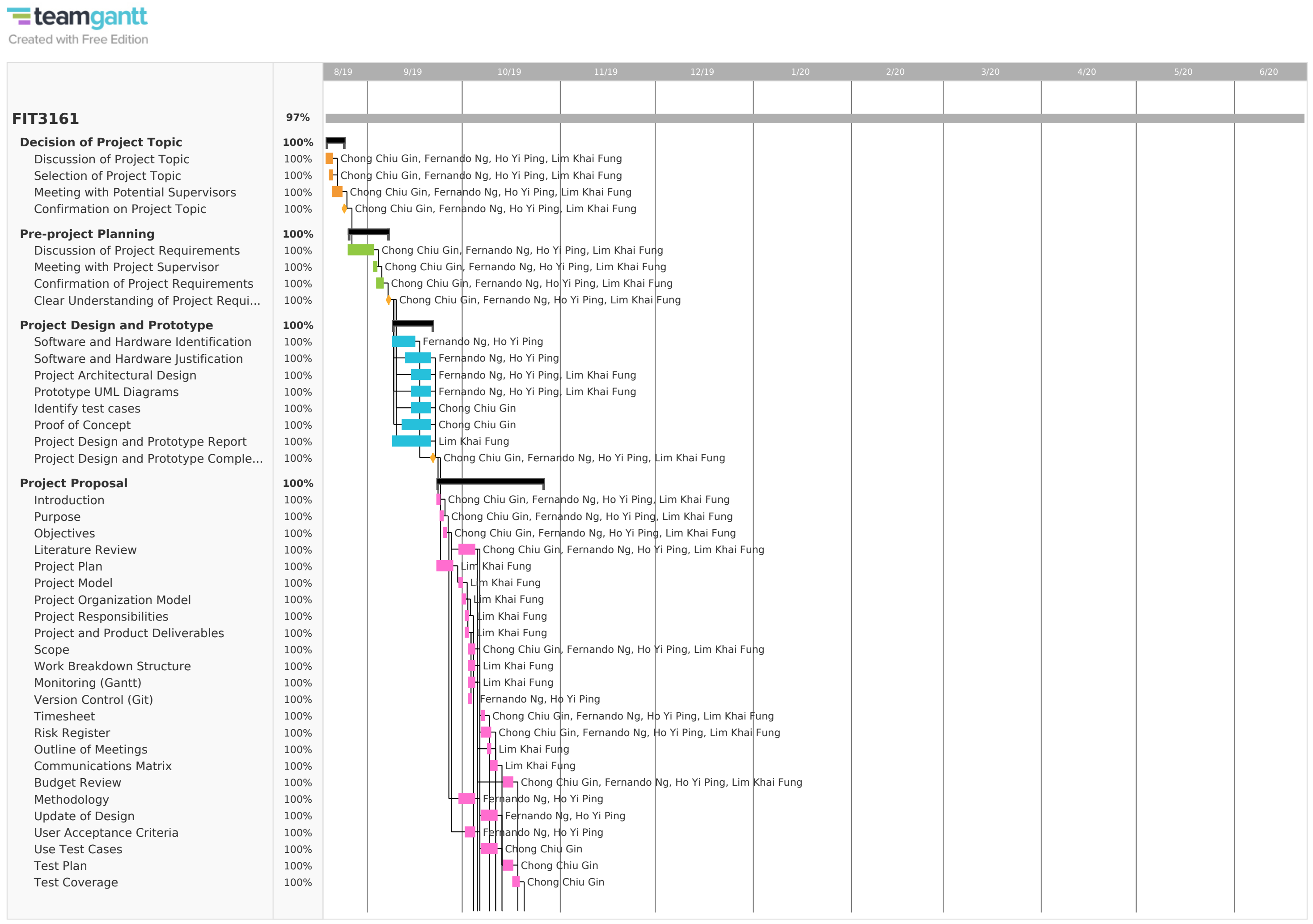
Table 18: Communication Standard

Communication Type	**Objective of Communication**	**Medium**	**Frequency**	**Audience**	**Owner**	**Deliverable**	**Format**
Ice breaking	Introduction of project team members	Face to face	Once	Project Team	Project Manager	-	-
Kickoff Meeting	Selection on project topic	Face to face	Once	Project Team	Project Manager	-Agenda -Meeting Minutes	Soft copy saved on Google Drive
Project Supervisor Meeting	Confirmation on project	Face to face	Once	-Project Supervisor -Project Team	Project Manager	-Agenda -Meeting Minutes	Soft copy saved on Google Drive
Project Team Meetings	To discuss and confirm the tasks that are required to be done for the week	-Face to face -WhatsApp -Email/Slack -Zoom	Weekly	-Project Supervisor -Project Team	Project Manager	-Agenda -Meeting Minutes	Soft copy saved on Google Drive
Technical Design and Report Meetings	-Discuss and confirmation of software and hardware components -High level design of project and product -Proof of concept and validation of design	-Face to face -WhatsApp -Email -Zoom -Slack -Teamviewer -Trello	As Needed	Project Team	Project Manager	-Agenda -Meeting Minutes -Software Requirements Document -Software Configuration -Proof of Concept Document -Code Report	All Soft copies saved on Google Drive
Weekly Status Reports	Update on Project Supervisor to ensure project on track	-Face to face -Email/Slack -Zoom	Weekly	-Project Supervisor -Project Team	Project Manager	-Agenda -Meeting Minutes	Soft copy saved on Google Drive
Urgent Team Meetings	-Clear out any confusions -Resolve any miscommunications	-Slack -Zoom -WhatsApp	Ad Hoc	Project Team	Project Team	-Project Presentation	-Videos Recorded stored on Google Drive
Project Clarification Meetings	Clear out simple questions on project	-Face to face -Email -Zoom -Slack	Ad Hoc	-Project Supervisor -Project Manager	Project Manager	-	-
Test Plan and Report Meetings	-Review test methods -Review test results	-Face to face -Email -WhatsApp	As Needed	Project Team	Quality Assurance	-Test Plan Document -Test Report	Soft copy saved on Google Drive
Programming Meetings	-Perform pair programming	-Zoom	Weekly during Implementation	Software Developers	Software Developers	-Code Base	Code pushed onto GitHub
Code Review and Reporting	-Review efficiency and structure of code	-Face to face -Zoom -WhatsApp	As Needed	Project Team	Software Developers	-Code Base -Code Report	Code pushed onto GitHub, Report saved on Google Drive
Project Plan and Team Management Review	-Review progress and feasibility of project plan	-Face to face -Zoom -Email -WhatsApp	As Needed	Project Team	Project Manager	-Project Management Plan -Project Management Deliverables -Team Management Report	Relevant documents are updated and saved on Google Drive
Post Mortem	-Perform reflections on project	-Zoom	Once	-Project Supervisor -Project Team	Project Manager	-All documents	All documents of project archived on Google Drive

Table 19: Communication Matrix

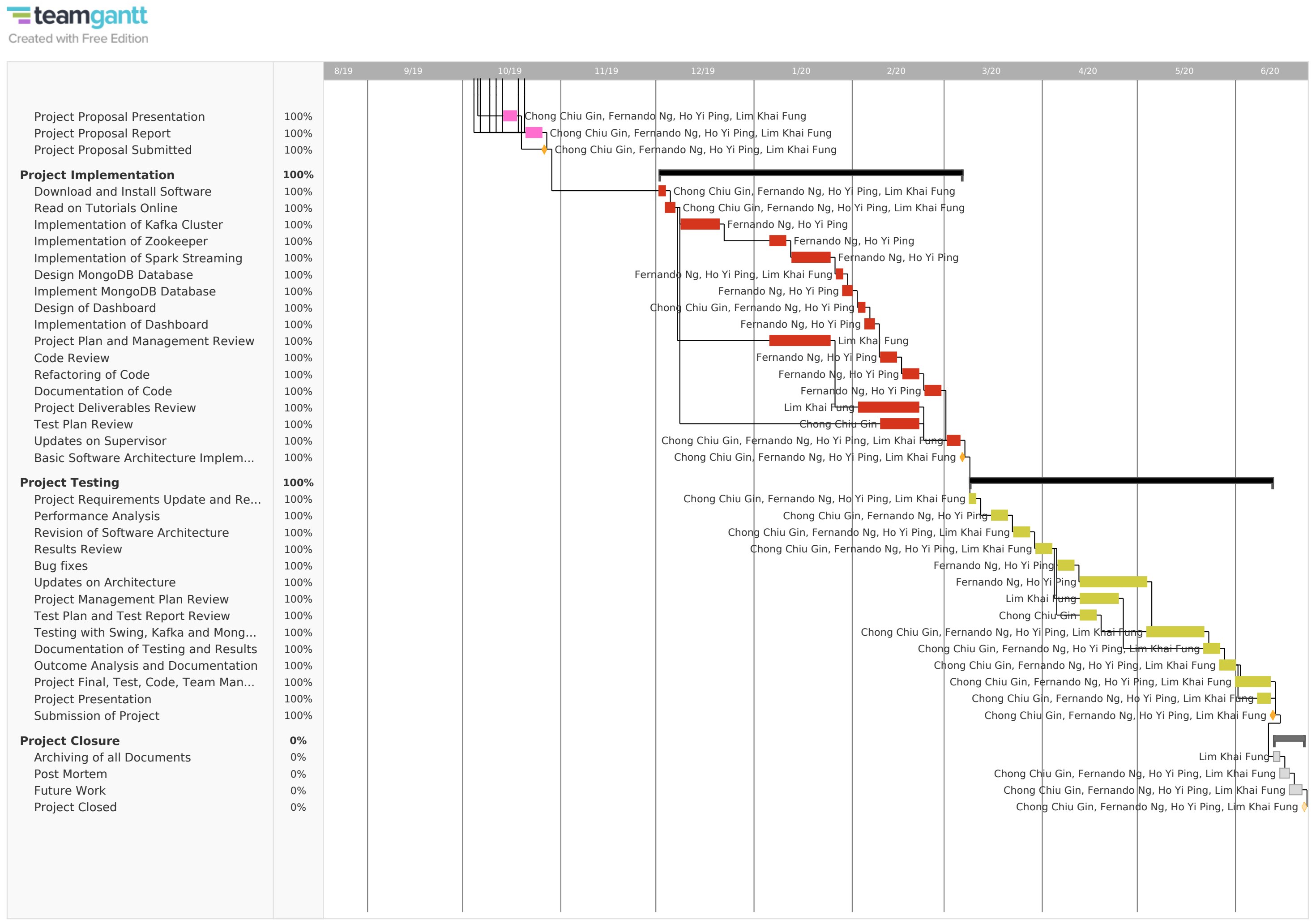
Priority	**Definition**	**Decision Authority**	**Time frame for Resolution**
1	Technical issues on project requirements and deliverables that will cause a major impact which will cause the project to fail.	Project Supervisor	Within three business days
2	Issues on project management processes and decisions that will cause the project to be delayed.	Project Manager	Within one business day
3	Issues on software design and implementation that may cause the project deliverable to not meet the requirements.	Lead Software Developer	Within three business days
4	Issues on the quality of the project deliverables that may have an impact on the project.	Quality Assurance	Work continues but workaround and fix will be required within five business days

Table 20: Communication Escalation Process



FIT3161	**97%**	
Decision of Project Topic	**100%**	
Discussion of Project Topic	100%	Chong Chiu Gin, Fernando Ng, Ho Yi Ping, Lim Khai Fung
Selection of Project Topic	100%	Chong Chiu Gin, Fernando Ng, Ho Yi Ping, Lim Khai Fung
Meeting with Potential Supervisors	100%	Chong Chiu Gin, Fernando Ng, Ho Yi Ping, Lim Khai Fung
Confirmation on Project Topic	100%	Chong Chiu Gin, Fernando Ng, Ho Yi Ping, Lim Khai Fung
Pre-project Planning	**100%**	
Discussion of Project Requirements	100%	Chong Chiu Gin, Fernando Ng, Ho Yi Ping, Lim Khai Fung
Meeting with Project Supervisor	100%	Chong Chiu Gin, Fernando Ng, Ho Yi Ping, Lim Khai Fung
Confirmation of Project Requirements	100%	Chong Chiu Gin, Fernando Ng, Ho Yi Ping, Lim Khai Fung
Clear Understanding of Project Requi...	100%	Chong Chiu Gin, Fernando Ng, Ho Yi Ping, Lim Khai Fung
Project Design and Prototype	**100%**	
Software and Hardware Identification	100%	Fernando Ng, Ho Yi Ping
Software and Hardware Justification	100%	Fernando Ng, Ho Yi Ping
Project Architectural Design	100%	Fernando Ng, Ho Yi Ping, Lim Khai Fung
Prototype UML Diagrams	100%	Fernando Ng, Ho Yi Ping, Lim Khai Fung
Identify test cases	100%	Chong Chiu Gin
Proof of Concept	100%	Chong Chiu Gin
Project Design and Prototype Report	100%	Lim Khai Fung
Project Design and Prototype Comple...	100%	Chong Chiu Gin, Fernando Ng, Ho Yi Ping, Lim Khai Fung
Project Proposal	**100%**	
Introduction	100%	Chong Chiu Gin, Fernando Ng, Ho Yi Ping, Lim Khai Fung
Purpose	100%	Chong Chiu Gin, Fernando Ng, Ho Yi Ping, Lim Khai Fung
Objectives	100%	Chong Chiu Gin, Fernando Ng, Ho Yi Ping, Lim Khai Fung
Literature Review	100%	Chong Chiu Gin, Fernando Ng, Ho Yi Ping, Lim Khai Fung
Project Plan	100%	Lim Khai Fung
Project Model	100%	Lim Khai Fung
Project Organization Model	100%	Lim Khai Fung
Project Responsibilities	100%	Lim Khai Fung
Project and Product Deliverables	100%	Lim Khai Fung
Scope	100%	Chong Chiu Gin, Fernando Ng, Ho Yi Ping, Lim Khai Fung
Work Breakdown Structure	100%	Lim Khai Fung
Monitoring (Gantt)	100%	Lim Khai Fung
Version Control (Git)	100%	Fernando Ng, Ho Yi Ping
Timesheet	100%	Chong Chiu Gin, Fernando Ng, Ho Yi Ping, Lim Khai Fung
Risk Register	100%	Chong Chiu Gin, Fernando Ng, Ho Yi Ping, Lim Khai Fung
Outline of Meetings	100%	Lim Khai Fung
Communications Matrix	100%	Lim Khai Fung
Budget Review	100%	Chong Chiu Gin, Fernando Ng, Ho Yi Ping, Lim Khai Fung
Methodology	100%	Fernando Ng, Ho Yi Ping
Update of Design	100%	Fernando Ng, Ho Yi Ping
User Acceptance Criteria	100%	Fernando Ng, Ho Yi Ping
Use Test Cases	100%	Chong Chiu Gin
Test Plan	100%	Chong Chiu Gin
Test Coverage	100%	Chong Chiu Gin

Figure 28: Gantt Chart Part 1



Project Proposal Presentation	100%	Chong Chiu Gin, Fernando Ng, Ho Yi Ping, Lim Khai Fung
Project Proposal Report	100%	Chong Chiu Gin, Fernando Ng, Ho Yi Ping, Lim Khai Fung
Project Proposal Submitted	100%	Chong Chiu Gin, Fernando Ng, Ho Yi Ping, Lim Khai Fung
Project Implementation	**100%**	
Download and Install Software	100%	Chong Chiu Gin, Fernando Ng, Ho Yi Ping, Lim Khai Fung
Read on Tutorials Online	100%	Chong Chiu Gin, Fernando Ng, Ho Yi Ping, Lim Khai Fung
Implementation of Kafka Cluster	100%	Fernando Ng, Ho Yi Ping
Implementation of Zookeeper	100%	Fernando Ng, Ho Yi Ping
Implementation of Spark Streaming	100%	Fernando Ng, Ho Yi Ping
Design MongoDB Database	100%	Fernando Ng, Ho Yi Ping, Lim Khai Fung
Implement MongoDB Database	100%	Fernando Ng, Ho Yi Ping
Design of Dashboard	100%	Chong Chiu Gin, Fernando Ng, Ho Yi Ping
Implementation of Dashboard	100%	Fernando Ng, Ho Yi Ping
Project Plan and Management Review	100%	Lim Khai Fung
Code Review	100%	Fernando Ng, Ho Yi Ping
Refactoring of Code	100%	Fernando Ng, Ho Yi Ping
Documentation of Code	100%	Fernando Ng, Ho Yi Ping
Project Deliverables Review	100%	Lim Khai Fung
Test Plan Review	100%	Chong Chiu Gin
Updates on Supervisor	100%	Chong Chiu Gin, Fernando Ng, Ho Yi Ping, Lim Khai Fung
Basic Software Architecture Implem...	100%	Chong Chiu Gin, Fernando Ng, Ho Yi Ping, Lim Khai Fung
Project Testing	**100%**	
Project Requirements Update and Re...	100%	Chong Chiu Gin, Fernando Ng, Ho Yi Ping, Lim Khai Fung
Performance Analysis	100%	Chong Chiu Gin, Fernando Ng, Ho Yi Ping
Revision of Software Architecture	100%	Chong Chiu Gin, Fernando Ng, Ho Yi Ping, Lim Khai Fung
Results Review	100%	Chong Chiu Gin, Fernando Ng, Ho Yi Ping, Lim Khai Fung
Bug fixes	100%	Fernando Ng, Ho Yi Ping
Updates on Architecture	100%	Fernando Ng, Ho Yi Ping
Project Management Plan Review	100%	Lim Khai Fung
Test Plan and Test Report Review	100%	Chong Chiu Gin
Testing with Swing, Kafka and Mong...	100%	Chong Chiu Gin, Fernando Ng, Ho Yi Ping, Lim Khai Fung
Documentation of Testing and Results	100%	Chong Chiu Gin, Fernando Ng, Ho Yi Ping, Lim Khai Fung
Outcome Analysis and Documentation	100%	Chong Chiu Gin, Fernando Ng, Ho Yi Ping, Lim Khai Fung
Project Final, Test, Code, Team Man...	100%	Chong Chiu Gin, Fernando Ng, Ho Yi Ping, Lim Khai Fung
Project Presentation	100%	Chong Chiu Gin, Fernando Ng, Ho Yi Ping, Lim Khai Fung
Submission of Project	100%	Chong Chiu Gin, Fernando Ng, Ho Yi Ping, Lim Khai Fung
Project Closure	**0%**	
Archiving of all Documents	0%	Lim Khai Fung
Post Mortem	0%	Chong Chiu Gin, Fernando Ng, Ho Yi Ping, Lim Khai Fung
Future Work	0%	Chong Chiu Gin, Fernando Ng, Ho Yi Ping, Lim Khai Fung
Project Closed	0%	Chong Chiu Gin, Fernando Ng, Ho Yi Ping, Lim Khai Fung

Figure 29: Gantt Chart Part 2

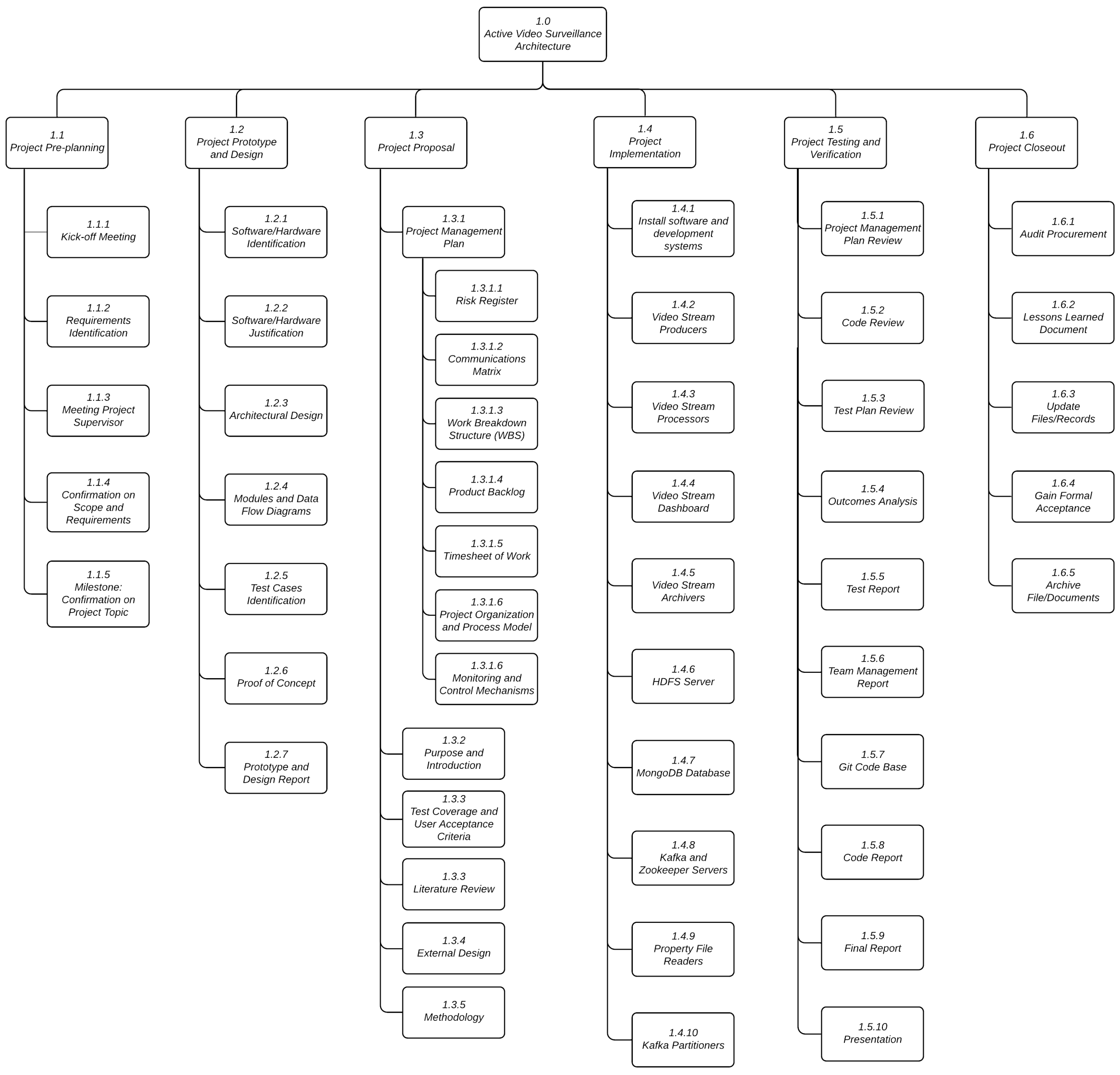

1.0 Active Video Surveillance Architecture
1.1 Project Pre-planning
1.1.1 Kick-off Meeting
1.1.2 Requirements Identification
1.1.3 Meeting Project Supervisor
1.1.4 Confirmation on Scope and Requirements
1.1.5 Milestone: Confirmation on Project Topic
1.2 Project Prototype and Design
1.2.1 Software/Hardware Identification
1.2.2 Software/Hardware Justification
1.2.3 Architectural Design
1.2.4 Modules and Data Flow Diagrams
1.2.5 Test Cases Identification
1.2.6 Proof of Concept
1.2.7 Prototype and Design Report
1.3 Project Proposal
1.3.1 Project Management Plan
1.3.1.1 Risk Register
1.3.1.2 Communications Matrix
1.3.1.3 Work Breakdown Structure (WBS)
1.3.1.4 Product Backlog
1.3.1.5 Timesheet of Work
1.3.1.6 Project Organization and Process Model
1.3.1.6 Monitoring and Control Mechanisms
1.3.2 Purpose and Introduction
1.3.3 Test Coverage and User Acceptance Criteria
1.3.3 Literature Review
1.3.4 External Design
1.3.5 Methodology
1.4 Project Implementation
1.4.1 Install software and development systems
1.4.2 Video Stream Producers
1.4.3 Video Stream Processors
1.4.4 Video Stream Dashboard
1.4.5 Video Stream Archivers
1.4.6 HDFS Server
1.4.7 MongoDB Database
1.4.8 Kafka and Zookeeper Servers
1.4.9 Property File Readers
1.4.10 Kafka Partitioners
1.5 Project Testing and Verification
1.5.1 Project Management Plan Review
1.5.2 Code Review
1.5.3 Test Plan Review
1.5.4 Outcomes Analysis
1.5.5 Test Report
1.5.6 Team Management Report
1.5.7 Git Code Base
1.5.8 Code Report
1.5.9 Final Report
1.5.10 Presentation
1.6 Project Closeout
1.6.1 Audit Procurement
1.6.2 Lessons Learned Document
1.6.3 Update Files/Records
1.6.4 Gain Formal Acceptance
1.6.5 Archive File/Documents

Figure 30: Work Breakdown Structure

Software	**Description**	**Link**
Ubuntu Linux 18.04.3 (LTS)	Operating System	https://ubuntu.com/download
Java Version 8 Update 221	Programming Language	https://java.com/en/download/
Java SE Development Kit 8u221	Programming Environment	https://www.oracle.com/technetwork/java/javase
IntelliJ IDEA Version 2019.2.2	Java IDE	https://www.jetbrains.com/idea/
Apache Maven 3.6.2	Software Project Management Tool	https://maven.apache.org/
Apache Kafka 2.3.0	Distributed Streaming Platform	https://kafka.apache.org/
Apache Zookeeper 3.5.5	Centralized Server	https://zookeeper.apache.org/
OpenCV 4.2.0	Computer Vision Library	https://opencv.org/releases/
MongoDB 4.2.7	NoSQL Database	https://www.mongodb.com/try/download/community
GitHub	Version Control	https://github.com/
TeamGantt	Project Schedule	https://www.teamgantt.com/
Overleaf	LaTeXEditor	https://www.overleaf.com/
Google Sheets	General Editor	https://www.google.com/sheets/about/
WhatsApp	Instant Communication	https://www.whatsapp.com/
Google Drive	Document Storage	https://www.google.com/drive/
Lucidchart	Diagram Drawing	https://www.lucidchart.com
Trello	Kanban Board	https://trello.com/
TeamViewer 15.6.7	Remote Control	https://www.teamviewer.com/en/download/linux/
VirtualBox 6.1	Virtualization Software	https://www.virtualbox.org/
Slack	Communication Tool	https://slack.com/intl/en-my/
Zoom	Video Communication Tool	https://zoom.us/

Table 21: Software Used