ARCHITECTURE FOR AUTOMATIC SOURCE CODE COMMENT GENERATION

R.M.N.S Samarasinghe

(199360G)

This dissertation was submitted in partial fulfilment of the requirements for the degree of MSc in Computer Science Specializing in Software Architecture

Department of Computer Science and Engineering

University of Moratuwa

Sri Lanka

June 2022

DECLARATION

I declare that this is my work. This Thesis Report does not incorporate without acknowledgement any material previously submitted for a degree or Diploma in any other University or institute of higher learning and to the best of my knowledge and belief. It does not contain any previously published material written by another person except where the acknowledgement is made in the text.

Also, I hereby grant to the University of Moratuwa the non-exclusive right to reproduce and distribute my thesis, in whole or in part in print, electronic or other media. I retain the right to use this content in whole or interest in future works.

....

R.M Nuwan Shashika Samarasinghe Date

I certify that the declaration above by the candidate is accurate to the best of my knowledge and that this project report is acceptable for evaluation for the Thesis Report.

....

Dr. Indika Perera Date

(Supervisor)

ABSTRACT

This research tries to give a highly available software architecture to a source code commenting tool. Code commenting is helpful for many phases like code updates, bug fixes, etc. Developers need to know which task each code is performing, and also, adding comments for each code is essential. But most developers take more time to add comments which has created the problem of wasting time. Using an automated comment generator tool for source code, we can avoid that problem and save time for another task. Currently, there is not a properly defined architecture for an automated comment generator tool. Therefore, we are trying to give an evaluated architecture for that tool that supports overcoming the problem.

In this architecture, we plan to follow the microservice architectural concepts. Then modularization of each component based on the service it is going to provide. Also, plan to use ActiveMQ for the queue technique, Scheduling techniques to reschedule things, No SQL databases for data saving, Caching techniques to store temporary data, etc. This procedure ensures that it will provide a highly available architecture. Also, this architecture will have the following quality attributes: Maintainability, Performance, Interoperability, Usability, Availability, Reliability, Testability, Modifiability, Scalability, and Reusability. Finally, we use presentations, proof of concept, and ATAM & CBAM methods to validate whether the architecture is commercially viable or not.

Key Words: Development, Zookeeper, ActiveMQ, REST, Database, Queue, Asynchronous, Cache.

ACKNOWLEDGEMENTS

My profound gratitude goes to Dr Indika Perera, my supervisor, for the knowledge, supervision, advice, and guidance provided with his expertise throughout making the thesis a success.

My appreciation goes to my family for the motivation and support provided throughout my life. Also, I would like to thank my colleagues in the MSc batch and at my workplace for the help and support provided in managing my research work.

TABLE OF CONTENTS

1 INTRODUCTION

1.1 IMPORTANCE OF CODE COMMENTING

Every software project that was created should have the maintenance phase. It is a critical element of a thriving software development environment. These maintenances will fix the bugs that have been identified during the testing or the actual usage. Furthermore, there will be new customer requirements, and some components might not be needed anymore. Therefore, it is essential to have the maintenance phase that removes those unwanted features and adds new features to enhance performance. In maintenance, code commenting will reduce the developer's effort tremendously [1]. According to IEEE [2], maintenance is the most crucial factor in software engineering. Different authors have stated that 60% of the cost is allocated for maintenance. Some critical components of the maintenance cost are an annual hosting fee, software licensing, defect resolution, UI/UX upgrades, Code refactoring, etc. Code commenting is a direct feature that can reduce maintenance costs in certain areas.

The concept behind the comments in the programming is to give some explanatory information about the created source code. These comments will be used to document the program and provide a reminder to the programmers of what they have done in the source code. Also, those will help the later generations to code or maintain the same source codes. All the compilers will consider these statements as non-executable statements. There are a lot of programming languages, and each of them uses different kinds of commenting techniques. Using comments will explain something that the programmers did during the development [1].

FIGURE 1: APPLICATION MAINTENANCE LIFE CYCLE

1.2 IMPORTANCE OF SOFTWARE ARCHITECTURE

When creating software, we need to consider several vital aspects, such as the system's requirements, which can be divided into two parts: functional requirements and non-functional requirements. Secondly, what are the components of the system, and what are each component's functionality and behaviour? How is each component communicating with the other? Thirdly, what external dependencies or services will be used, guidelines & risks seen in the software project, technological aspects, etc.? Through software architecture, we can see the following three main reasons that are very important. First, Software Architecture is a Basis for communication, and it is a plan of the system primordial for understanding, negotiation, communication between stakeholders. Also, it will give a piece of complete proper knowledge about the system. Second is the earliest decisions, and all the decisions are made at this stage. Because in this stage, it can change or update the decisions in very advance without impacting the system later. The third is the transferability of the model. Using this software architecture can give a broad idea about the component functionality, which will help estimate the time and cost needed for this software [2].

1.3 PROBLEM

The main problem of this research is to create proper architecture for code commenting software with architecture validation, as mentioned in the above introduction. Code commenting is an important feature. Without that, the projects can survive, but when comments are there, it will dramatically reduce the maintenance costs of a project. There are plenty of components that can generate code comments using various technologies such as Natural language processing, Machine learning, etc. When going through these components, none of them is available for industrial usage. They have been created to cooperate with specific needs or are only research components without any practical implementation. Therefore, in this research, I am presenting a high availability architecture for the commenting service. In this research, several problems might occur during the implementation process. Those are as follows:

1. What will be the language that is going to support in the first and research phase?
2. We are selecting a good code commenting tool.
3. Identify components in the commenting tool whether there is any main component that can be separated and added as services.
4. What are the other components that are needed for the application?
5. What type of architecture can be used for each component?
6. How to build one software as a service that can be used?
7. How to validate the generated architecture out of the validation methods?

Throughout this research, we will find suitable answers for each question, which will create a high availability software as a service architecture. With the final implementation, the developers do not need to make comments as it will automatically generate comments for them. It will directly support reducing the development and maintenance cost.

1.4 MOTIVATION

The motivation around the project is to create better software as a service for code comment generation. Also, this architecture can be extended into any other application because we are trying to give an optimized architecture by using minimum new components. Furthermore, it will use existing components to generate comments. Going through code commenting will benefit many areas, such as any update or bug fix for the project unless it may seem unwanted during the development. It is the cause that we are trying to reduce that unwanted time from the development time.

Doing architectural research means it will be entirely new with existing components and methods. No two architectures will be the same, and there may be some slight differences. Also, any architecture cannot be wrong if the software is working as expected. But when trying to give a suitable architecture, we need to consider many non-functional and functional requirements as required. New concepts can be learned, and we can apply software architecture concepts to a practical scenario. After generating the architecture, it will need a validation method to check whether we have added all the aspects and confirm that this architecture is the most suitable for this type of work.

While going through these aspects, I have decided to create a code commenting software architecture by using some existing and newly built components. In the implementation phase, architecture will be implemented using AWS services to implement the architecture with minimum cost, including the validation of the architecture.

1.5 OBJECTIVES

The following are the objectives of this research.

- To learn about the latest and existing technologies that can be used to generate software architectures. Also, further research has been carried out using various tools, hence understanding the techniques they have used.
- Get and go through relevant documentation to get more information about architectural design decisions because these decisions may vary according to each situation. Therefore, we need to go through each scenario and identify whether any possibilities can fail our application.
- Use the architecture and software-related knowledge to generate an architecture and demo application for the code comments.
- Give a reusable architecture component with proper documentation.
- Create an architecture validation method to evaluate the quality attributes.

2 LITERATURE REVIEW

This section will go through each research that has been done to create a code comment component and some of the architectural concepts that are found and can be implemented in this research.

2.1 SOFTWARE CODE COMMENTING COMPONENT

Code comments will be used for a lot of purposes. One is to do the planning and to review. In this technique, we write a pseudo code before creating the actual source code for the logic. That helps the reviewer to review the code quickly because pseudocodes are closer to natural language.

```
Ex:

// GCD BY EUCLID'S ALGORITHM

/* Euclid(m, n)
{
      while m does not divide n
      r = n mod m
      n = m
      m = r
      end
      return m)*/
```

FIGURE 2: PSEUDOCODE COMMENTING

The second is Code description. In there, the programmer intends to summarize what he has done in the code segment.

```
Ex:
{
 /* A Multiline Comment -- Define a
variable of string type and assign
value to it*/
    string msg = "GeeksforGeeks";
}
```

FIGURE 3: CODE COMMENT AS A PARAGRAPH

The third is giving algorithmic descriptions. This section of commenting will contain some mathematical proofs or diagrams to clarify a segment of code. This may or may not contain some natural language explanations.

```
Ex:
list = [ f(b), f(b), f(c), f(d), f(a),
... ];

// Need a stable sort. Besides,
// the performance really does not
matter.
insertion_sort(list);
```

FIGURE 4: MATHEMATICAL CODE COMMENT

Fourth is resource inclusion, where developers add logos, diagrams, and flow charts with ASCII arts constructions. Those will be able to add to the source code to give proper explanations and readability. Also, this will consist of the copyright notices for the source codes.

Metadata of the codes will be the fifth purpose of the source commenting. It contains the name of the creator of the original version, the current maintainer of the program, the date when the first version was created, the name of the people who have edited the program files so far, etc.

For the debugging, also programmers use comments. Programmers are adding code statements to get some values printed (for testing purposes) in the runtime. After doing them, it will be commented. But commented code in source code is not a better practice to have.

```
Ex:
int fun(int m) {
    int count = 0;
    while (m > 10) {
// printf("m is less than 10, m=%d",
m);
        count++;}
    return m;
}
```

FIGURE 5: CODE COMMENT

Another most popular reason to use comments is to generate automatic document generation. Most of the programming environments will create document metadata as a comment. These will have insert positions for automatic header file inclusion, commands to set the file's syntax highlighting mode, or revision number. These comments will refer to annotations. [3], [4] The different programming language has different kind of code commenting techniques such as,[5]

- PHP: // This is a Comment in PHP
- Python: # This is a comment in Python
- Java: // This is a single Line Comment in Java /* This is a multiline comment. in Java */

- SQL: -- This is a comment
- XML: <! -- This is a comment in XML-->
- CSS: /* This is a comment in CSS */
- HTML: <! -- This is a comment in HTML-->
- JavaScript: // This is a comment in JavaScript.

This research focuses on giving a suitable architecture for the code comment software, but we use Java programming language for the demonstration. This language has three kinds of commenting techniques single line comment, multiline comments & documentation comments. Most beginner-level programmers use this single-line comment to describe the core functionality of the basic level of commenting in the java environment.

```
Ex:
//Java program to show single line
comments
class Scomment
{
    public static void main(String
args[])
    {
         // Single line comment here
         System.out.println("Single
line comment above");
    }
}
```

FIGURE 6: SINGLE LINE CODE COMMENTS

In Java, a multiline code describes a complex code in multiple lines because some codes may not be defined within a line.

```
Ex:
//Java program to show multi line
comments
class Scomment
{
    public static void main(String
args[])
    {
        System.out.println("Multi line
comments below");
        /*Comment line 1
          Comment line 2
          Comment line 3*/
```

FIGURE 7: MULTILINE CODE COMMENTING

In the documentation, comments programmers used to give metadata about methods present, their parameters, etc. Also, there are many annotation tags in Java to create java docs to represent things like the params and the return type.

This research will give a better architectural infrastructure for automatic code commenting by using better components and libraries. There are currently plenty of components that can generate comments for the given code. Code comment is the main component of this architecture and several other components used in this research. First, we go through the comment generation component [6].

In the past, researchers have done many things around automatic code commenting. We can divide them into several categories like the Template-based method and the Keyword-based method.

2.2 CHALLENGERS OF AUTOMATIC CODE COMMENTING

When generating the source code comment generator, a few steps need to follow. The first is to create a source code model to express the structure, lexical, grammatical, semantic and context features. After generating these, they will be processed to develop natural language comments. In the first step, we just generated the structure of the code comment, but in the second step, it will convert to a more readable comment. Finally, these generated comments need to evaluate then we can decide that the comments re-correct for the scenario. Code has more information about the classes, methods, parameters and logic, but the comments will have more natural language sentences that express the same meaning that code is giving. These types of automated comment generators will get the following challengers.

Automatic commenting algorithm

Currently, there are a lot of algorithms which is going to generate code comments. Some of them are automatic & some of them are partially automatic. Here, it will divide into three main categories: information retrieval based, the deep neural network, and finally, the automatic comment generation algorithm. Following are the automatic code commenting issues. The source code model is one of the critical issues in that algorithm.

1. Abstract Syntax trees
2. Parse tree
3. Token context
4. Control flow graphs
5. Data flow

These methods can be classified into two sets. One is a token-based source, code models. This will tokenize the code and extract the keywords in the given code. This collection of words is called a bag of words. The second type of model is the syntax-based source code model. These will try to model the source code into a syntax tree. This type of model is used in the NLP techniques as well.

Text generation is the second issue that can face while using the algorithm natural language is unstructured, and the expression form is very flexible. Still, the

problem is that before generating the comment's algorithm, it should extract all the information from the source code to develop the natural language comment.

Comment quality assessment

In the code comment generation, quality assessment is one of the other critical problems because it is tough to check the quality of the comment generated. We can divide the problem into two first is the unification of the dataset used to validate and test the comment algorithm. Then what are the criteria that are going to use for the evaluation?

While going through the automated code comment algorithms, we can classify them into several categories: template-based and keyword-based. Also, there are other classifications too. First, we'll go through the following classifications & the algorithms, and then we'll go through the different classifications in detail. With this approach, we can show the most excellent algorithm that can be used for our approach.

2.3 CLASSIFICATION TYPE 01

2.3.1 TEMPLATE-BASED TECHNIQUES

Few researchers used this technique to generate comments for the given language. One of the leading teams is Giriprasad Sridhara, and his team had members from different universities and different countries such as India, the USA, and Colombia. Their technique is to generate comments summaries of the content rather than relationships of each class. The comment generation used the method stereotypes by combining some heuristic methods to select the words for the summaries. Finally, they generate summaries with lexicalization. Following are the steps to create comments in this method.

1. Determine the stereotype of the class and methods used to determine the summary information.
2. Then to generate the summaries, they use some conjunction with predefined heuristics with the above stereotypes.
3. Then it goes through some predefined ruleset to generate natural language phrases.

When we go through the method deeply, we can identify the algorithm used to generate the comments. They have selected some names that give the idea about the class like following parent classes, interfaces, and inner classes (if any); and its attributes and methods but do not contain any relationships. After selecting the exclusive content, it is divided into two sets, and one can add directly to the summaries, and the other needs further analysis before adding to the summary. In the data extraction process, they use stereotypes because they contain the role or responsibilities of each class method, etc. Once all are selected to generate the comments, then it has four parts

1. General description based on the interfaces, superclass, and the stereotypes of the classes.
2. Description of the structure based on stereotypes.
3. Description of the behaviour based on enumerating its most appropriate methods.
4. List of inner classes if exists.

For the research evaluation, they have used 22 graduate students with industry knowledge and java language knowledge to check the created comments based on the following criteria Content adequacy, Conciseness, Expressiveness. In this evaluation, they have made 40 summaries. According to them [7], 69% of the summaries do not miss important information, and 96% of the summary are concise and understandable.

The above research has some limitations, but it was the basis for all of the other studies. One of the significant limitations is it guarantees the comment generation for some code structures only—for example, one method body or group of method calls. Also, the performance is highly dependent on the high-quality identifier names and method signatures. To overcome these, Edmund tried to use Stack Overflow to generate comments. Because Stack Overflow contains many questions and answers written by developers, using those can generate more accurate comments for the code snippets. In Stack Overflow, there are nearly 5,509,302 posts as of August 2013. Hence, there will be at least one answer there. Their approach used the NLP technique to improve Stack Overflow's comment in their comment generation technique. They have used the following methods to achieve the success criteria.

1. We are proposing to search on Q&A sites to get proper comments for the code segments.
2. Evaluate the auto comment using 23 projects with 102 comments and check whether those are accurate, adequate, concise, and valuable.
3. Get an NLP logic to improve the quality of the comments.
4. Build a code description mapping database to leverage in the future.

While answering the above questions, they have faced several other challenges like Comment selection and description refinement. In the first challenge, they have used the following approach. First, when the system gets a code segment, they choose the most relevant post, and then it needs to determine the most suitable sentence out of the post. They filter out the comments based on the word types and do some keyword filters. Finally, to improve that, they used an NLP technique to refine the sentences [8].

Another outstanding research was done in the same related manner by McBurney and Collin McMillan from the USA. They built a system that uses NLP concepts to generate comments for the java class. In their implementation, they have made some assumptions like if the rendered content has more information from the java method content, it will be accurate and effective. Hense to do that, they must answer the following questions.

1. What method do you use internally?
2. Why does the method exist?
3. How to use the method?

They used a novel technique to generate the documents and used 12 programmers for the evaluation. For that, they have created two individual summary docs. One contains the generated summaries and the other consists of the generated comments with state-of-the-art. Summaries based on those programmers confirmed that those generated comments are precious and understandable. In the development phase, they have used some key components. The first key component is Software Word Usage Model (SWUM). It converts the program statements into a set of nouns, verbs, and prepositional phrases, and not only that, but it also extracts noun phrases such as "public id" and creates a relation noun to verbs. The second component is the Natural Language Generation System (NLG). It was an invention of another research done by Reiter and Dale. It can be divided into three other components. The first one is a Document planner. It needs some facts, and then it generates some messages that humans can read. The second component is the Micro planner. It decides what word will be used to describe each message. There is a sub-component called the Surface realizer. It generates the natural language sentences from the given phrases. The third principal component is Page Rank. It ranks the importance of the comments. These components and the evaluation have proved that they have done the summary generation [9].

2.3.2 KEYWORD-BASED TECHNIQUES.

Sonia Haiduc and her team have conducted research to generate automatic codes based on statistical text retrieval and summarization. This framework has two components. The first is to extract the text from the source code and convert it to the corpus. The second is to determine the most relevant terms for documents in the corpus and then include those in the summarization.

After implementing the proposed system, they used two code bases to generate summaries. The first is ATunes (media player code), and another is Art of Illusion (3D modelling code). They have chosen a few categories of methods, such as less than 20 lines, between 20 to 50 lines, and more than 50 lines. Then they used flowing techniques to generate the summaries lead, VSM, LSI, then evaluated the summaries with some developers by ranking the autogenerated with the developer's idea [10].

Laura Moreno and his team researched the same scenario to get automated comment generation. Their technique tried to select the best suitable methods by looking at the methods stereotype and other access levels of the methods. Also, these generated comment summaries consist of four parts: the general description based on the interfaces, superclasses, and stereotypes of the classes. Second, description structure based on the stereotype; third, the description of the behaviour based on enumerating its most suitable methods and last if any inner classes exist remainder section describes in detail steps of summery generation.

They evaluated the method within three categories as Content adequacy, Conciseness, and Expressiveness[7].

2.4 CLASSIFICATION TYPE 02

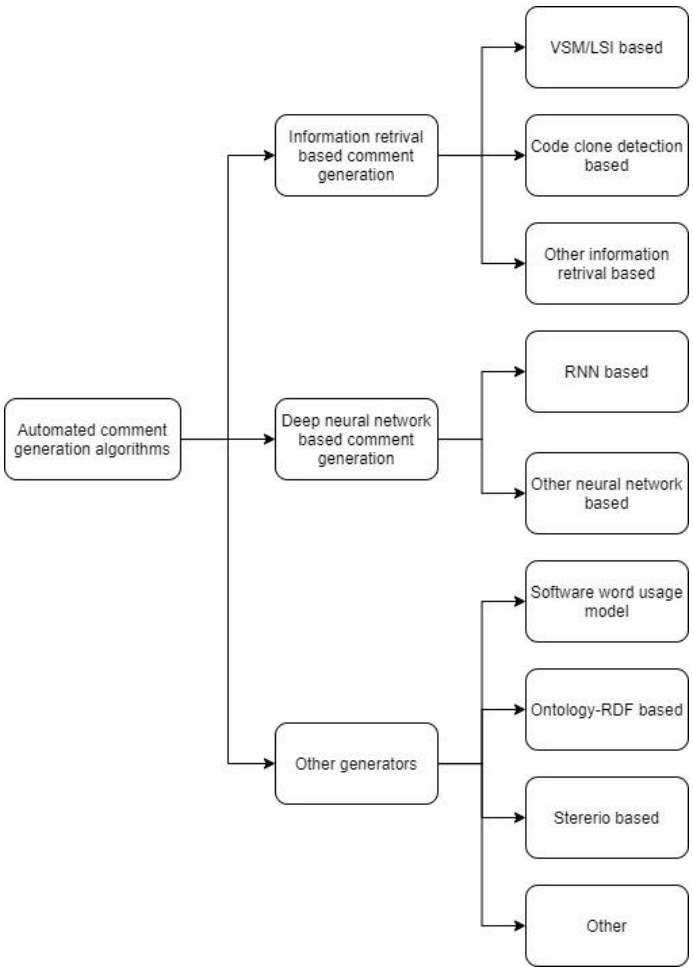


FIGURE 8: CLASSIFICATION TYPE 02 BREAKDOWN

2.4.1 VSM/LSI BASED COMMENT GENERATION ALGORITHMS

VSM's expanded meaning is Vector Space Model, and then LSI means Latent Semantic Indexing. When creating comment generators, most of the research has used either one of them or both hybrid algorithms, but both are information retrieval-based algorithms. Following will be the usage of those two methods. Source code or query tests are usually represented as vectors, matrix or tuples. Each element in the vector will represent the weight of a word in the document. So, to calculate the weight, there are many methods such as VSM and term frequency-inverse and document frequency. This is the most widely used method. Then this LSI method will recognize the relevancy between the terms and the concept. Finally, these will extract the conceptual topic from the text then, according to the weight calculation, topics will be chosen from the algorithms. Following are the researches have done according to the up given method. [11], [12]

In the above classification, we have given the details about the following research, but here it is more about the usage of the algorithm. So, here once they have generated the matrix of the word collection. Finally, they have used the VSM technique to get the most used terms in the code. These documents are selected based on the weights. They have used the LSI technique to check the similarity between the corpus and the vector and the document that will summarize. These selected tests will generate a more meaningful full comment for the given code segment. [10]

And in the following research, which is done by Vassallo et al., also used a VSM model to generate code comments but in a different method such as source code text with developer question and answer text in stack overflow be as vectors. They have calculated the cosine similarity between the target source and the discussion source on that website, so based on that similarity value, they have decided the comment for the source code. So, they have used the method called crowdsourcing. [13]

Panichella et al. Is also a researcher who has done a similar comment generation. His method uses heuristic and Vector space model processes and analyzes developer communications for the method description. These developer communications mainly refer to the emails and bug reports related to the classes, methods, and parameters. Here also checks for higher similarity to stand for the code comment for the given code segment. [14]

This method has the following drawbacks because it only considers the terms that appear in the source code. Still, while creating a comment, we must make sure that it takes program invocation, data dependency, and words sequence in the source code.

2.4.2 CODE CLONE DETECTION-BASED COMMENT GENERATION

This method mainly uses code clone detection algorithms. So, it will find similar code in given databases. Then, corresponding comments for the matched code or discussion text are taken as the comments for the codes.

Wong et al. Is also a researcher who has used a source code clone detection mechanism to detect the comment. He has used stack overflow as a comment database. Once given the stack overflow as the database, it will build a database, and for the given code snippets, it will check the similarity using the code clone detection & based on the value, it will select the correct comment out of the dataset [8]

This type of algorithm has an issue because these comments depend highly on the external Datasource, and that source may sometimes be wrong.

2.4.3 LDA BASED COMMENT GENERATION

Latent Dirichlet Allocation (LDA) is mainly used to extract topics from a given document. It also uses probabilities while generating the topics from the given document. N-gram is the most widely used method while generating the topics from the documents, and this is an NLP algorithm. [15]

Movshovitz-Attias and Cohen are also two other researchers that tried to build an LDA method to generate comments & in this method. They also used an n-gram to predict the comments for java source codes. They trained an n-gram model with the same source code but with different data set models. From that model, they have computed the posterior probability of document topics. [16]

Rahman et al. Analyzed a Stack Overflow Q&A site discussion to recommend inciteful comments for the open-source projects. They have adopted a heuristic method to generate the comments. [17]

2.5 OTHER INFORMATION RETRIEVAL-BASED

Oda et al. tried to create a method that would generate comments. She used Is using a Phrase-Based Machine Translation (PBML) and tree to string machine translation (T2SMT). These methods will generate pseudo-code from the python application. These two methods are statistical machine translation frameworks. So, if we have pseudo code, we can easily understand the logic behind the scene, so, with these two methods, researchers were able to generate pseudo-code from the given source code. Allamanis et al.also build a probabilistic model to create comments here. He has developed a probability model concerning the NLP expression and the source code. [18]

2.6 DEEP NEURAL NETWORKS-BASED COMMENT GENERATION

In the deep neural network, three kinds of algorithms can be used for comment generation. First is the Convolutional neural network (CNN), Recurrent neural network (RNN) and Recursive neural network (RvNN). Then when coming into the code comment related neural networks, we can divide that into two different structures first is encoder-decoder based, and another one is attention-based.

Encoder-Decoder framework

Encoder and decoder are also named sequence models. This encoding will do the following task: encoding the source code into a fixed-size vector and the decoder handles decoding the source code vector. Also, it will predict the comment for the given snippet. These can choose the following algorithm to work RNN, CNN, and other RNN such as Gated recurrent unit (GRU) and extended short-term memory model (LSTM).

Attention mechanism

Attention framework is also added to the same encoder & decoder framework, but the difference is the weight method in the attention method. This attention method was created by Bahdanau et al. This is best for the mediocre performance, which is long sequences. [19]

2.6.1 RNN BASED COMMENT GENERATION

This RNN is wildly used in deep learning, and it was a viral algorithm. Also, as mentioned, the above attention methods are more to improve the accuracy of the comment system. Other than this, attention-based, there are more variants, such as the Long short-term memory model (LSTM) and Gated recurrent unit (GRU). This LSTM is very capable of learning long term dependency information. This LSTM structure will create a special kind of neuron that is a controlled memory neuron that will solve the gradient descendent and gradient explosion in the traditional RNN method. If we check the drawback of the algorithm, LSTM is a complex structure that is needed to maintain complicated realization and lower execution efficiency. Then if we check the GRU method, it will just use two gates. First is the update gate, and the second is reset.

2.6.1.1 SINGLE ENCODER-BASED COMMENT GENERATION

Iyer et al. Proposed an LSTM based code comment generation model, and the project was Code-NN. This was to generate summaries for C# and SQL source codes. This model was created based on the Stack overflow. In this data set, they have used the title and the code segment pairs. [20]

GRU is another encoder and decoder framework that can generate comments for the source codes. Zheng et al. Used this method to generate the comments for this research. So, their attention module in the encoder is a global attention method, and they take some embedding symbols in code snippets as a learnable then. Also, these will have prior weights to evaluate the importance of different parts of the sequence. This attention method can understand the structure of the code better. [21]

Hu and Li researchers used AST sequence source code generation using Structure-based traversal (SBT). They have used the sequence to sequence model and attention method for the comment generation for this method. DeepCom is their project that proved this concept of code commenting. After training the neural network model, it can learn the semantic information from the source code. Then those will be used for the comment generation. [22]

A recursive neural network (RvNN) can represent the parse trees of natural language sentences. Lian et al. Used this method for this comment generation, so in this algorithm, he combined the semantics and the structural information with vectors. Then used the recurrent neural network translator (Code-GRU) decoded the vectors while generating the comments. So, his overall framework is to generate textual descriptions, and his accuracy is more remarkable than other researchers. [23]

2.6.1.2 MULTIPLE ENCODER-BASED COMMENT GENERATION

In the multi encoder-based comment, generations are using two or more encoders and decoders in the implementations. Also, these methods are a bit more accurate than the methods mentioned above. Hu and Li are some of the researchers that used this method, and they used automatic API summaries to generate the comments. And they have used two algorithms first is API encoder, and another one is code encoder. This API encoder uses an RNN based encoders recorder model. [24]

Wan and Zhao are the other two researchers who tried to use multi encoder-decoder method for the comment generation. They have used the LSTM model to represent the sequential information on the code, and there is another AST based LSTM model to represent the structure of the source code. These are two classical encoders frameworks. [25]

2.6.2 OTHER NEURAL NETWORK-BASED COMMENT GENERATION

Seq2seq is one of the models that come under this category, and it will have RNN based encoder-decoder structure. Not only RNN, but we can use CNN also for the job. In this CNN method, the nice thing is it can parallelize the computation during the training process. Also, it can represent the syntactic and semantic information on the source code.

Allamanis et al. Introduce a convolutional network into an attention mechanism to create short descriptive summaries for the source code snippets. In this system, the fundamental element of decoder is GRU. The convolutional attention module uses the input source to detect the pattern and determine the critical token sequences. [26]

Mou and Li et al. Also used a convolutional neural network to generate the syntax tree of a source code as a vector representation. Also, this classified the program according to the functionality. In their system, they take the whole AST programmed as an input. Then they designed the convolutional kernel over this AST of source code to capture the structural information of the source code. Still, their intention was not to generate comments that were their future work on this implementation. [27]

2.7 QUALITY EVALUATION OF GENERATED COMMENTS

In this section, we are going through some methods that can evaluate the generated comments are correct & are those comments illustrate the right meaning of the source code. Following is the research is done to find those methods.

Khamis et al. Created a java Doc Miner, the tool that can analyze the quality of the java doc comments automatically. In this method, they have used a heuristic way to confirm the generated comment in quality language. Daniels Steidl et al. Also performed a study on the same case to identify a better-quality method to classify the comments into several categories such as copyright comments, header comments, member comments, and inline comments, section comments, code comments and task comments. Finally, they have given each comment a quality check model, and their main focus was to check the quality of the inline comments and the member comments. Then Yu et al. also researched finding a quality matrix to check the code comment quality. He has used aggregation classification algorithm, which will evaluate comments based on the format, language form, content, and the correction degree of code. [28]–[30]

2.8 COMPONENT ARCHITECTURES

The first section clearly describes the code commenting component and how the component has grown in the past decade. When analyzing that section, it is clear that no single method gives a proper software solution that can be used practically now.

2.8.1 TEXT FILE READ (I/O OPERATIONS)

This section contains the component of the file read. After creating the code, each code segment needed to be offloaded to the online system. For that, the application requires a small tool. Before building that component, it is necessary to identify the techniques and programming languages that can be used to get a faster file read component. This section will go through one-by-one research which has been done to achieve the same situation.

The Iraq researcher read an XML file through various programming languages and used two different platforms to test the results. He has used six programming languages C#, Php, Java, Python, and Perl, for the testing. Windows and Linux were the platforms that he used as these two operating systems have different kernel architectures. Through that, the researcher can check whether there is any influence behind the operating systems. Programming languages use all the native OS-level functionality like I/O operations. Therefore, it was an excellent decision to use the OS for this research. To reduce the system's memory usage, they have used the terminal and the DOS for the program execution. For the analysis, he has used the following versions.

Windows		Linux	
The Compiler	**The Version**	**The Compiler**	**The Version**
Visual C# 2008	3.5	Mono	4.0.12
g++	5.3.0	GUN g++	4.8.4
Jdk javac/java	1.8.0_73	Jdk javac/java	1.7.0_95
PHP	5.6.15	PHP	5.5.9
Python	3.4.4	Python	2.7.6
Perl	5.22.1	Perl	5.18.2

FIGURE 9: BEST PROGRAMMING LANGUAGE FOR I/O

Not only the memory he has considered but also the CPU usage. As per the research, Perl got an enormous execution time among the six programming languages. C# was the least time taken to execute, but the java windows version took less time.

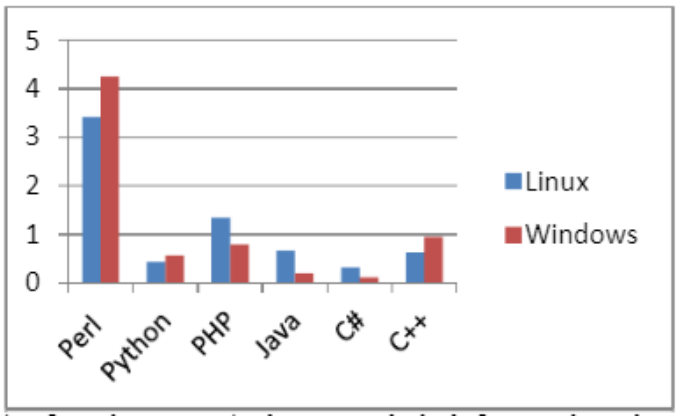


FIGURE 10: I/O TIME TAKEN FOR WINDOWS & LINUX (CPU)

According to the research [31], when considering memory usage, Linux java has the most prominent memory usage, and C++ has the least memory among these two platforms. Still, the C# Linux version also has some considerable use of memory.

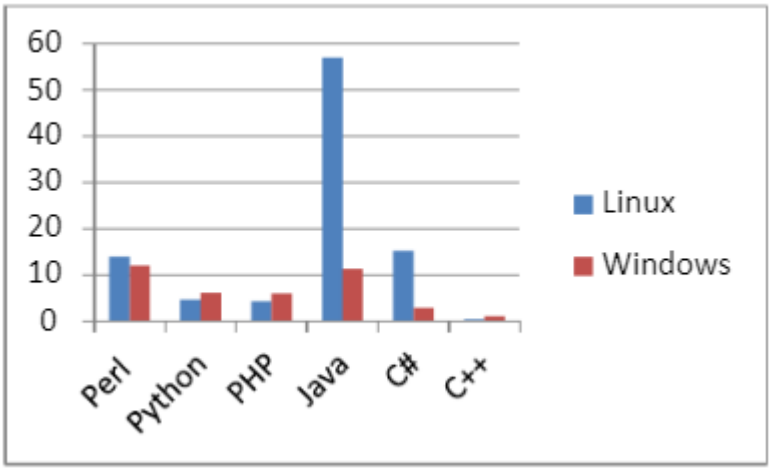


FIGURE 11: I/O TIME TAKEN FOR WINDOWS & LINUX (RAM)

When considering the CPU usage, all the five programming languages except Java Linux got the same CPU usage. Therefore, according to this research [31] C++ language has less resource consumption among any OS platform.

Another research has been performed to find whether there is any way to use Java programming language to perform extensive data files. For that, they have used Federal Elections Commission's data set to test the algorithms' performance. To compute that, he has used the following criteria. First, to get each name from the 8th column and add those to a list, try to print 432nd and 43243rd names of the list and other criteria to identify standard terms. To do that that he has used three methods which are in Java.

01. FileInputStream () & Scanner ()
02. BufferedReader () & FileReader ()

03. Apache commons FileUtils.LineIterator()

During the performance evaluation, he used a 2.55GB text file to perform the test & got the following results from the research.

Solutions	FileInputStream	BufferedReader	LineIterator	Percentage Improvement
Line Count	261,803 ms	152,961 ms	150,694 ms	73.73%
Names at XX Index	261,826 ms	152,965 ms	150,696 ms	73.74%
Donations Per Month	262,806 ms	154,318 ms	151,644 ms	73.30%
Most Common First Name	265,113 ms	156,253 ms	153,333 ms	72.90%

FIGURE 12: PERFORMANCE EVALUATION FOR I/O READS

According to this research [32] java Line, the Iterator has the best performance to read files.

A set of students and researchers from the Perking University of China, University of Monash Australia, and the University of Singapore has researched to generate code comments for java applications based on Neural network-based implementation. It is going through each method and trying to get its idea, which means the feature that will be implemented. Finally, it generates the relevant comment based on the learned feature. The team of researchers has done this with 9714 sample data on git hub projects. This DeepCom approach consists of three different stages. The first is data processing, the second is model training, and the third is online testing. Before adding the java code into the given model, then convert that to an AST (Abstract Sequence Tree) sequence. It is done by using some unique traversal approach. Then it sends to the training model, and finally, it gets the code comments. They have used the sequence-to-sequence model, primarily used for machine summarization, text summarization, dialogue systems. The encoder learns the codes the user and the Attention component keeps track of the words, especially like "whether," "if," etc. In this project, they have analyzed automated code comments and user-generated code comments as well. As per this analysis, they have achieved some significant improvements throughout the research. But also, they have missed the software architecture that they are supposed to introduce. [22].

2.8.2 SOFTWARE ARCHITECTURE DESIGN

This research will not give a new code commenting tool but tries to use an existing tool to create a proper architectural design for the code comment. When considering each related research, They have archived many things and have developed many methods to make code comments. But none of them has created a proper software architecture to introduce the tool as a solution. This research tries to generate a software architectural model that can be a support for any other researchers. It can use when they want to add their component to a suitable place, not only that any researchers can extend this idea to any component. In this research, I have gone through many software architectures like microservices, service-oriented architecture, etc. In the implementation stage, we need to support many more clients like browsers, IDEs, etc. Then more and more components will be created based on this solution, and also, many IDE providers will use this by making their plugins. If we use microservices, there will be some pros and cons, such as application components will be loosely coupled. Solution providers can quickly put their solution (components instead) and deploy separately without any downtime to the entire solution. Furthermore, each component can be owned by some small amount of developers [16],[17],[35].

In the following research, they have surveyed to identify the best architectural solution that can be used. Also, they have conducted 800 more surveys. They can get the following advantages from the research: Loosely coupled components that ensure independent development and deployment. Front-end integration will give the ability to generate more applications without using core UI components. Another most important thing is that resource management of the application will reduce the complexity. Also, these services can be configured for a continuous health check monitoring system. If we only have a component-based solution, they can do exactly when the service is down (Failure recovery). This enables the users to do continuous integration and deployments (CICD); likewise, there are many technical benefits. When we go through the cost benefits of the microservices architecture, it does not need more training. All the developers do not need to know all the things of the domain; they only need to focus on its area, which will reduce the overall cost of the application development. Migration from one to another is also straightforward because we are using a conceptual diagram to create the microservice and no component dependency among each other [34], [36], [37].

In this research, they have identified some challenges while converting the project to microservices. The first one is modelling. Most of the time, developers cannot remember the individual layers of the MSA ecosystem. It becomes more complex when we try to convert a monolithic system to an MSA system. Some models can use Microservice business entity models, Microservice API models and inter

microservice communication pattern models, and deployment strategy specifications. After this modelling process, the next one is the development process. It has the following processors such as user-defined microservices layer. It implements the business logic of the proposed application. The second is the development of infrastructure microservice. This ensures accessibility, availability, and durability and needs to monitor each microservices in this development layer. The third is the development of inter-microservices communication patterns. This covers up all the communication-related things such as message passing etc. Likewise, in this research, they have gone through most of the challenges that can occur—some researchers from the University of Novi Sad Serbia [35], [38].

Then we are going to work with performance analysis. It needs to know the essential capabilities. This site says that they will have three more steps to work on. Following are the changes, like starting the application in a monolith way. Another one is to build microservices.

A team of researchers from the University of Duhok in Iraq tried to create a framework for the performance analysis of microservices. They used the Kieker framework for the performance evaluation. When we go through microservices, it has a lot of benefits like individually scalable, etc. But in this research, they tried to answer the following: how can we monitor Microservice and analyze the system's performance. Therefore, they have done the following. First, they have used some microservice-based applications made of the following frameworks Spring, Jolie, Play. Kieker is a framework that was built by the apache software foundation. That has two main components, one is monitoring, and the other component is analysis. This monitoring component does the following tasks such as program instrumentation, data collection, and logging. It uses some measurement probs for the instrumentation of the software and writers for collecting data. The analysis component is used to analyze the data and visualize the data. Using it, they have evaluated each framework's microservice performance [39].

2.8.3 DOCKER ENVIRONMENT

Docker is a platform that we can deliver things quickly without any dependency on the infrastructure platform. This is an open platform with the capability of developing, shipping, and running applications. In the docker environment, they give the platform independence by using containers running on top of an OS, but it can handle many environments. This docker entire platform gives some tools to manage the container life cycle.

These docker environments are streamlined to CI/CD processors. Using those will enable fast delivery of the applications and no environmental boundaries while developing. Also, some images are ready to run state components. No additional

configuration is needed. This environment also supports dynamic scaling based on the requests count (heavy load). They can automatically get the new containers running. This cannot be done with only docker. There should be Kubernetes on top of docker to do this functionality. Also, docker enables you to run more services on a single PC without an issue because it can share the resources as needed.

Docker environment uses a client-server architecture to form the service. Docker client passes the commands to the Docker daemon that is the one responsible for controlling the docker container life cycle. Both systems can run in the same environment no need to be separated. This docker client & the daemon is going to communicate via REST calls and Unix sockets. There is another client called docker-compose that will enable the user to create multiple containers and can manage different services in each one config file (YML)

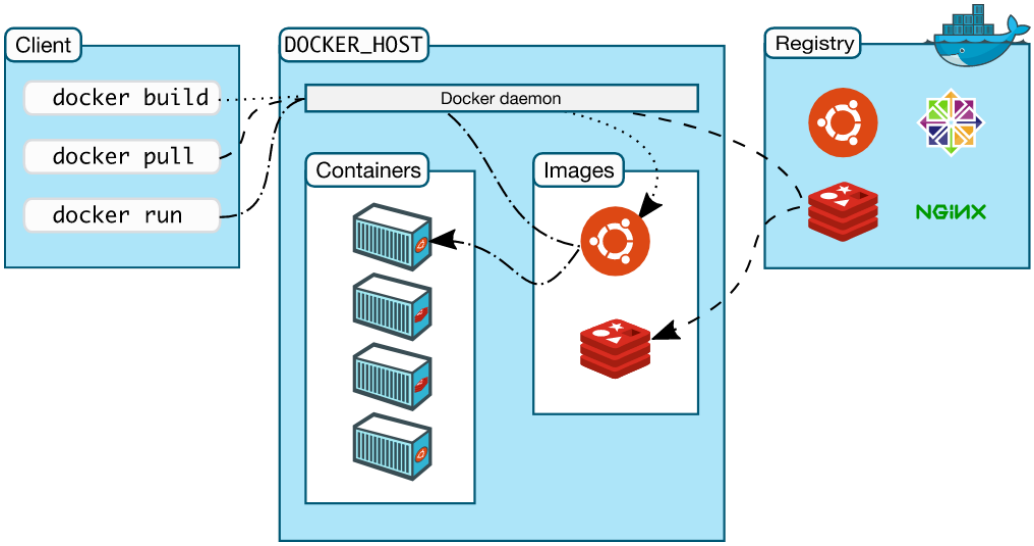


FIGURE 13: DOCKER ARCHITECTURE

Docker daemon: This docker daemon will listen to the API requests from the clients, and then it will manage the containers, images, volumes not only that they can communicate with each other daemon and manage the cluster of docker environment without an issue.

Docker client: This will be the human interaction component, and it will generate the requests to the daemon according to the given commands. Also, this client can interact with multiple daemons as well.

Docker registry: docker registry is the image repository that we have on the internet & it allows users to store images publicly or privately. Still, some other registries are also enabled not only that we can even host our registry to manage the docker images.

Docker images: docker images are mainly based on another image & we will do the necessary configurations to start the application on top of those images. For the image configuration, we need to use the Docker file. They have some specific language styles & commands to do the configuration.

Docker containers: docker container is a runnable version of the image. It will create the environment that needs to run the application.[40]

2.8.4 DOCKER SWARM

When we go through docker, it has an inbuild capability to automatically manage the environment with multiple container support, which means scalability that software is swarm kit. This swarm will have multiple docker hosts who will run as swarm mode. In that mode, those will act as managers & workers to do the given task. While creating the docker swarm service, following configurations such as we must use several replicas for the defined benefit to provide the high availability and then we must create a network to support the load balancing and in that we must expose a port to use that will be used to create the clustered environment, and then in the service it needs to make an endpoint to expose that service to the outer world.

One of the best advantages of the docker swarm service is modifying the service end configurations without any manual restart of the docker services. It will just apply whatever changes are given in the service config file. Even if the docker is running on the swarm mode, we can run some other docker containers in the same docker environment. Docker swarm is the first step to the Kubernetes environment where we cannot do any dynamic configurations. It cannot configure automated container creation while the app is getting more load. Still, it supports multiple containers to run the same service & it will manage all the load balancing parts among each container.

What are the components in the docker swarm?

Nodes:

Node is an instance of the Docker engine that in the swarm environment. Running multiple nodes is possible in any of the docker environments currently also. This can be distributed on several computers as a distributed system. A manager for these nodes will take the service request given by the YML or other configuration method and initiate the worker's nodes to start the application.

Services and tasks:

Service is a set of workflow that will give to the manager then. It will do one by one tasks and start the service in a node. In this task, we must specify the image that we are going to use and any other environment variables & other config-related

changers. In the same service, there is another component called replicated service. It will handle the replicated clusters from the other service. It creates one environment, and then this service will replicate the environment configured in the configuration document. Task will execute the run commands in the docker image configuration.

Load balancing:

The docker swarm uses an ingress load balancer to expose the services outside of the swarm network. This swarm manager will manage the port assignment if we are not explicitly given it in the configuration file. Also, the internal swarm system has its own DNS with load balancer to manage the service internally that was used to manage the internal traffic among the nodes. [41]

2.8.5 KUBERNETES

Kubernetes is a portable and extendible open-source software, and Google manages this. This is based on the Docker environment that will help the system owners to auto-scale & scale own the system. Not only that, but it also gives additional more features. Following are the architectures that we have passed during the decade

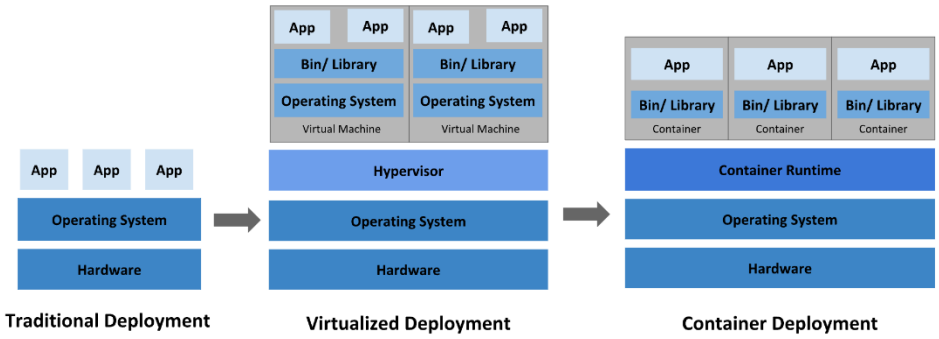


FIGURE 14: KUBERNETES ARCHITECTURE

Traditional deployment era:

In the era, we have used primarily physical service. In that physical server, we can run multiple services and there we cannot define any resource limitations to each service that running on that location due to that reason, it sometimes causes cascade failures because if one service keeps getting increased resources from the physical service, others might get a smaller number of resources that cause that failure. Also, if we needed to scale up the resources, it was not easy. We must do a lot of migrations, and many underneath hardware compatibilities are required to be checked while updating the system.

Virtualized deployment era:

In this era, we have virtual machines on top of the physical servers. From this approach, we were able to isolate services in one server and, they can give more security than running on the physical PC since the VMs separate other applications in the physical servers resource issue also solved out because of this approach, we can keep adding those till the physical server can handle the changes, but still, virtual servers need more resources we have to update the physical server's resources to increase virtual PC's resources. And this virtual machine is already an OS running machine. It has all the capabilities that a physical device has.

Container deployment era:

These containers are like VM's, but they share the OS among the applications that mean it has minor things that need to run the application, so these are lightweight. These containers have their file system to manage the applications also. These are decoupled from the underline infrastructure. Following are the benefits that containers are going to provide.

1. Agile application creation and deployment: increased efficiency on image creation than using VM images
2. Continuous development, integration and deployment: this will increase productivity, and clients will get the product immediately
3. Dev and Ops separation: create application container image rather than deployment time, decouple the infrastructure.
4. Observability: from this container approach, we can both watch the host sever stats and the container stats.
5. Environment consistency among the environments.
6. Application-centric management
7. High resource utilization

What Kubernetes can give?

Container bundles applications and runs them in the given containerized environment. Following are the services that can get with Kubernetes environment

1. Service discovery and load balancing: if one pod is getting more load, then it will give the load to the other pods dynamically
2. Storage orchestration
3. Automated rollouts and rollback: based on the state, it can create new pods with the given conditions automatically and instantly
4. Automatic bin packaging: we can give how many resources that need to take when creating a pod, and Kubernetes will manage those resources
5. Self-Healing: It automatically restarts or creates the pods if a dead pod unexpectedly

6. Secret and configuration management

The following cannot get from Kubernetes

1. There is no limitation for the application support. It can run a variety of workloads. Also, it can include stateless and stateful apps, data processing workloads.
2. No builds and deploy the code: CICD is not available
3. No application-level services like message bus, MySQL, or any other can be started as services in pods, but no such services are included by default.
4. No logging alerts are there in the solution but can add externally to monitor the health etc.
5. No machine configurations such as maintenance, management, self-healing
6. Kubernetes is not an orchestration system [42]

2.8.6 LOCAL STACK

Local stack is a cloud service emulator that is going to emulate the AWS services mainly. Using this will simulate all the services we are using in a cloud environment with this stack even we can even use terraform scripts to generate the environment. This stack can emulate the following AWS services: Lambda, S3, DynamoDB, Kinesis, SQS, and SNS. Also, in the pro version, it gives a lot more support. It can start on the local environment in a docker environment. [43]

3 METHODOLOGY

This research tries to present a solution and support for the developers when creating documents for the given projects. Most of the developers spend much time on this task as it is one of the essential jobs. Because if the developer who developed the system is currently not there, other developers need to go through the code and do the bug fixing or do dome enhancements for the system. The developer needs to know the business logic behind each class to update the relevant needs. Therefore, it's conducive if we have the comments for each code segment. The developer can manually write those. It is the most accurate way, but it takes more time than expected, and project clients do not care about those practices. They need the projects on time. Therefore, if we have this type of solution, it will increase the productivity of the developer.

Currently, there are many researchers around NLP to generate code comments, but most of them do not have any architectural solution for proper usage. This research tries to present an architectural solution for the comment generation software. In this architecture, the expected answer is based on the following criteria.

1. The application should handle as many requests as possible.
2. When the request comes, the response should be instant because if we keep waiting until the final output comes, there will be a bottleneck, and some of the requested users may be missed. To avoid it, we used asynchronous process management.
3. To store the data, we used a no SQL database because we do not need to maintain a structure, we can save the final data with some ID, and these are very powerful for that kind of solution.
4. Multiple server instances will increase productivity and decrease the overall processing time.
5. To reduce the Database reads, we suggested using a Cache. Then most of the reads will be the cache reads.
6. Microservice architecture for the services and will then be easy to maintain and easy to develop.

3.1 HIGH LEVEL ARCHITECTURE BUILD FOR CODE COMMENTING TOOLS

When developing the Implementation architecture there needs to be a reference or high-level architecture created. It will demonstrate the architectural principles and qualities that used to develop the conceptual architecture. These principles will define the fundamental assumptions and rules to maintain good architecture. Without these guidelines there will be no compass to guide the involvement of the architecture and it will be a poorly designed architecture, with these softwares it will be very hard to think of a future and needs major changers for the decisions that have already been made.

In this architecture we have separated each service based on different concerns, such as file upload service will do the file uploading into a storage. Then we have given a separate service to help the comment generator to pre-process the things before sending the request to the comment generator and there is another service that is going to help to check whether comments are generated or not. Last component is the comment generator, and it only has the comment generation functionality. I have divided the services based on the tasks without mixed up tasks. I have gone through the selected services only having the pre planned functional requirements because best architectures will have most needed functionalities not everything then this will help us to keep the services less complicated and clean.

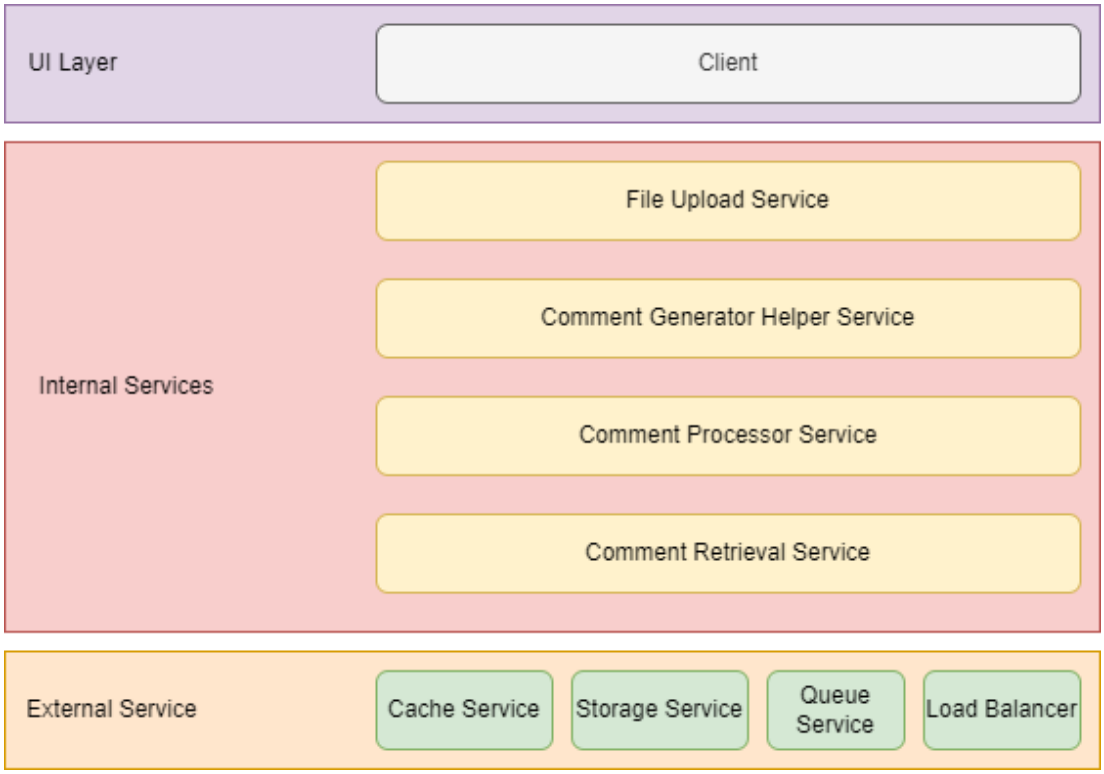


FIGURE 15: HIGH LEVEL ARCHITECTURE DIAGRAM

According to the above diagram these are the components & the services are selected for the comment generation architecture. Client is just an interface to use the comment generator service in a practical implementation that will be an IDE plugin that initiates the comment generation. And then as internal systems there will be 4 services those will be the services that going to process the comments and final layer has the External services those are mainly different type of storage methods that can be used for different reasons

Queue Service: To give async behavior & store requests to process in queue order

Cache Service: Temporary save some data and store data that going to read constantly

Storage Service: That has the capability to store data

Load Balancer: Distribute & pass the request to correct service to process

In the following diagram it will depict the entire architecture and workflow of the research. Mainly it will contain all the components & what needed to obtain through the implementations of this conceptual architecture

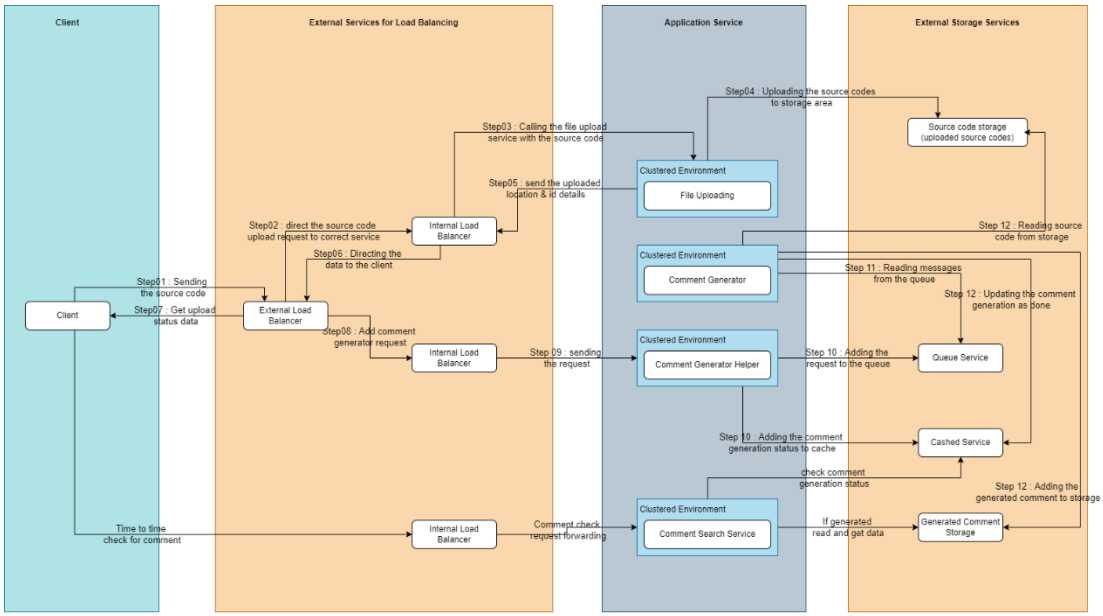


FIGURE 16: DETAILED CONCEPTUAL ARCHITECTURE DIAGRAM

In this architecture, each Application-level service needs scalability to achieve more comment processing once there are more requests to generate comments then it is required to either increase hardware or number of instances then it will gradually increase the number of requests they are going to process. By using separate tasks to separate threads on the same CPU will increase the processing and by adding a few other instances will create a resource pool to generate more threads to process comments. While adding several instances then there should be a load balancer to give the requests to each instance then there will be an external service to do it. Then increase performance I have simplified the services based on tasks there will be a service for individual task not to do complex things so that will reduce the time taken to process each things not only that we are planning to use some queue services to give async behavior to the system once given that requests no need to block they can keep getting requests and once completed they will be able to identify the final output. When using data storages to read & get data it will be very costly to reduce that we are planning to use caching then it will reduce the number of reads to a storage. When planning the service that is going to put the requests to process might fail in different scenarios. In that case I am planning to give a scheduler service to re-process the requests. Efficiency and the performance calculation will be done based on user requests and we are planning to test the implementation architecture (this is a high-level conceptual architecture) there will be several requests per given time limit & based on that performance will be calculated. By having this conceptual architecture, we can easily identify things that can be improved. Also, it can easily be implemented by using different techniques such as Elastic container services, Kubernetes etc.

This architecture is a hybrid architecture due to the following reasons this is a part of a client server. Client is similar to IDE plugin, or a UI application and

server is similar to the back end processing part so over rall architecture is an client server but once we take the backend then it will be a component based architecture since we are separating it into components based on the functionality also it has the message bus architecture as well because it is going to communicate with each component by using a queue service.

3.2 WHAT ARE WE TRYING TO GIVE IN THE SYSTEM?

- Multiple nodes to accept the file requests -> to serve as many requests as possible. Here we are trying to give a thread pool approach to get the requests because a limited number of request threads are available in a system. From this, we can decrease the request thread wait time as we are leasing those threads immediately.
- Creating two different services serves requests to generate comments and another separate service to check whether a comment is generated. This will reduce the usage of each service, and we have control over the applications.
- Need to use an LB to distribute the requests or need to use an implementation like workload distributions.
- In the comment generator, we are using a queue to put the request which is coming from the helper service (to give the async behaviour to the and system no need to wait till data is processed)
- Multiple nodes to process data (thread pool implementation) parallelly process comments.
- Once the process is done, update a queue with hash and done message. Then we can reduce DB calls, get data once it is done, and minimize connections.
- Separate training module every configured time intervals or when triggered, we can do training out of the datasets and once done, it will upload the to the S3 which we are going to use.

If the architecture is a good one, the users will feel the rapidity and accuracy. For the research, the following are the components and technologies used to generate the code commenting architecture. Each component's functionality and the task will be covered in the below sections. Not only for code commenting, but we can also use this structure for any other similar cases. We need to replace the code commenting engine with the relevant business logic. Following is the component and architectural diagram for the proposed system. That component diagram contains the initial set of identified components during the project execution and demo phase. These will be replaced with the Amazon Web Services components because some can be easily replaced and developed by using those components. We do not need to optimize them.

If we use them, then the final product will be a fully optimized solution. But these components will be the base components for the architecture.

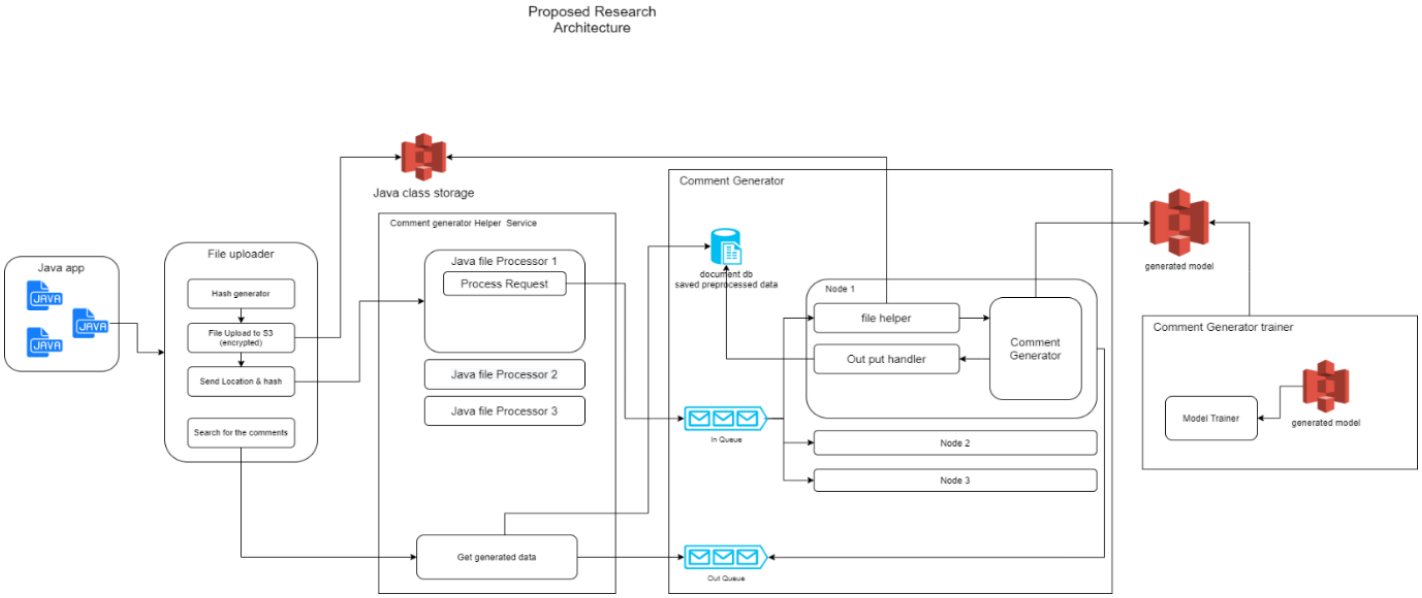


FIGURE 17: PROPOSED ARCHITECTURE DIAGRAM

There are four main sections.

- File Uploader (Client-end application)
- Comment Generator Helper Service (API)
- Comment Generator
- Comment Generator Trainer

Before going into the details of each component, here, we are trying to use an existing comment generator with minimal changes. Because here, we are not trying to re-invent the wheel. We are just providing an architecture solution that can be used for this type of project.

3.2.1 FILE UPLOADER (CLIENT-END APPLICATION)

This is the first component of our application, and this needs to go through the code upload content and constantly check whether the content is created. Also, it needs to add the relevant comment to the placeholder that is created. To develop this component, we plan to use the following technologies Java, AWS SDK, Spring boot, AWS Resources, Python, Shell Commands. File Uploader has four subsystems following are those.

1. File Processor
2. File Upload to S3
3. Comment Generator Helper Service Client (call generator helper service)
4. Search & get Comments that are generated.

File Processor:

This needs to read the java file by separating the methods to create a unique hash code for the client and the method. Once all are done, create a placeholder in the code, we are keeping that because once comments are generated, we can easily add the comments to the given location (place holder). Furthermore, this component must keep track of the file that needs to update the code comments and send the hash to the search comment component to check any similar generated values. If not, send the info to the file S3 uploader to further processing. This component is a significant part of the application because it will select the data to process. In the future, we are planning to shift the tokenization part into this, and then we can answer the security-related issues as well.

File Upload S3:

This uploads the files to the S3 with the metadata (hash value and filenames, etc.) and sends the location back to the requester. We are using this component to store the code segments before processing. We are not considering the security in the initial implementation, and here we save the code. But when we are going to generate the comments to a domain that is not allowing us to keep the code segments, we must make some slight changes to the architecture. Here we are planning to use AWS SDK to upload the data to the S3 storage.

Comment Generator Helper Service Client

Calling the helper service with the location and the hash value generated by the system is an interface for the helper service. We can quickly reach the service and get the job done.

Search and get Comments that are generated.

Call the backend and check whether the comment is generated or not. This will call in every configured minute, and it will check the status of the comment. To reduce the load to the backend DB, we use a queueing technology to track the completed tasks. Then if and only if the changes are done according to the queue, we will serve this request ahead.

Following is the component diagram that focuses on this major component, and it contains how the request will work after deploying the system.

We have created an activity diagram to get more ideas and clarification about

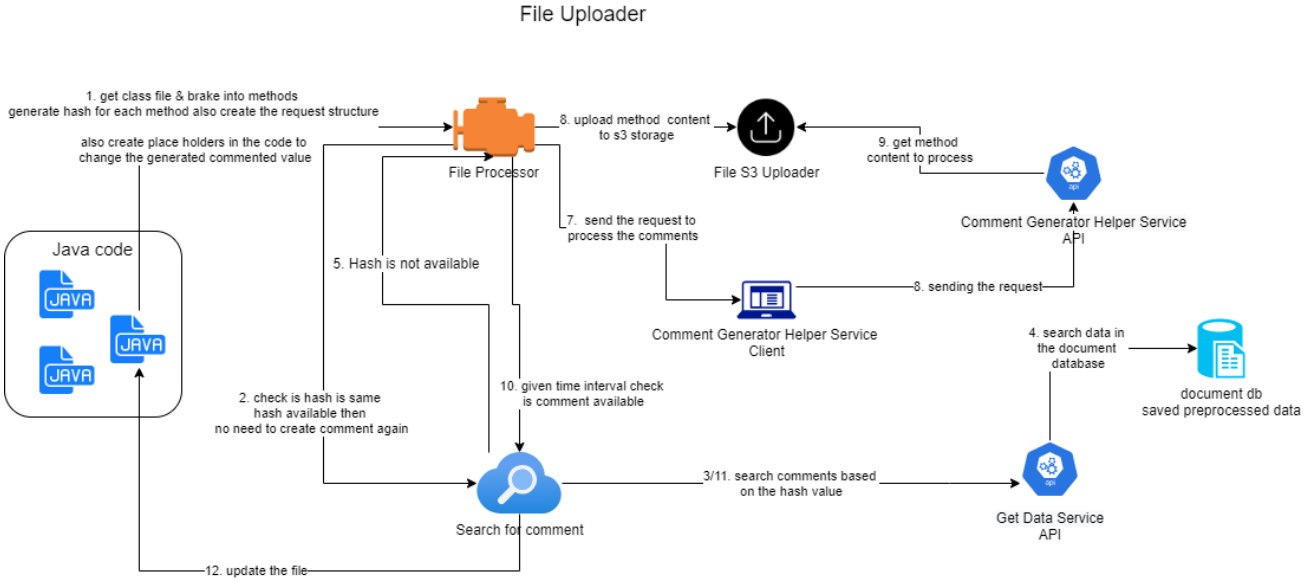


FIGURE 18: PROPOSED FILE UPLOADER COMPONENT DIAGRAM

the client application, but this only contains the significant functionalities. Other less critical are not denoted here.

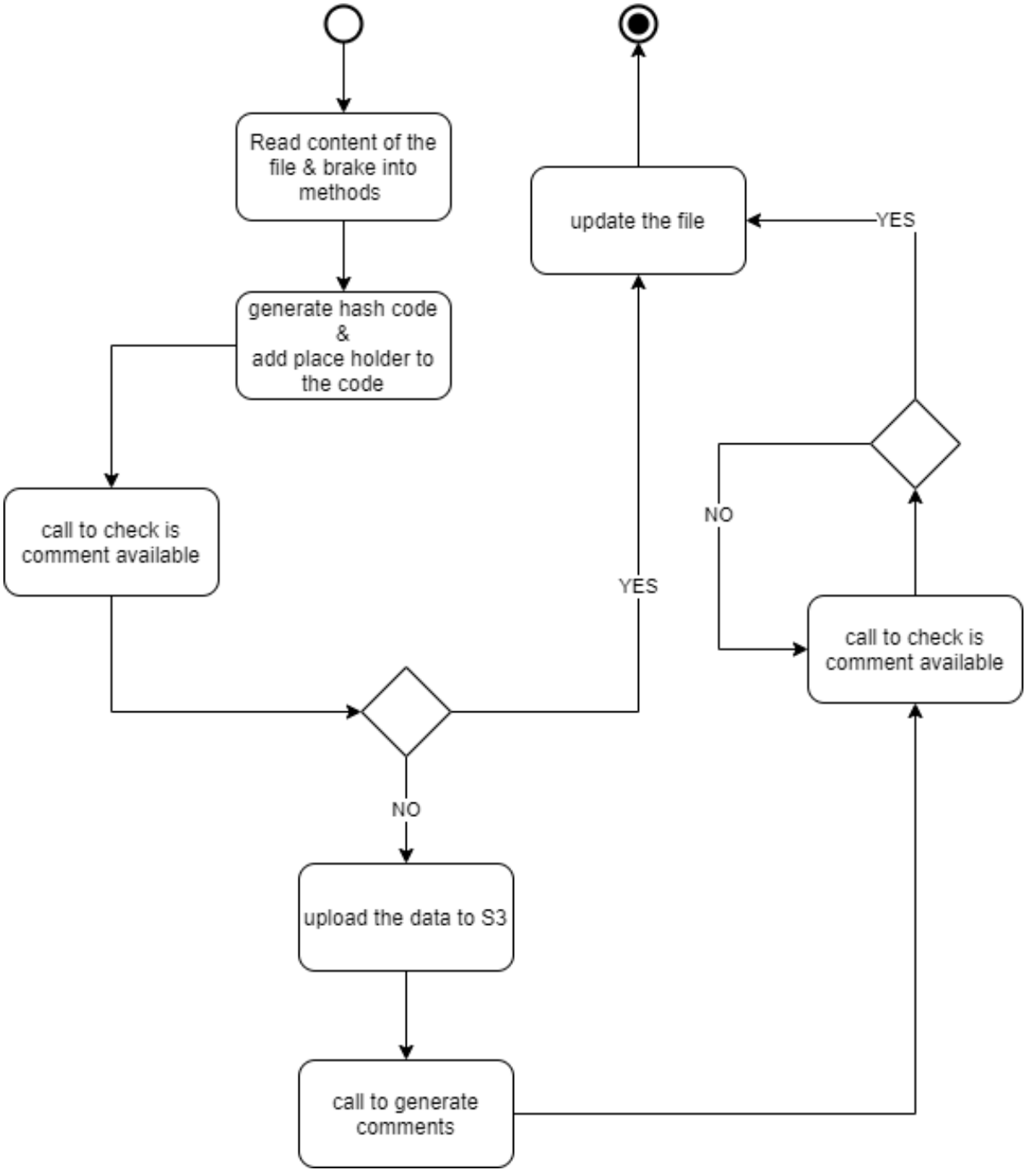


FIGURE 19: ACTIVITY DIAGRAM FOR FILE UPLOADER

3.2.2 COMMENT GENERATOR HELPER SERVICE API

This is the back end that accepts the comment generator requests. In this research, we are storing the code in the S3. But in later developments, we plan to get rid of that because some projects do not want to share the code. Because some of the business logic will be exposed, but for now, that will be an enhancement. In this, we have a request thread pool, and it will be limited. Therefore, we are planning to dump the request as soon as we get it. We are using async behaviour to the system, that means after getting the request. This will be given to a queue. According to the design, this is just a service provider to the front end to check comments are generated or request to be developed. Here we are planning to use spring boot without security. As I mentioned before, we are not considering that, but we just wanted to add another component into our architecture if needed.

3.2.3 COMMENT GENERATOR

This is the most prominent part of the architecture because this will generate the comments for the given requests. Also, this will need more instances other than others. Hence, we plan to use several instances of this application to process the requests parallelly, and it will have a queue. It will be subscribed to that queue from that queue, and once the process is done, it will take another object out of the queue and process it. After finalizing the process, it will flag the other queue because we wanted to reduce the number of requests going to the database in the following activity diagram for the comment generator. This will explain the tasks of the comment generator application.

We are using an external comment generator in this component, so we plan to find a better-performing existing comment generator. We have selected the code2seq, which we have gone through in the lit review. That component uses a python code to generate the code comments, and here we are using that with a java wrapper to subscribe to the queue mentioned above.

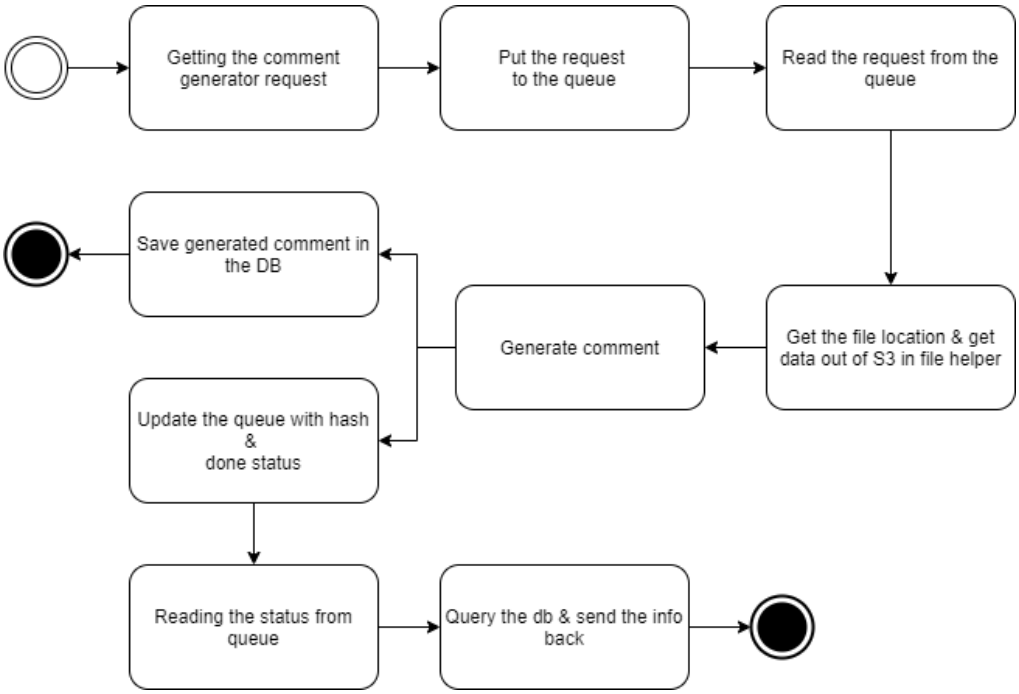


FIGURE 20: ACTIVITY DIAGRAM FOR COMMENT GENERATOR

3.3 IMPROVED ARCHITECTURE TO USE DOCKER ENVIRONMENT

While going through the EC2 environment, there are a few fundamental issues where EC2 is a virtual type of machine managed by AWS side and it has an Operating system in it and virtual type RAM and hard disks of distinct types. But Docker is a container-based application & that container is “a lightweight, stand-alone, executable package of a piece of software that includes everything needed to run it.”

These docker containers are platform-independent like if you have a Windows PC, it can run a Linux container on it without an issue. Also, docker is made for distributed applications, which is one of the primary reasons to select this technology for this research. Also, EC2 still need some maintenance effort & it does support distributed ness. At the same time, high availability is tough to support since container-based systems primarily support auto-scaling without any increased cost. Significantly less security maintenance is needed for the changers mainly it has comprehensive backward compatibility than using EC2 machines. Docker is based on Images that means those are ready-made. You can get & spin up a container in a second, but when going into an EC2, you can still spin up with an image, but it will take time to initialize the environment. In this research, we try to give an app that will not use OS functionalities, so that is also a good reason to migrate the app into a docker environment. Docker containers need more fever resources than an EC2 instance. While using a docker, it is easy to convert the project into a docker because more base

images can be used, those are fully optimized, and it has all the security updates. These are the main reasons to use docker as the app running environment.[44]–[46]

3.4 DOCKER CONVERSION

After going through the benefits of converting the application into a docker environment, I decided to follow the architecture with docker. For the first step, I decided to use the lift & shift method to convert the app into a Dockerized environment using the same EC2 instances. I'm going to deploy docker environments & then add the app on top of it because this is a step-by-step process.

First, I have decided to remove some of the unwanted services and give a few services with a better scope, so while migrating the system, I have decided to reduce the service count to 5 services following are those services. Also, I have decided to use a naming convention for the service following is that naming convention.

1. r: Research / cg: Comment Generator
2. r-cg-search-service: Comment search & retrieve service
3. r-cg-service: Comment Helper & main service that manage the application
4. r-cg-file-service: File upload service
5. r-cg-engine: Comment generator service
6. UI app: UI client to support comment generation

Following is the docker architecture that I followed only for the docker migration. Once it is done step by step, I plan to migrate this to the docker swarm environment because it can have high availability with easy scalability than using EC2's. Also, I plan to remove a couple of EC2's since we need only two EC2 instances to up this clustered environment.

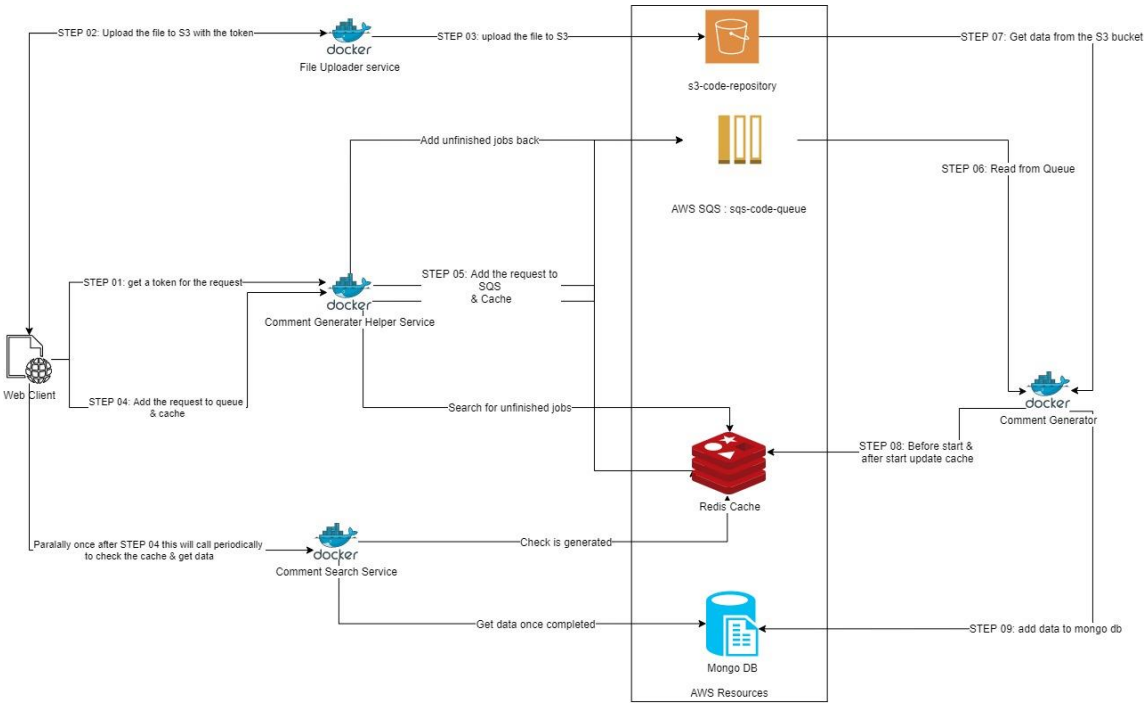


FIGURE 21: DOCKER ARCHITECTURE

Following are the scopes of each updated service.

Search Service:

Check whether the NLP engine already processes the given token or the id. There is a Redis cache we are using to store that information. By using that, it can decide whether the request is processed or not processed.

Another functionality is getting the processed data from the Mongo DB, so once a request is processed by NLP, it will store those data in a Mongo DB collection, so in this REST service, it will supply those data.

Another primary function of this app is rescheduling the failed requests. If the requests are not being processed in 60000 ms, then those requests will be rescheduled in the SQS & Redis side to process by the NLP engine. This is a CRON job & every that time. It will check the unprocessed data and do the rescheduling.

Comment Helper Service:

The main functionality is adding the comment request to the queue once the upload service's response is from the front-end side. Once it gets the request, it will add the request token to the AWS SQS queue, then again add a process request data to Redis cache than from that cache. It will track the changers information.

When the user wants to generate a token, the service will do that because once generated the token. It needs to check whether that token is already issued so that checking & sending the unused token is also the functionality.

File Service:

This will upload the file to the S3 bucket for further processing & it will let the client know about the upload status. Based on that, it will schedule the process after calling the helper service.

Comment Generator:

Generating the comments based on the requests & this service will keep watching the SQS queue. Then it will get the request from its & it will search that file in S3. Finally, it will read and generate the comment & update the Redis about the readiness of the requests & upload the generated data to the mongo database.

UI Client:

On the client-side, it will handle all the calls to the back end & it manages all the generated comments. At this stage, there are no clients to update the code itself that will be a future development that need to do and there are a couple of other future developments like account & security management & etc.

This migration was an easy step for the java Rest services because it was built to deploy jar files & directly run on the java environment. In this case, I just used the openjdk:11.0.11-jre alpine as the base docker image and then others are simple like I just wanted to import the jar file & need to give the start command like EX. "java","-jar","/r-cg-search-service.jar" then it started all the spring boot related apps.

But it was a challenging task. At the same time, it is converting the comment generator to a docker service in the earlier section into a scheduled job. Still, while learning python & converting it to a docker image, that was challenging. For that, I have used python:3.6 because it needed the TensorFlow-1.12.3 version. It only supports python 3.6. As mentioned earlier, Docker has good backward compatibility. That’s why we were able to use 3.6 with the installed version of TensorFlow & after creating this image, no need to worry. We can use it as it is.

After creating the working docker environment like the above architecture, I planned to store the working images because those are the last code changes needed since all the test cases are about to pass. For the storage of the images, I have used docker hub image repository for that there I have created four image repositories & stored those images by following the versioning strategy like Ex. v1.1 in the first digit it denotes that this is the first significant release & second will stand for what minor release that going to build.

3.5 DOCKER SWARM MIGRATION

Once all the services are converted into docker environment, for now, all the service side implementation is done only pending item is availability & fail safeness of the application so I have searched about the changes that can be done to have this is one change is moving the changers into beanstalk environment & keep using EC2 or using the Kubernetes environment but using EC2 is again giving the issues that mentioned above, so I have decided to use Kubernetes, but before converting the changers to that, I have done an intermediate change docker swarm by using this we can have multiple instances of each service & it will maintain a load balancer to distribute the requests among the application. This is an intermediate stage to convert the entire app to Kubernetes. Before converting it, we need to check the issues while using high availability in the application. Wide Ex. Properties management etc. Architecture wise, not much different from the above architecture. Rather than using multiple EC2's, this was under one EC2 instance with bit high resources & this is not with two availability zones since this is not going to use for the architecture & this is just an intermediate state of converting the application.

3.6 KUBERNETES MIGRATION

As mentioned in the literature survey chapter, Kubernetes has a lot of advantages. It has high portability and flexibility among other technologies like Linux & windows without an issue like other techs. This supports multi-cloud capability, such as it can scale from one cloud to another cloud without a problem. For example, it can work with different cloud providers, AWS and Google cloud, with few configurations' changers. Also, this technology reduces the work from the DevOps side because only a few configs need to do to deploy the changes in the Kube cloud, and a few things need to be done from the developer side to support this. Opensource ness is one of the main reasons to use this technology. Others give the availability with a cost, but this gives freely we only need to pay for the cloud instance, not for the Kubernetes features. And most of the major companies use this as their service supplying platform. One of the industries is security solutions. Now this has been used for more than five years in the industry also it is one of the market-leading technologies nowadays & most of it is stable & no significant issues available not only that this is supporting by Google so they always try to update the features on this environment even the community who is using this also going to give solutions to this opensource project. Due to those reasons, I have selected this technology.[47], [48]

3.7 COMPONENT CHANGERS

In the earlier sections, we have migrated our changers to Docker-based images. We converted our application to docker swarm, the other significant alternative to Kubernetes but not better than this. During the conversion process, we have found the properties that need to change while trying to start & when exposing the endpoints one into another, what should be the changes required to support. We have used a local pc environment for the migration since it was very costly in AWS to have that mentioned high available cluster for the research purpose. We have used the following PC configurations

- RAM: 16GB RAM
- Processor: Intel(R) Core (TM) i5-10210U CPU @ 1.60GHz 2.11 GHz
- System Type: 64-bit operating system, x64-based processor
- OS: Windows 10 Pro
- HDD: 250GB

In this PC, we have installed the docker environment with the Kubernetes environment. This is the latest open-source version & it has all the capabilities that we are searching for. As prementioned in the above architecture diagrams, we are planning to use some AWS resources such as.

- S3 storage

- AWS SQS (ActiveMQ Queue)
- Redis Cache but AWS managed
- Mongo Database or document DB

But when we checked the prices for the implementation that was a bit higher for a prototype application, I searched for a solution that could replicate the AWS mock environment. Once you can mock it, we can easily convert the system into AWS. While searching it, I found a local stack project that an opensource group generated to mock all the AWS services we will use, so I have used that solution to create the S3, SQS, and then Redis & mongo are separate docker containers. For building the S3 & SQS, I have used terraform script because it creates the instances instantly with a few commands in place. And this script can be used to generate the same resources in the AWS environment.

While using the Kubernetes, first, we must create the environment-related configurations before starting any of the service pods. After beginning, the Kubernetes cluster's first job is to create a service account because we will use Kubernetes UI for easy management. For that, we must have a service account & for that, that user needs to have cluster role admin to manage the whole environment through the UI given. Then finally, we are using Deployment configurations to create the containers for the local stack, mongo database, Redis Cache, & their services. These will act as a load balancer and the endpoint for the clusters. This will be the primary environment configuration that is going to use. Once this is completed from the UI, we can check the service health & if the cluster is up and running without any issue. Then Initiating the terraform script will start the S3 and SQS on the system. At the last step, we are creating the app Deployments & the endpoints for the services. [49]

Following is the implementing architecture. It guarantees comment generation and availability. The architecture evaluation in the CBAM will discuss how much it will take to implement the system in the real environment. This technology mainly supports scalability.

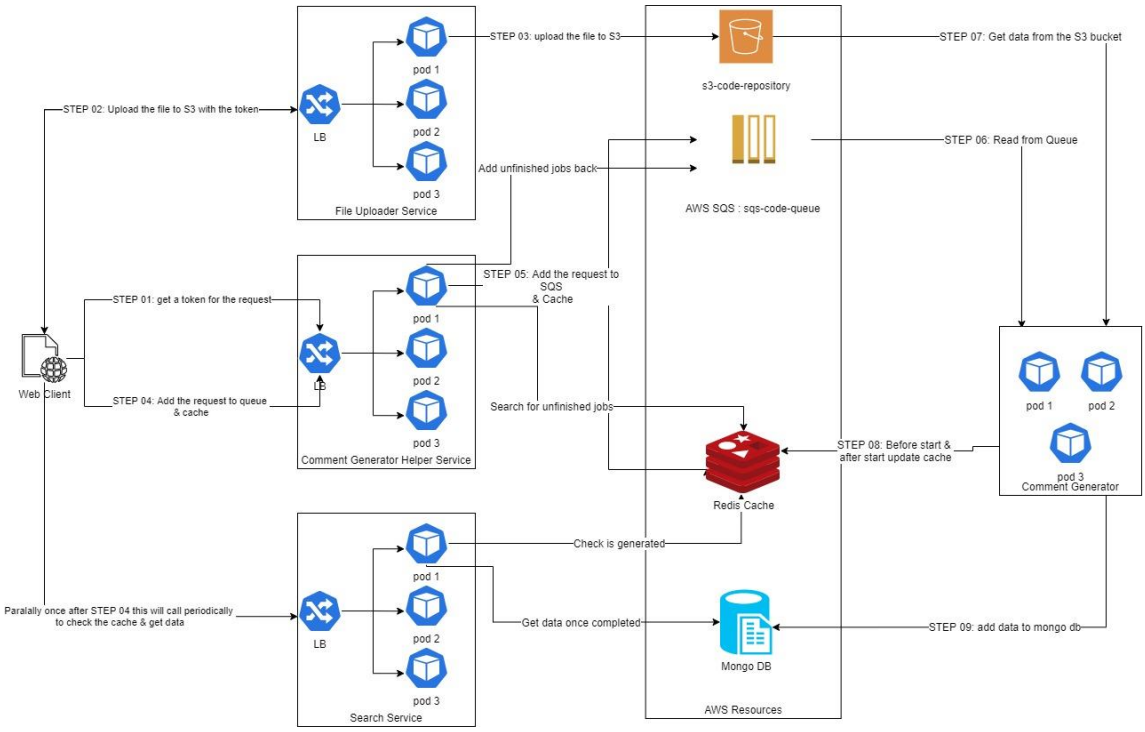


FIGURE 22: KUBERNETES COMPONENT DIAGRAM

4 EVALUATE ARCHITECTURE

Different types of aspects need to be checked when creating an architecture because architecture is not tangible. Then we have to use specific methods to evaluate this. But before that, we have to model the architecture into different aspects like security, maintainability, etc. Based on those, we can see what we are going to give. Also, for example, if the project needs more security, we can add more security on top of the project. But we have to model first to know about the architecture that we are going to propose.

4.1 SELECTED QUALITY ATTRIBUTES IN THE DESIGN

4.1.1 MAINTAINABILITY

When we select this as a quality attribute of architecture, we must ensure it is achievable in the designed system. In the current architecture following will be done to achieve that.

How to achieve maintainability in this architecture?

- Create services into the microservice structure.
- Keep good coding standers among components.
- Get review comments from expertise on the code changes.
- Create code with high understandability and readability.
- Use version control and add correct comments in each place.
- Feature-wise components break down.
- Monitoring tool for the system.

Once we have maintainability with the up-given features, the system can be updated according to the new changes.

- Fixing errors are easy in individual components.
- Can do easy deployments individually if needed without affecting another system.
- They can easily replace the component if they want to update it.
- If the application is getting increased, can put teams according to the components.
- Components are easy to understand.
- If an application needs to increase, nodes need to deploy the application.
- Automatic resolve some issues in the system by restarting or sending alerts to the relevant persons.

4.1.2 PERFORMANCE

Performance assessment of architecture means supportiveness of the architecture for the scalability and responsiveness of the system. This is one of the critical aspects of architecture because once you do not have performance, you may not get the maximum output of the system. Also, sometimes fixing performance issues are very costly. If we are thinking about this aspect early as possible, we can easily overcome those issues.

How to achieve performance here?

- Increase this by using multiple instances of needed microservices or increasing hardware (horizontal or vertical scaling).
- Create the system as much as async possible.
- Use a proper Load Balancer to balance the workload.
- It improved network connectivity among components.
- High transaction power document Database.
- Use frameworks like spring boot etc.

After using the above-given features in the application, the following can be achieved.

- More instances can process multiple requests at a given time.
- For some scenarios, applications need more hardware this need to be checked & confirmed whether it is increasing the performance.
- LB's can identify what the free instances are, and they will put the requests to them.
- If the system is async, then no wait will happen. All things are done non blocked.
- If we have more transaction-powered DB, transactions will be fast.
- Frameworks were more optimized than using pure changers.

4.1.3 INTEROPERABILITY

In this section, we see the system or part of the system responsible for its operation and the transmission of data and its exchange with other external systems. Here we have ActiveMQ as an external service for that we must have an interaction method for that then it can handle that external service. If we have that, we can abstract it from the code (can easily change services to others)

4.1.4 USABILITY

Users can see directly how well this attribute of the system is worked out. One of the critical issues is too many interactions or too many actions need to accomplish. To be functional, many aspects need to achieve before creating a production-ready architecture. Usability mainly gives that how easy users can use the system. Therefore, designing the system that should follow all the usability approaches like accessibility, how easy it can be learned after going to production, how much time it is spending to send a response more time will reduce the system's usability. Reliability: If the system is more reliable, it is good because it is always up and running, even what happened. Adaptability is another aspect this is nothing, but it should change across the usages because the user experience will change every time. In the usability, these need to follow to get a better usable software platform.

1. Give formal guidelines to follow before using the system.
2. Give a proper client to interact with the system (user-friendly).
3. Recovering from failures
    a. If failed after saving the request to the database, then there will be a scheduler to pick requests and send them to the generator again.
    b. If the request did not give in a configured time, re-initiating it from the client-side.
4. Reusability of the architecture
    a. Modularized comment generator (can easily replace with other)
    b. Modularized helper services

4.1.5 AVAILABILITY

Availability is nothing, but if the user wants to carry out some tasks using the software, it should always be ready to support the study. This availability is on top of the reliability, but it has some additional features like error recovery. Somehow system should be functional in architecture. If we select this as a quality attribute, we can put some enhancements to give the availability.

How to achieve availability?

1. ActiveMQ will be multiple queues (master-slave)
2. I am using horizontal scaling to have multiple instances of service.
3. Add an alerting service.

After using it

1. Then ActiveMQ will be highly available once the master is down slave will pick the place and run the system without interruption.

2. When having multiple services of the same, if one goes down, service will run. Also, if the one got down, it can be re-initiated again.
3. Alert service will give notifications about the system.

4.1.6 RELIABILITY

Reliability is nothing but how correctly the system is working or how trustworthy it is. If we were given a request to the system, how much probability will it provide the correct output, the period of the right output time, and how easily the system can come out of an error. This is a customer-oriented view and is based on the following. We are planning to enhance the system's reliability.

How to achieve reliability?

1. I am adding a scheduler.
2. I am adding Timeout on the client-side.
3. Check while getting data from the DB.

After using these

1. The scheduler will pick up the requests given some issues and try to re-initiate the process.
2. In the client-side time, out will constantly check the ActiveMQ, and it exceeds that value, it will re-initiate the request.
3. If the DB is not there, it will initiate the request again because it determines that some things happen while saving the data.

4.1.7 TESTABILITY

This is nothing but testing the software artefacts. When we are using the architecture, there should be a method to test the system individually or with the whole system at the given time. If this testability is a bit hard, we will not be able to correct the reliability, etc.

How to achieve testability?

1. Individual testability of each component to check whether it is working well or not.
2. Should possible the full system test.

4.1.8 MODIFIABILITY

Modifiability is how easily can be added a new change or fix something in the system. To achieve this feature, we use different concepts like micro servicing the

system, using managed service for some components, and documenting the steps for the special features. From the below method, we are planning to achieve this.

1. I am planning to use Design patterns.
2. I was planning to break the system according to the service.
3. Use proper coding strategies.

Then the application

1. If the system needs a modification, it should be quickly done, so while it needs a change

4.1.9 SCALABILITY

Software scalability is to increase the capacity and functionality based on the user requests to the system. When planning to do an architecture design, we must design it anyway because any system can increase user count. The system should not give less performance. If we plan to overcome these, then it will be a scalable architecture.

1. The system should be able to scale based on the request load using a Kubernetes cluster.
2. The system should scale vertically by adding more hardware resources, but this needs to be minimized.

4.1.10 REUSABILITY

Once we achieve all the functionalities, we will be able to get a reusable architecture because we have a reliable async system & this can be converted into any of the software systems like generating something from an external system.

4.2 PRESENTATIONS

Using presentations, we can deliver the architecture that we are trying to build in this research. Reading documents some time a bit time-consuming and hard to understand, but when using a presentation medium, it would be straightforward because we are trying to use some symbolic interpretations for the presentations. For the explanations, architecture is not a tangible one. That is why we use many methods to interpret it; one is the component diagram. From that, we can give an idea to the stakeholders about the components that we are going to use and the tech stacks used for system building. This presentation will have a basic idea of the system, and they will have some enhancements that need to be made. This will be used mainly at the beginning of architecture creation. We can show the initial build architecture to a set of experts and then get their reviews and modify it. This may do a few sessions and gather information. Only with this method can we not create excellent workable and quality architecture. Hence, there are some other evaluation methods also used to evaluate the architecture. In the initial project design, we have done the initial architecture presentation. At that level, we identified the issues, and that was the initial phase. After getting the feedback, we initiated the second phase of the design.

In the first iteration of the presentation, I have identified the service components & how they are going to connect that presentation. Our primary focus was on how the app will solve an issue that currently has in software development. In that presentation, we have explained how we can increase the developers' productivity by adding automated comment generation. By this comment generator, we were able to prove that it will reduce the development & project maintenance time tremendously. In the first presentation, we have given a basic idea of the components that we will use. We got feedback from the industry experts, like how we need to maintain the availability and make the system distributed to serve multiple requests. In my first presentation of the architecture just focused on the NLP side & not the architectural components. Following was the initial design diagram.

FIGURE 23: FUTURE IMPLEMENTATIONARCHITECTURE DIAGRAM

As I mentioned, the focus is on comment generators and security. Still, after going through the first iteration, I have the update from the industry experts to move the focus to the overall architecture and use an existing generator instead of rebuilding the weel not only that instead of multiple clients need to build a platform that can be used by any of the clients within the given time frames. Then I modularized the components based on those feedbacks.

Then we moved to the second iteration of architecture preparation in the second presentation. I have made the architecture Figure 2 there. I have promised to give scalability, high availability, serve more requests parallelly & etc. To serve those requirements, I have decided to use queues because we can separate the threads & asynchronously run the requests & get the response after all is done. We plan to have as many more minor services as possible according to microservice architecture. Then it will be pretty easy to maintain. In the second presentation, we have focused on those scenarios mainly and, we have given the plan of the development & how it is going to manage & how much work that needs to be done initially. What type of resources are

required from the AWS side & what will be the initial cost for the project, and is there any alternative solutions that can be used for the app-building?

4.3 FORMAL REVIEWS AND STRUCTURED WALKTHROUGHS

In presentations, we are using the same method formal reviews and walkthroughs. But we cannot do many presentations, and it wants to be feasible because stakeholders or expertise will not have much time for presentations. But this review can be done in other ways, like sending emails with some explained content and diagrams because people can easily understand diagrams. Therefore, we have planned a considerable number of reviews to get better architecture for this scenario. In this phase, we initiated the second phase onwards. Here we corrected the first phase things. We are trying to identify the quality attributes that will be used, what the structure will give, how we will provide that quality for the architecture, and the best component breakdown the system. These evaluation methods are sufficient, which can be discussed during these formal reviews that we planned during the architecture review process. Not only the architecture, but we also had sent a couple of emails and documentation for our reviewers to review and get their understanding and the potential updates that can be done. According to them, we planned to migrate the architecture to the Kubernetes environment. Still, we had to migrate the application to docker & then to docker swarm since we proposed the docker swarm. Reviewers' thought was, can we easily have the auto scaling instead of any human involvement. Hence, they told me to check if any alternative technologies can be used, like Kubernetes. Once I had gone through that initially & once given the architecture diagram, they were happy about the architecture. But here it is just a prototype application & it is not having any of the security & that is a future work that can be done since here focus is availability, Scalability & application component structure (how the system is going to act on a multiple request environment)

4.4 PROTOTYPES AND PROOF-OF-CONCEPT SYSTEMS

This is another evaluation technique that we are trying to use. Here is nothing but trying to prove the concept that we have mentioned in the documentation. We are showing component diagrams and some activity diagrams etc. Here we will implement the system and show it is working. This will be a time-consuming and costly method. As this is research, we can use these methods to evaluate the system. This can be derived as a skeleton system because we are not partially implementing it. We are trying to implement a working solution. From this prototype, we can check whether our program is going to work as expected also.

In the initial development phase, I plan to create the services according to Figure 12 & then, in the other iterations planning to migrate from docker to Kubernetes. In this iteration, we just created each application in the local PC & planned to run those on EC2 instances in AWS, and we have used java as the development environment and used java 11. & For the comment generation engine, we have used the 3.6 version since the selected comment generation engine needs TensorFlow & it was the older version of the TensorFlow, and it required that older python version to work. This first iteration also needed the mongo, SQS, S3 resources from the AWS & we have used the local stack alternative since AWS has a bit high priced for a prototype application. Still, it acts as that environment, so we have chosen terraform scripts to generate the AWS resources. Once converted that process to terraform, it is pretty easy to maintain the application resources. Once all the apps were developed, we ran all the services on the local PC (same as EC2) & integrated the complete application. After that, we did the second iteration of migrating the changers to a docker environment.

In the second iteration, we planned to migrate the app to the docker environment & there were very few changes that needed to be done for the migration. For the java services, it was Pretty much the same. We just wanted to get the jar to start the container, but the python comment generator engine needed to install some dependencies before creating the image. Once all is done, I have begun to push the images to the Docker hub to store the images then we can re-use the images if needed. Before migrating the application to Kubernetes in the third iteration, I needed to check the changes required by the container-based applications & it was converted to a docker swarm & which gives multiple container functionality to get the availability also scalability. There are drawbacks to using the docker swarm because it cannot maintain auto-scaling and cannot communicate with in the network (can communicate but need to do lot of configuration level changers) so after the second review, they told to use it Kubernetes.

In the third phase, we have migrated the app to the Kubernetes environment. After Step 01 and Step 02 is done, there are no container level changers. We only needed to install Kubernetes in the local pc and deploy the changers & for that, we

have created a separate service for each & it has a load balancer in it. Once you get a request, the endpoint will pick it and then be directed to an available pod running the application. In this environment, it is pretty easy to spin up new containers & there is no extra work needed to handle the requests. Also, it has the capability of auto-scaling if the threshold limit gets exceeded. In the planning sessions, we have planned to use two high capacity EC2 instances of Kubernetes for prototyping.

4.5 SCENARIO-BASED ASSESSMENT APPROACHES

4.5.1 ATAM (ARCHITECTURE TRADE-OFF ANALYSIS METHOD)

This method is used to evaluate software architecture based on the quality attributes that we will select for the architecture, and we are taking the architectural decisions based on these attributes. This method allows engineering tradeoffs to be made among possible conflicting system quality goals. ATAM can easily see the potential risk areas & in the first stage of the development, this method is quick and can be done early as possible main thing is the less expensive method. During the evaluation, most of the project figures involved, for example, project managers, clients, developers, maintainers, testers, and senior development resources. Following will be the outcome after going through this method.

Risks: Can find the future problems that can occur in the quality attributes.

Sensitivity Points: what are the critical properties of a given component based on the quality attributes. Suppose the properties have got to change how it will affect the quality of the system.

Tradeoffs: we can get an idea about what will happen to others while improving one quality attribute.

For ATAM, there are nine-step processes that need to follow to get a full report.

Step 1 to 3 is the presentation of architecture.

Step 1. Present the ATAM.

This step needs to explain the initial architecture to the following persons to get the review comments customer representatives, the architect or software architecture team, user representatives, maintainers, administrators, managers, testers, integrators, etc. In this phase, we just wanted to give the stakeholders the method to evaluate the architecture. Here our focus is to collect the knowledge pool (person who will use the project and the investors for the project, finally, some professional expertise in the area to give ideas).

As mentioned in the presentation paragraph, we were successfully planning to have multiple presentations to the stakeholders who are going to use them and to the industry experts who are going to comment on the architecture and based on those comments, we can fix the issues & improve the architecture to give the quality attributes that we have selected.

Step 2. Present business drivers

In the second step, the project manager's involvement is high, and he will go through the business goals that will motivate the development effort. That means the business interest will be there in the proposed software and the architecture, and only investors can invest money in the projects. Following will be some of the primary architecture drives while developing architecture. In the explanation, we must give the high-level functional requirements of the system, the quality attribute requirements, & the constraints of the system. On the business side, what is the targeted market & is this is an excellent time to be in the market. What are the benefits along with the cost? Finally, what will be the human resource constraints with the technical aspect of the system?

Functional & Non-Functional Requirements

Functional

1. Create comments based on the code given.
2. Keep updated the cache based on the comment generate status.
3. Search the comment after generated.
4. Storage of the generated comment.
5. Queued requests.
6. Async & Parallel comment evaluation.
7. Historical data maintenance.
8. An external storage medium for the code uploader service.

Non-functional

1. Low database quaring
2. Performance
3. Scalability
4. Availability
5. Reliability
6. Recoverability
7. Maintainability

Selected Quality Attributes

1. MAINTAINABILITY
2. PERFORMANCE
3. INTEROPERABILITY
4. USABILITY
5. AVAILABILITY
6. RELIABILITY
7. TESTABILITY

8. MODIFIABILITY
9. SCALABILITY
10. REUSABILITY

Business constraints

1. This is an excellent time to have this kind of software because currently, projects need to decrease the development time, so adding comments would be an additional effort.
2. We are planning on Java, but we can increase the count of supporting language by minor enhancements.
3. The estimated cost for the resources is a bit low, but when we are getting more requests, we can plan to increase those (planning to use AWS)
4. Our target market is developers & software development companies.
5. Payment schemes will be significantly less for those companies after we go live.

Technical constraints

1. AWS is a resource provider.
2. I am planning to use Java for the backend services.
3. Queue (ActiveMQ) for the backend service.
4. Mongo & MySQL databases for data storage.[50]

Step 3. Present software architecture

After doing the 1st & 2nd steps, we can get the knowledge pool & the business & technical drives. In this step, we must give an initial architecture for the system, and that should clearly describe the areas mentioned above because those are the drives for the project. Using the above people, we can get valuable comments on the architecture we are proposing and then address those modifications to the system according to the comments.

We have done a series of reviews with the industry expertise via emails & a few presentations from those sessions. We were able to identify some significant issues & we were successfully fixed those issues

Then, 4 to 6 is mostly investigation and analysis

Step 4. Identify architectural approaches

In this phase, it is mainly to select a base architecture approach to the proposing architecture. This will be an architect's job role & he wanted to go through the scenarios & propose an architecture to the new architecture.

In this research, we have selected the microservice architecture but not the fully microservice. We are defining services based on the tasks & adding those as a component to the architecture also, and we are planning to use the async behaviour in the system. Async will ensure that system will parallelly process the requests & requester no need to wait till it processes the request. Finally, to reduce the database

reads, we are using a caching solution to check whether the request is being processed or not, then only a single time will it hit the database & get the responses.

According to the current architecture, we are not following the fully microservice because we must have different resources to maintain the individual ness of each service for each service. Still, if we use multiple resources, it will not give the correct behaviour that we are expecting. For example, SQS, S3, mongo databases are not unique resources. They are shared, and all of those are AWS related managed services. This architecture is more or less a hybrid architecture, and it can give all the quality attributes such as availability and scalability.

Step 5. Generate quality attribute utility tree

Using the utility tree, it is straightforward to identify the subtask that needs to be done to archive quality attributes. Following is the proposed utility tree diagram that we have created.

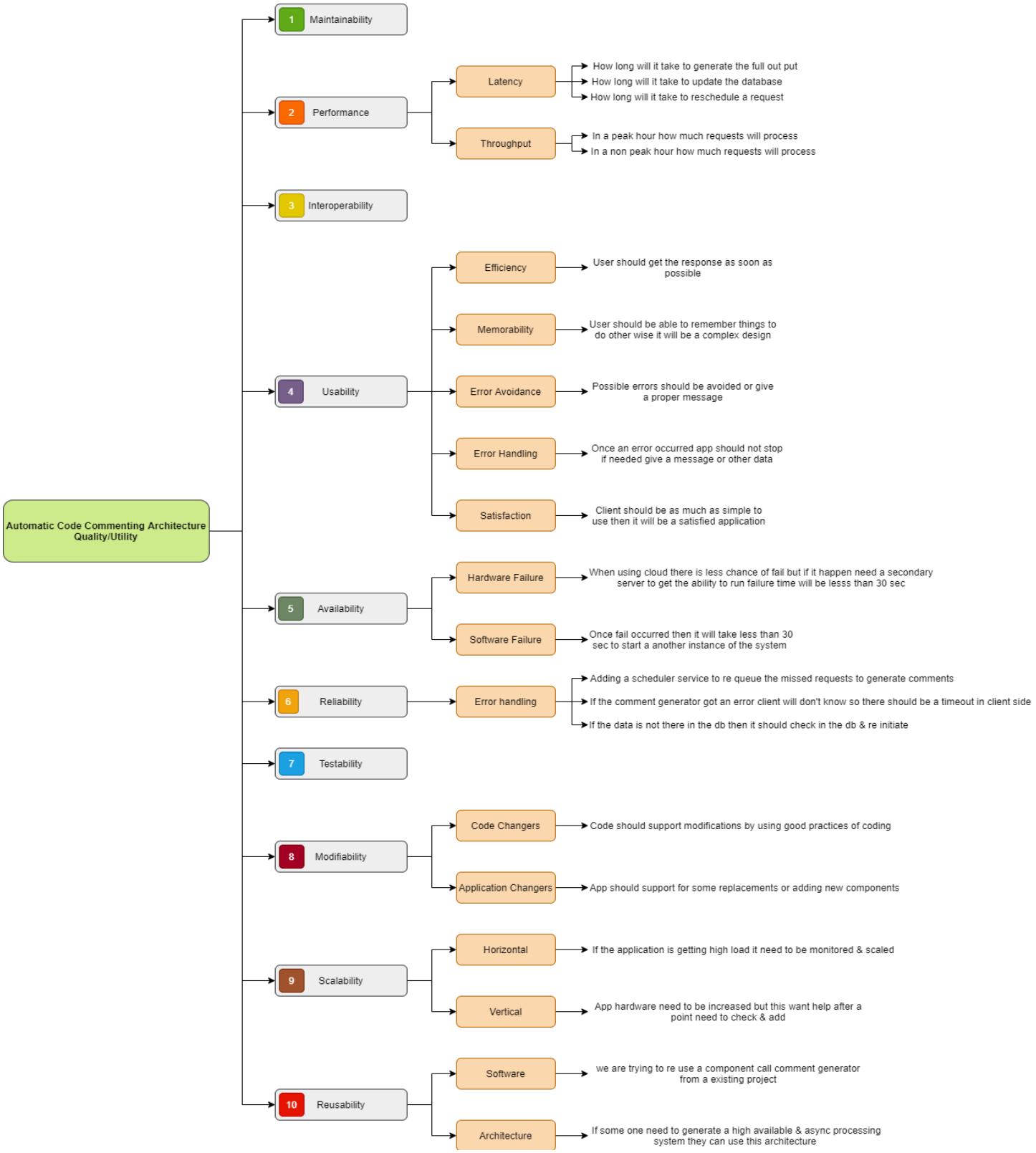


FIGURE 24: ATTRIBUTE UTILITY TREE

Step 6. Analyze architectural approaches

In this step, we are trying to group up the items that we found in the previous step & we are trying to find the priority list of the quality attributes. Then, based on that list, we will put some changers & technologies to the proposed architecture. So, here we are planning to follow this order & primary focus is to give high Usable & Available software architecture. Accordingly, the other will be provided following we will describe what & how we are trying to achieve those.

The following will be the priority list of our approach

1. Usability & availability at the same level
    a. give the responses as soon as created.
        i. once got the request put the process into an async behaviour
        ii. once completed it call & get it
    b. keep things in a cache or database
    c. masking the errors into a proper thing
    d. simple UI to work with
    e. if some service got an error within 30 sec to create another container & start working
    f. in the cloud no need to worry. They will handle those failures (Hardware related)
2. Reliability
    a. adding scheduler to reschedule to process not scheduled things
    b. if the comment generator got an error, then
        iii. the client will have a time out & it will give re-initiate the request to generate
3. Performance
    a. I was given a response quickly by adding the request to the internal thread & releasing the request threads because it takes some time to generate & more requests can be served.
    b. auto-provisioning containers if getting more requests
4. Scalability
    a. Kubernetes technology to control the auto-scaling
5. Interoperability
    a. working with managed database & ActiveMQ of the AWS
6. Maintainability/Testability/Modifiability/Reusability
    a. using git to store code
    b. adding relevant comments then any time can make the changes
    c. adding good practices to the coding
    d. individual testability & the complete system testability
    e. components should be changed if needed

Then, Step 7 to 8 is testing related

Step 7. Brainstorm and prioritize scenarios

In this step, we are mostly trying to show the in-detail picture of the architecture we have created. After the 6th step is done here, we need to have several

discussions to identify whether this priority list is ok to go with or if we need to make any changes to the architecture. During the review sessions, we were able to create increasingly robust systems.

Step 8. Analysis of architectural approaches.

This step tries to go through the architecture only for selected high-ranked scenarios by creating the test cases. In these test cases, we can get to know about the missing items & some of them will cover the low priority items as well. It will also give the tradeoffs and the things that are going to be risky. All those will be documented in this step.

The final step is Reporting the findings

Step 9. Present results

This is the final step of the architecture evaluation. Here we are trying to show the risks & how we will create the architecture, etc., then this will be the final evaluation with the stakeholders [23] [24].

In the ATAM, we can't get an idea about the risks and the technologies that will be the qualities in the system. Still, we cannot evaluate the cost estimation of the resources or the project's feasibility, so we propose using the CBAM method to assess the complete architecture to know about the cost-wise feasibility.[51], [52]

4.5.2 SWOT ANALYSIS FOR ARCHITECTURE EVALUATION

SWOT analysis is mainly based on Strengths, Weaknesses, Opportunities and Threats. Using this framework, we can check whether our planned architecture or the software will be a successful one or not based on these four facts. This analysis will check the internal external facts and how they will affect the product, project, place and person and then the future potential upgrades that can be made for the architecture or the software. In this analysis, we must openly mention things. Otherwise, this will not accurately analyse the project, for Ex. Suppose the project or the software team has personal issues. In that case, they may be not focused on what the changes need to do if that is the case, the project may fall into critical situations, but before that, if we identify the issues & strengths, we can avoid those. In the SWOT analysis, we mainly focus on doing the architecture evaluation using that model.

The strengths section consists of the strengths, what type of unique technologies and techniques are going to use, and what things are done to attract investors because they mainly think with less cost get more benefits. For the weaknesses section, what are the missing parts of the system design? Basically, what are the missing parts? In the opportunity section, I have described the system's benefits & will this investment is suitable for an Investor. The Threats section mainly it is going through the potential threats that can occur to the system. This will be done few iterations to finalize all four items. Then according to those, we have to plan the architecture. Also, we can get to know about the things that need to improve and what we can get out of this.

For this analysis, I have used several colleagues that works with to identify each of the issues & good things & came up with the following SWOT analysis diagram after a few iterations[53], [53], [54]

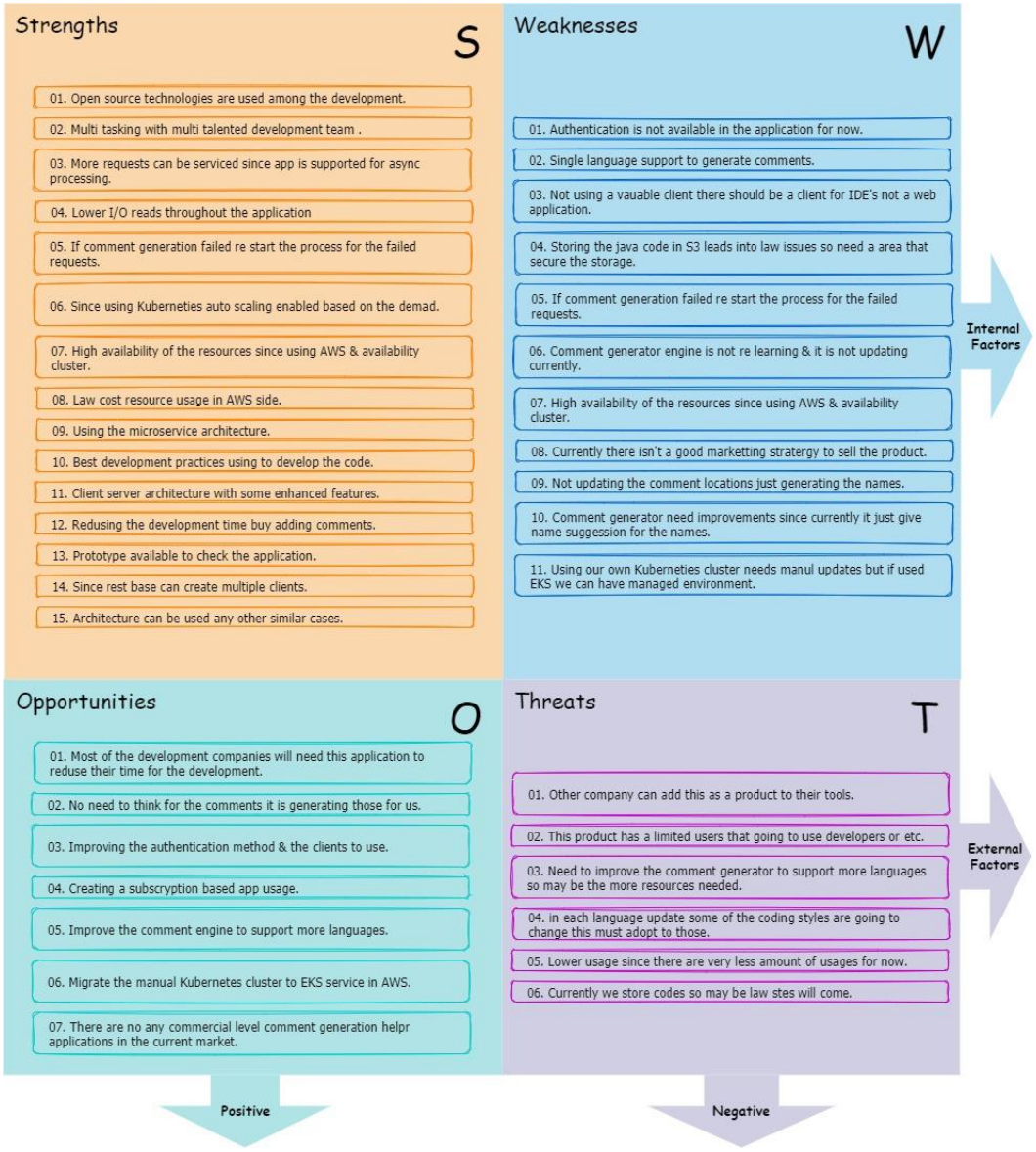


FIGURE 25: SWOT ANALYSIS

4.5.3 COCOMO MODEL ARCHITECTURE EVALUATION

COCOMO model is known as the Constructive Cost Model. This is a regression model mainly based on the Lines of Code (LOC). In this model, mainly for software purposes and when we try to predict the cost, various attributes need to take into the picture, such as size, effort, time and the quality of the software that we are going to make. COCOMOS primary outcome will be the Effort and the Schedule of the development

Effort: Amount of labour (development time) time required to develop the system

Schedule: The amount of time required to finish the task is proportional to the efforts that we will put into the system.

In the COCOMO model, there are three levels to calculate the cost estimation in each level, accuracy is going to differ Boehm introduced this, and he has given those three approaches

1. **Organic Model:** Team size will be small (same as our software development), and the problem domain is clear. Also, this was already solved by the team previously, which means this type of project should be implemented by the team & they should have experience with the technologies that are going to use.
2. **Semi-Detached:** In the semidetached means it already relies on the criteria mentioned above but coming into these, developers have less development experience on this type of project & they have a risk environment around the project. They need better guidance & better creativeness while developing the application.
3. **Embedded:** When coming into this level, that means this has a lot of risks like no one has any experience developing these applications & the team also needs to have a best-experienced team to work on.

Also, In the COCOMO model, they have three models based on the accuracy of the calculation. Following are those models.

1. **Basic COCOMO Model:** This is a quick & rough & basic calculation that gives an idea about the cost that can go to develop the application. Also, it just has a fewer amount of factors while calculating the cost
2. **Intermediate Model:** In the basic model, they think that cost mainly relies on the lines of code. They are not considering other possible factors. In this advanced COCOMO model, think of 15 other attributes while calculating the cost following are those
    - Product attributes
    - Hardware attributes
    - Personnel attributes
    - Project attributes
3. **Detailed Model:** In this cost estimation it will incorporate all the possible scenarios. In this analysis it will divide the application into different modules to estimate & there are 6 phases

a. Planning and requirements
b. System design
c. Detailed design
d. Module code and test
e. Integration and test
f. Cost Constructive model

Detail Model Calculation for the proposing architecture.

There are five microservice projects in the system, and they only have the basic functionality, and they are not having any security-related or any other non-essential but important functionalities, so we are predicting the code based on the current usage of the line of code [55]–[57]

1. Comment search service:
    1. Current line code count -> 500
    2. Predicted line count (with all non-functional changes) -> 2.5KLOC
2. Comment generator helper service:
    1. Current code count -> 600
    2. Predicted line count (with all non-functional changers) -> 3KLOC
3. File upload helper service:
    1. Current code count -> 300
    2. Predicted line count (with all non-functional changers) -> 2KLOC
4. Client UI:
    1. Current code count -> 2KLOC
    2. Predicted line count (with all non-functional changers) -> 5KLOC
5. Comment Generator:
    1. Current code count -> 4KLOC
    2. Predicted line count (with all non-functional changes and another language support) -> 10KLOC

Overall, all predicted code line count is 22.5 KLOC is there in the application

A quick analysis of the COCOMO model

Effort = a (KLOC)^{b} person month

Time = c (Effort)^{d} Months

We are going to calculate the Effort & Time based on each COCOMO difficulty level. The following is to get a rough idea about the time frame for the development.

Here, we are given KLOC = 400

4.5.3.1 Basic Model Analysis

Software Projects	a	b	c	d
Organic	2.4	1.05	2.5	0.38
Semi Detached	3.0	1.12	2.5	0.35
Embedded	3.6	1.20	2.5	0.32

FIGURE 26: COCOMO VALUES

1. Organic:
   Effort = a (KLOC)b person month

   = 2.4 (22.5)^1.05 person month

   = 63 person month

   Time = c (Effort)^d Months

   = 2.5 (63)^0.38 Months

   = 12.1 Months

2. Semi-detached
   Effort = a (KLOC)b person month

   = 3 (22.5)^1.12 person month

   = 98 person month

   Time = c (Effort)^d Months

   = 2.5 (98) 0.35 Months

   = 12.4 Months

3. Embedded
   Effort = a (KLOC)b person month

   = 3.6 (22.5)^1.2 person month

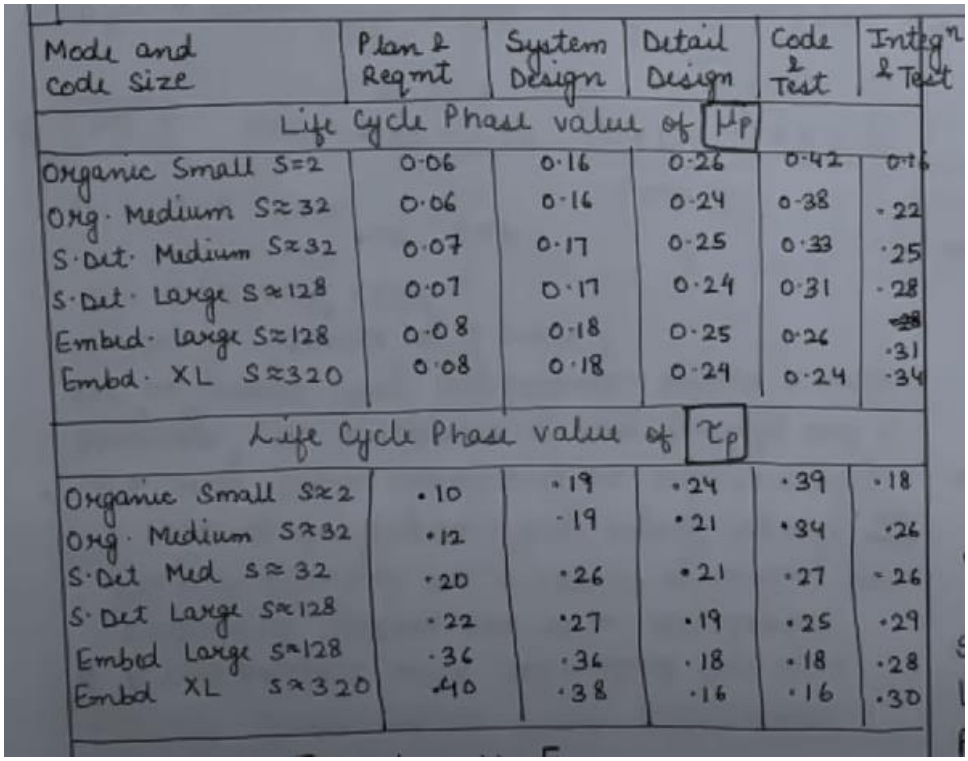
   = 151 person month

   Time = c (Effort)^d Months

   = 2.5 (151)^0.32 Months

   = 12.45 Months

4.5.3.2 Detailed COCOMO analysis



Mode and code size	Plan & Reqmt	System Design	Detail Design	Code & Test	Integ^n & Test
Life Cycle Phase value of μ_p					
Organic Small S=2	0·06	0·16	0·26	0·42	0·16
Org. Medium S≈32	0·06	0·16	0·24	0·38	·22
S·Det· Medium S≈32	0·07	0·17	0·25	0·33	·25
S·Det· Large S≈128	0·07	0·17	0·24	0·31	·28
Embed· large S≈128	0·08	0·18	0·25	0·26	·28 ·31
Embd· XL S≈320	0·08	0·18	0·29	0·24	·34
Life Cycle Phase value of τ_p					
Organic Small S≈2	·10	·19	·24	·39	·18
Org. Medium S≈32	·12	·19	·21	·34	·26
S·Det Med S≈32	·20	·26	·21	·27	·26
S·Det Large S≈128	·22	·27	·19	·25	·29
Embd Large S≈128	·36	·36	·18	·18	·28
Embd XL S≈320	·40	·38	·16	·16	·30

FIGURE 27: COCOMO VALUETABLE

In the detailed model, 15 cost drivers going to take into the picture in this architecture design. We have identified following are the drivers & they will have the respective status of the model.

- Product Attributes
  - Required Software Reliability: Very High -> 1.40
  - Size of Application Database: Low -> 0.94
  - Complexity of The Product: Low -> 0.85
- Hardware Attributes
  - Runtime Performance Constraints: High -> 1.11
  - Memory Constraints: High -> 1.21
  - Volatility of the virtual machine environment: Very Low
  - Required turnabout time: Very Low
- Personnel attributes
  - Analyst capability: Normal -> 1.0
  - Applications experience: Very High -> 0.82
  - Software engineer capability: Very High -> 0.70
  - Virtual machine experience: Normal -.> 1.0
  - Programming language experience: Very High
- Project Attributes

- Application of software engineering methods: Very High -> 0.82
- Use of software tools: Very High -> 0.83
- Required development schedule: Normal -> 1.00

So, following will be the Effort Adjustment Factor

- EAF = 1.40* 0.94*0.85*1.11*1.21*1.0*0.82* 0.70* 1.0* 0.82* 0.83* 1.00 = 0.5869

In this project it just has 22.5 KLOC so it will just denote as an organic project SO following will be the detail analysis

Organic:

Effort = a (KLOC)b * EAF person month

= 3.2 (22.5)^1.05*EAF person month

= 84.13* 0.5869 = 49.38

Time = c (Effort)^d Months

= 2.5*(49.38)^0.38 Months

= 11 Months

Effort

- Plan & Requirements = 0.06* 49.38 = 2.9628 person month
- Design = 0.16* 49.38 = 7.9008 person month
- Detail Design = 0.26* 49.38 = 12.8388 person month
- Code & Test = 0.42* 49.38 = 20.7396 person month
- Integration & Test = 0.16* 49.38 = 7.9008 person month

Development Time

- Plan & Requirements = 0.10* 11 = 1.1 month
- Design = 0.19* 11 = 2.09 month
- Detail Design = 0.24* 11 = 2.64 month
- Code & Test = 0.39* 11 = 4.29 month
- Integration & Test = 0.18* 11 = 1.98 month

4.5.4 COST-BENEFIT ANALYSIS METHOD (CBAM)

This Cost Benefits Analysis Method primarily focused on the cost side of each decision we made during the architecture creation, mainly after the ATAM process. We are going to have a benefit calculation that we are going to have after implementing the process. This Cost-benefit analysis is a mathematical approach that allows investors to consider whether they benefit after implementing the project in the production environment. This method has two purposes. One verifies that investment expected benefits more than its cost, and another is the investment to benefit ratio comparison.

Assumptions

- Pay for the following
    - One developer (1H will be 35 USD)
    - One manager (1H will be 50 USD)
    - One architect (1H will be 45 USD)
    - One team lead (1H will be 40 USD)
- Resource cost is for the three environments that we are going to maintain dev/pilot production
- When creating the indirect cost, we are considering all employees will take the evening food and the transportation for the evening and, we assume all the employees are from 50KM radios.
- The project employee team count is 3, with 15 total employees for the project.
- All the employees are working from home, and no other additional expenses are needed.
- 1st Year, we are assuming 10000 users will use the product.

4.5.4.1 COST ANALYSIS

1. Direct Cost

This direct cost will include all the resources cost three environments that we will provide to maintain excellent software maintenance health. Also, it consists of the total development cost with the yearly maintenance cost. It then includes QA effort to give the best quality product following a summary of the all-cost analysis. All the cost analysis documents will be attached as an appendix.[58]

Item	Monthly	Yearly
Total Direct Cost for resources	**$1,186.38**	**$14,236.56**
Total Direct Cost for development		**$22,511.25**
1st Year Direct Cost		**$36,747.81**
2nd Year to Another year Direct Cost (Maintenance cost 40% of the development cost)		**$23,241.06**

FIGURE 28: DIRECT COST TABLE

2. Indirect Cost

This is a cost that is not related to the project but needs to be in a cost analysis because these also need to be included while checking the total project cost

Per month there will be 20 working days.

Item	Monthly	Yearly
Per- person food cost per day: 2 USD	**$40.00**	**$480.00**
Per- person transportation cost per day : 3 USD	**$60.00**	**$720.00**
Per- person internet cost per day: 3 USD	**$60.00**	**$720.00**
Per person infrastructure cost: 5 USD	**$100.00**	**$1,200.00**
Per person Total cost	**$260.00**	**$3120.00**
Total Indirect cost for the project (5 members)	**$1300.00**	**$15,600.00**

FIGURE 29: INDIRECT COST TABLE

3. Intangible costs

Here we plan to give the following packages for the users who will get this product to generate comments for their projects. Since this is a new product, we are planning to give some free quota for the users. However, it is still a cost for the company, so this will be an intangible cost, but it will be a customer satisfaction mark because they can check the product whether it is helpful for them.

The free tire will be a 5-day comment generate period for a given user So

per-user cost for 24H trial	**$0.2**
expected user cost for trial	$2,000.00

The total estimated cost for the project is the following:

Total cost for the project 1st Year + added cost buffer	**$64500**
Total cost for the project after 1st Year + added cost buffer	**$55000**

[59]–[62]

4.5.4.2 BENEFITS

When we complete this software as a product, we will get a considerable amount of ROI for the investors, so we expect to get 20% of ROI in the beginning year, which is a bit higher. Still, an achievable amount, so the following will be the plan for that. These figures were taken from Forbes and other business sites. According to them, most of the current companies are trying to achieve that amount.

For the calculation, first, we have identified the cost that will be there while developing the software as a service, and for the first year, it was $110,747.81. Then following will be the profit need to take if needed to achieve that expected ROI from the new project.

ROI = Profit / Investment * 100% = (x / 64500)*100% = 60%

So, to get this ROI, profit should be = $38,700.00 so total earning should be $103,200.00. This much money needed to get in the first year to get that much we must have the following number of users our yearly cost will be $52.00 per person, and we need to have 2000 users in the first year

103,200.00/52 ~ 2000

Here I'm excluding all the marketing costs because we cannot send or products to the users without commercial ads or campaigns.

According to [63]–[66] Forbes, if the ROI is greater than 60%, it is a worthwhile investment.

4.6 EVALUATION (INDUSTRY EXPERTS)

I have given my architecture design to some of the industry experts, and they have given their suggestions. The following are the suggestions given by them.

Senior Architects viewpoint

According to his review he said that this system is having all the necessary components according to him we have several enhancements that can be done to the architecture. In the current one client call directly to the three services that we have exposed but if we can have a wrapper library to the client or an API gateway that would reduce the complication added to the service end. Another suggestion is to move the file upload part directly to the client end then there will not much latency on the file upload these are the enhancements that we can do to this system. Also, he has gone through the project costing done & we have done some slight changers to the costing model to align with normal software development project.

Updated architecture after his review comments

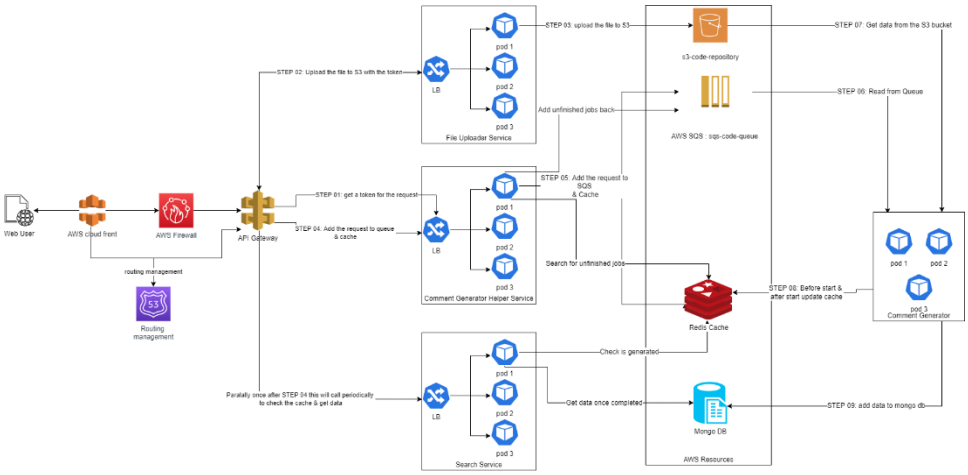


FIGURE 30: SENIOR ARCHITECTS UPDATED DIAGRAM

Done an evaluation by a Tech lead

According to him there are few enhancements that we can do to the overall architecture but current one is better for the version one according to his comments because architectures is need continuous improvements and we cannot give a fully optimized architecture at once. Following are the improvements that he is suggesting.

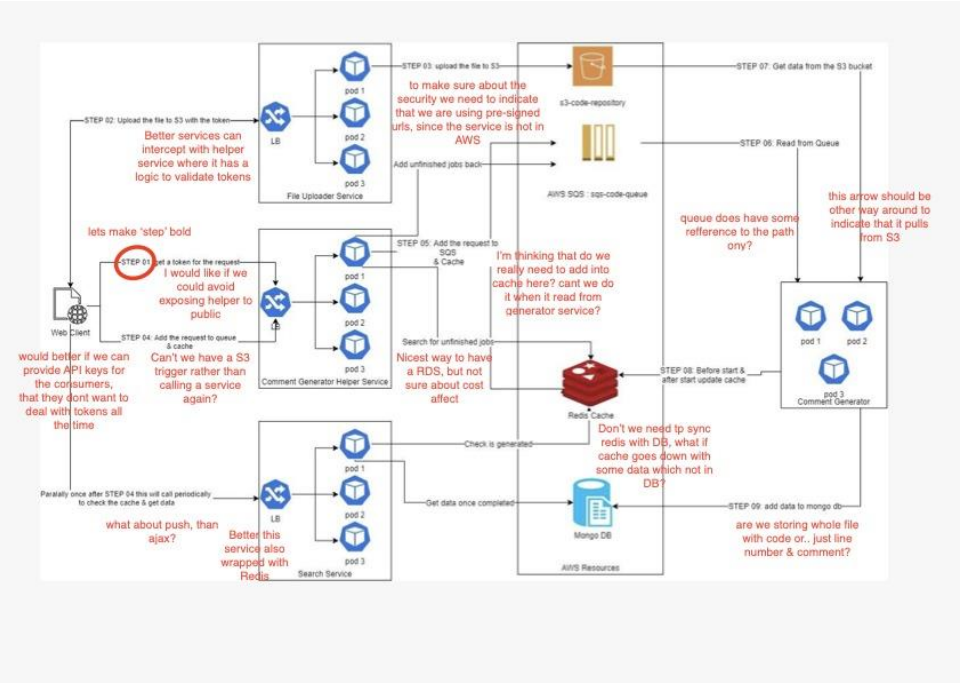


FIGURE 31: SENIOR TECH LEAD UPDATED DIAGRAM

According to the other evaluations also they seem to be ok with the current architecture since I have covered all the details of a good architecture design.

5 PERFORMANCE

After that I have done a full load testing on the architecture with the following system capabilities. For this testing I have used the local environment with the following system requirements

- Processor: Intel(R) Core (TM) i5-10210U CPU @ 1.60GHz 2.11 GHz
- Ram: 16.0 GB (15.8 GB usable)

I have used the local AWS environment (local stack) and Kubernetes local to start the application stack since AWS resource cost is bit high for the prototype.

Following is the outcome of the testing here we have used the JMeter for the API testing for this we have used all the API's that we are using and we have simulated the same process that user is going to do that means generating a token then uploading the file to S3 bucket and scheduling the comment generation finally searching the comment in the Redis & get it from the Mongo DB. For these we have simulated 200 each set of requests (this will simulate 200 users) in a given 5 second duration & that was the highest we can go with the local setup since we are not using AWS resources.

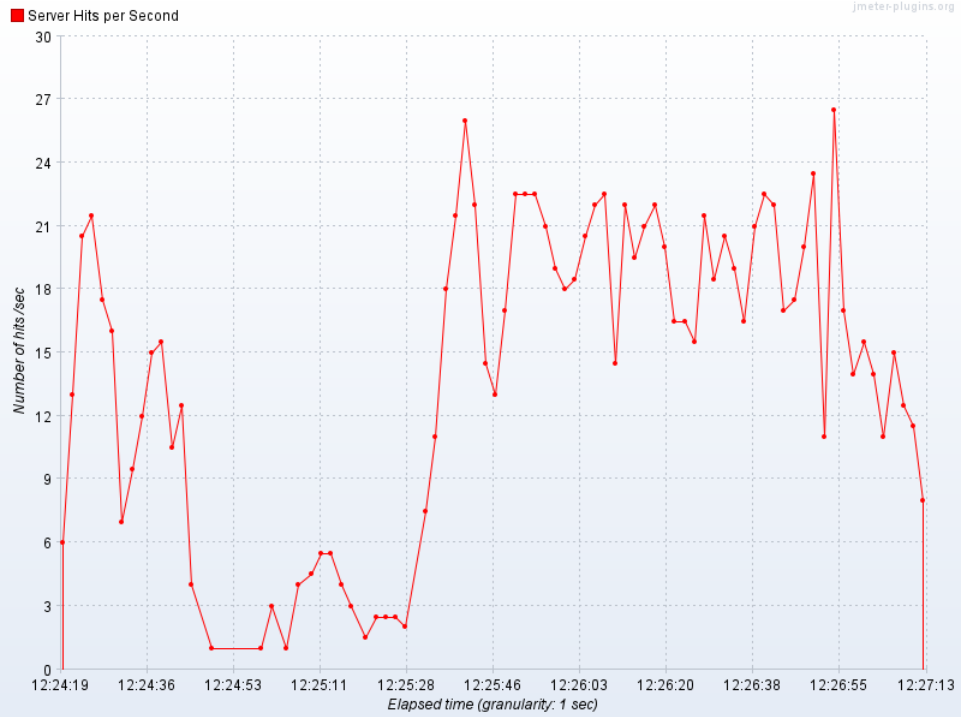


FIGURE 32: NUMBER OF HITS IN GIVEN TIME INTERVAL

According to the following graph we can see how much hits were taken by the server this graph stands for the per second request came rate & according to the statistics it can handle considerable number of users in each time even running the service locally also we can do some performance enhancements for the file upload to directly upload the files via the clients that we are going to build.

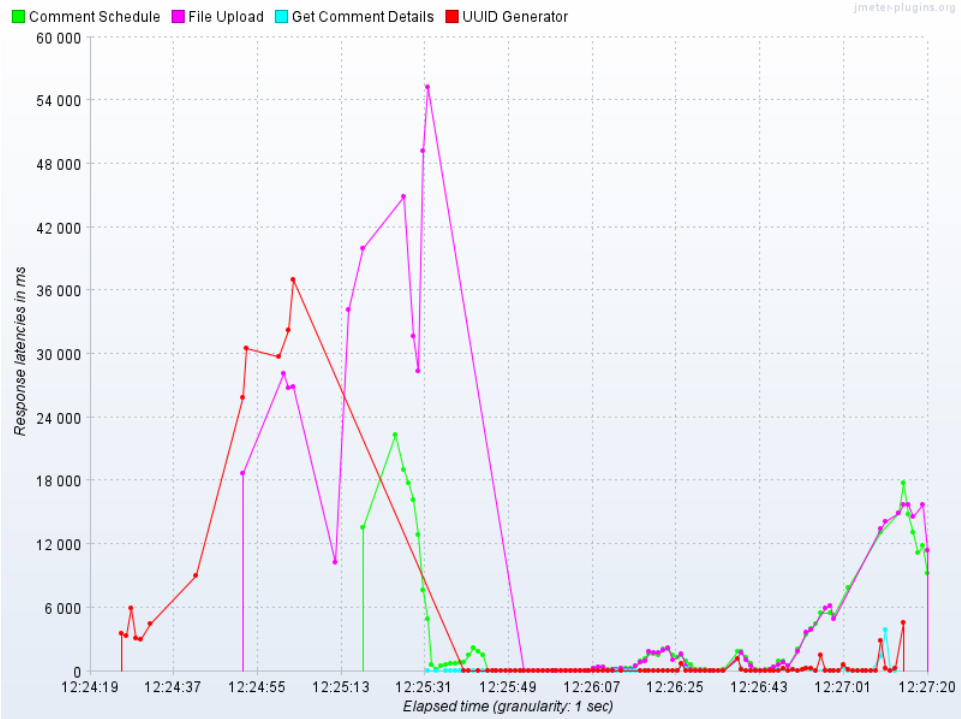


FIGURE 33: NUMBER OF REQUESTS FOR GIVEN API

According to this graph it shows the latency of the API calls we have 4 API calls & File upload is the one which is having the highest latency among the other applications. And it only happened in the first half of the then it reduced to very low latency.

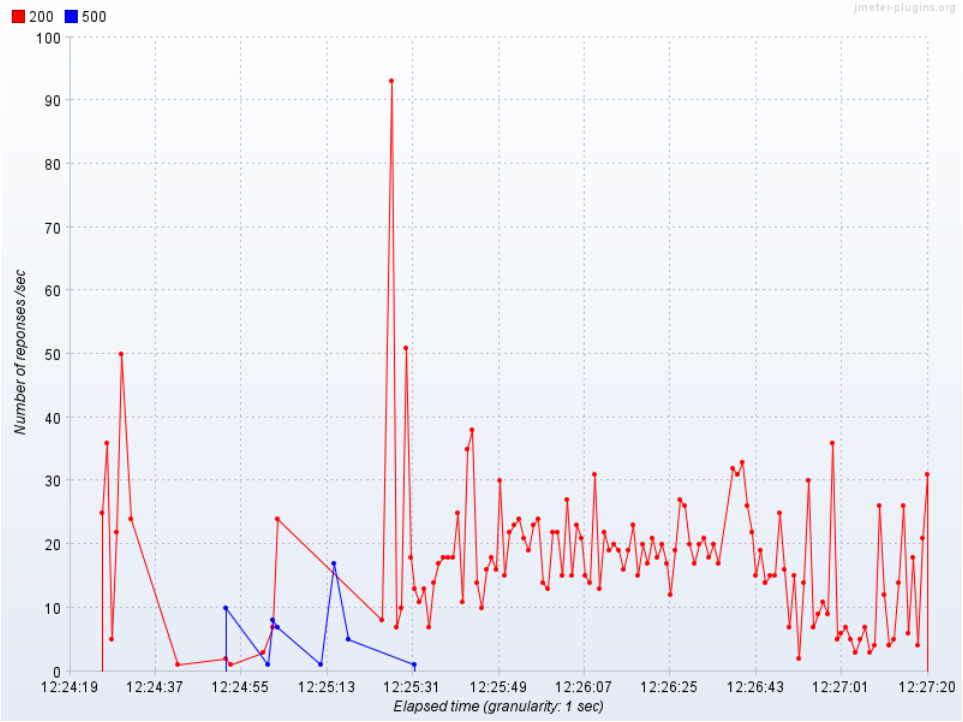


FIGURE 34: NUMBER OF 200 AND 500 RESPONSES

Above graph will show how much 300 responses came in each time period so you can see we only getting 200 mainly with 500 statuses when I go through the error log that was happened due to the local stack and it is not capable to handle more requests it only simulate AWS to a certain level if we have an actual AWS then we were able to have much more satisfactory results for this performance analysis

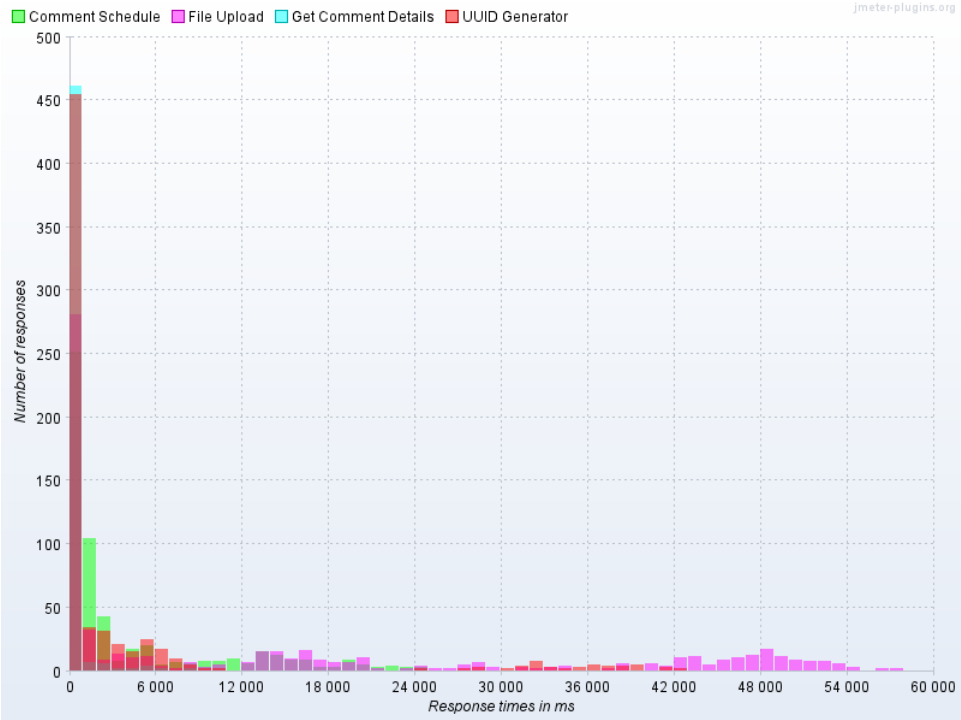


FIGURE 35: REQUEST RATE

According to this graph we can see when did each rest call happen and how much responses were there after the call so for the UUID generator which is going to generate token. Getting the highest request rate & the response rate likewise File upload also getting the correct responses back for each time period this is another indicator that our API's are performing nicely.

When we requested to generate comments, we had following number of active threads after 3 min down the line. You can see it is taken that much of threads to schedule the comment generation and to upload the files.

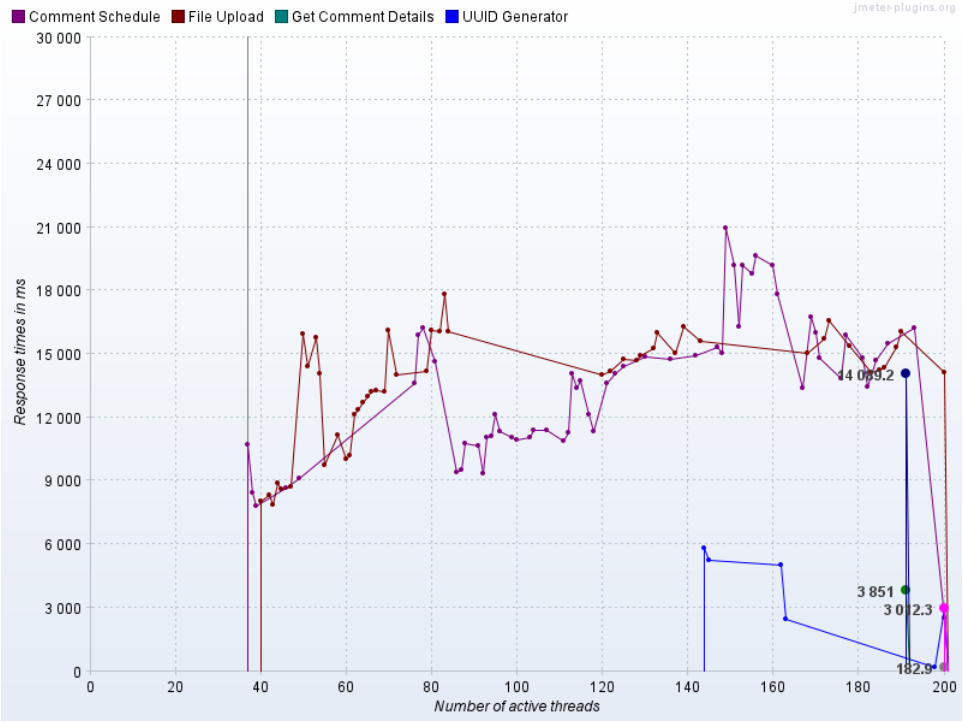


FIGURE 36: NUMBER OF TRANSACTIONS WERE DONE

In the following graph will show how much transactions were done and what are the success full once and what are the error one during the 3 minutes time period.

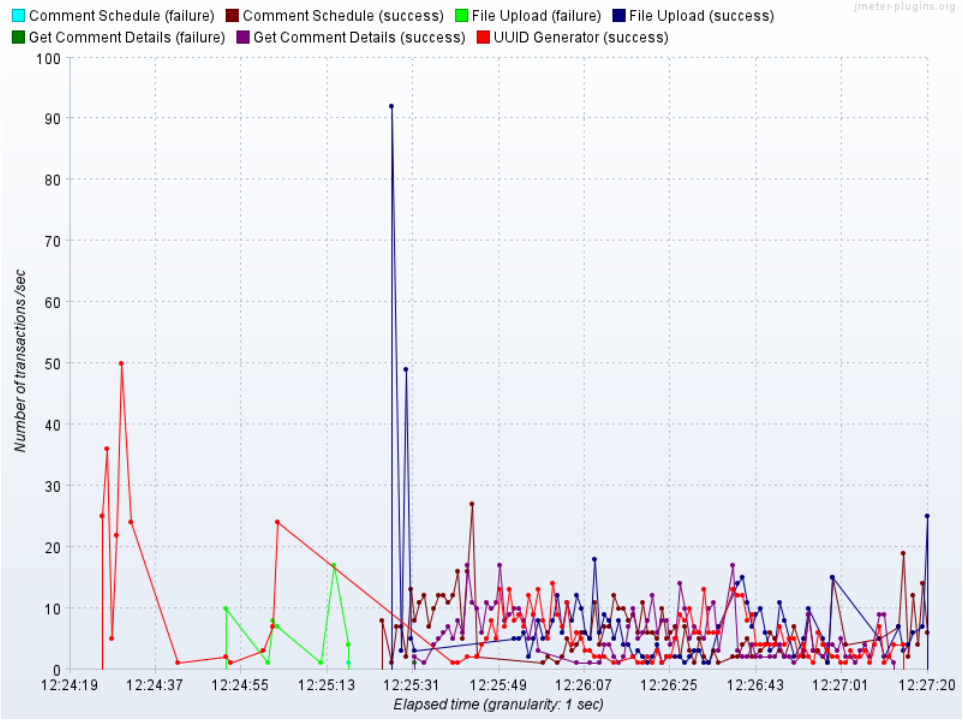
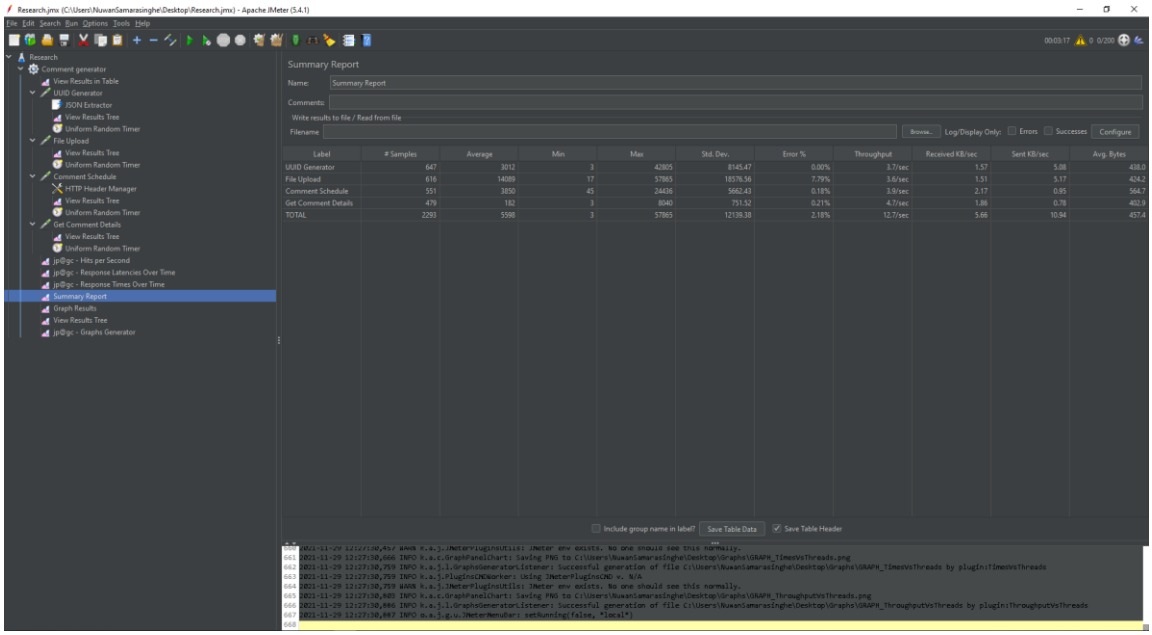


FIGURE 37: PERFORMANCE SUMMERY

Following is the summary of the performance analysis



According to this you can see for the 4 REST calls we have very less amount of error rate for this error rate also happen due to the local AWS environment if we deployed these to the actual environment this will not happen so according to this total error rate is 2.18% which is good. Also, we can see a good average through put on the system & we have taken these figures for mor than 400 calls per each request.

6 SUMMARY

During this research, we tried to give a solution to reduce the time taken to put comments on the code since it takes some valuable time from the software engineers. Also, these comments are beneficial when we are trying to add new features or when doing the code changes related to the customer needs or fixing bugs. Once we have a tool like these, developers will not need to create those comments.

In this research, our focus is to give a cost-effective & high available architecture to a software code comment generation tool. After going through many research articles and projects, we have selected a commenting tool capable of generating comments. Then we went through a couple of other projects related to comment generation & researched what they have done to create a usable product. Still, I could not find any architecture research has done for those most of the studies are designed to test the NLP engine capabilities none of them had a better architectural view to bringing that as a product to sell. Once I selected the comment generator tool in the first face, I went through a list of components that need to use while converting the existing mechanism. After the selection of the component, I have generated a simple architecture. Then had a brainstorming session with industry specialists to determine the architecture with their experience. So, I have selected the microservice architecture concept with high availability. While developing the conceptual architecture, I tried to create it as an independent component and given asynchronous behaviour to give easy availability. After a couple of sessions, I have selected an architecture that could develop.

Before going into the development, I have done ATAM & the CBAM process to identify the changers & costs that will take this project to develop because the price is high. The gain is less than this want a practical design, so while doing the architecture design, I have gone through the list of items that need to do (list of tasks for the project) & assigned points based on the workload. These Estimations are rough estimates but based on that I have calculated the cost for the development it already having 15% from the total development with QA efforts will be management there will be 601 hours' worth of work in the project that means 3 full time months of work will be there for a single developer but we have planned to use 1 developer and a 1 team manager with single team lead and a single architect this project will totally cost $22,511.25 because each person will have different job role to play and their pay also different, I have taken some assumptions while doing the cost calculation such as these will get following amount of money at a given time period 1 developer (1H will be 35 USD) 1 manager (1H will be 50 USD) 1 architect (1H will be 45 USD) 1 team lead (1H will be 40 USD) and while doing a project there are some indirect cost factors like following food infrastructure and transportation cost for the each member of the team here also I did an assumptions before calculating the total costs.

My initial plan was to use a couple of EC2 on the AWS environment. Still, after going through Kubernetes, they maintain most critical stuff like auto-scaling

maintaining multiple pods with one single endpoint, and load balancing functionality. Because of those, I have selected that as the final implementation method. While calculating the cost, I have chosen two PC environments to create the Kubernetes environment, and for the environment setup, it will cost $14,236.56. This cost will be recurring every year. All the cost calculation documentation is attached as an appendix. Finally, according to Forbs magazine, I have calculated the cost-benefit for the project. ACCORDING TO THE CALCULATIONS, if ROI is more excellent than 7%, it is a worthwhile investment in this project ROI is 30%. Also, all these resources are automated create I have used Terraform script to do so. It just needs AWS access & secrets to generate the system instantly.

After doing the partial cost analysis & the partial architecture design decision, I have started developing the components. While developing the components, I converted the NLP engine as a scheduling service, and others were just rest services. Also, I have used a couple of AWS services with Redis and mongo databases to save changes. This research focus is to create an architecture that is feasible as a product that can build.

7 PROBLEMS AND CHALLENGES

While doing the development, the biggest challenge was finding proper research that has done its best to generate code comments. In my research, I am just trying to give a functional architecture that can be implemented then used in the industry to reduce the comment generation time in each software project. Once I find a project, it works well. Still, it needed a lot of code changes required. For example, using AWS resources like S3 and Redis like cheche solutions and SQS queues was not recommended. So I had to change the comment generator to work as a scheduler service.

Selecting the components to create the architecture and the best way to attach those (architecture development) and how to evaluate the architecture and do the changers less amount of help on those in forums needed to dig deep to get an idea about how to assess architecture and how to calculate the cost & benefits of the system. I needed to go through deeply about the ATAM process and CBAM processors to evaluate my architecture and find other architecture evaluation methods.

Selecting AWS resources for the project because some of the resources are very pricy to use, so I needed to go through some alternatives for those resources. For example, mongo has a mongo-managed AWS cluster but compared to cost, it was a bit low if we used mongo without managed services. Also, for the research, it was hard to use actual AWS resources because it costs me, so I found an alternative to AWS local Stack is the one it will just mock the AWS resources and then act like one of them solved the resource issue.

I have gone through a step-by-step process of creating the architecture initially; my approach was to use a couple of EC2's in the AWS environment. Still, once I go through the cost estimations, I know that price is a bit high and maintaining the high availability is a bit hard in that environment then, I tried to use docker as a solution. Still, we also need more work that needs to be done to maintain the scalability but a bit less work than the EC2 approach then. After that, I moved my architecture into a docker swam environment there; I went through Kubernetes environment & according to them that I had the entire infrastructure ready to move I have created the images on docker hum & swam service is up and running I just wanted to do it lean the Kubernetes and convert it to that, so that was the challenging part need to learn a lot to do that because we have to create the services and the containers in that.

8 FUTURE WORK

I plan to expand this from Java to other languages in future work because currently, we only support Java. Still, in the future, we can enhance it to other languages.

Instead of self-managing resources Ex Kubernetes, we can convert this to AWS managed service like Elastic container Service or similar Google Kubernetes cluster after evaluating the prices.

Improving the NLP service to give more realistic comments needs some improvements to do so. Basic functionality is working only left is just adjustment to the application into the better summery builder.

Based on the severity of the code that we are trying to generate a comment giving different types of options because some of the users may not need to save the code on our end because that will be a massive vulnerability to them. Also, we are planning to use end-to-end encryption to improve security. Currently, there is no login method to create their accounts & manage the projects on their own that is a future work that I'm planning to do.

Convert all the unmanaged services to AWS managed services as much as possible. For the resources currently using AWS, it is not an AWS environment. It is a mock AWS environment, so there may be a few challenges while creating the environment on the AWS side.

9 CONCLUSION

According to the evaluation I did for the comment generation architecture, it is worthy of the building because ROI is higher and benefits are better. We have used the most cost-effective methods like AWS managing services with reserved instances, and we are using the Kubernetes environment to deploy our changes. Management wise also projects well with multiple teams. We can archive this project without any issues. According to the cost analysis, we can see its cost is less. Currently, for the research, we are using the following environment for AWS. We are using the local stack to mock all the AWS-related resources, but we use Terraform to generate with simple changers we can create the full resource stack on AWS. According to the cost analysis we did following is the summary report for the cost.

#	Item	Cost Yearly
1	Total Cost for resources	**$14,236.56**
2	Total Cost for development	**$22,511.25**
3	1st Year Cost: development + resources	**$36,747.81**
4	2nd Year to Other year Resource cost + Maintenance cost 40% of the development cost	**$23,241.06**
5	Total Indirect cost for the project (food + transportation + internet + infrastructure)	**$72,000.00**
6	Free tire cost for a user	**$2,000.00**
7	Grand total cost for the project 1st Year	**$110,747.81**
8	Grand Total cost for the project after 1st Year	**$97,241.06**

When we go the cost & the benefit for the project and the ROI

Cost-benefit ratio = 0.77 years (about 9 months).

ROI: 30%

According to Forbes, if the ROI is greater than 7%, it is a worthwhile investment.

The investigation done for the architecture technical part is also best because we have archived our primary goals like high available auto-scalable architecture by giving a Kubernetes environment. The other resources are managed services. So, this is an exemplary architecture that can be used to develop.

10 REFERENCES

[1] J. Germain, “Programming - Commenting,” *School of Computing University Of Utah*. https://www.cs.utah.edu/~germain/PPS/Topics/commenting.html (accessed Jan. 01, 2020).

[2] L. M. Karam, “The Importance of Good Software Architecture - DZone Integration,” *dzone.com*. https://dzone.com/articles/the-importance-of-a-good-software-architecture (accessed Jan. 06, 2020).

[3] N. Sinha, “Concept of Comments in Computer Programming,” *GeeksforGeeks*, Aug. 29, 2018. https://www.geeksforgeeks.org/concept-of-comments-in-computer-programming/ (accessed Jan. 01, 2020).

[4] B. Sourour, “Putting comments in code: the good, the bad, and the ugly.,” *freeCodeCamp.org*, Apr. 20, 2017. https://www.freecodecamp.org/news/code-comments-the-good-the-bad-and-the-ugly-be9cc65fbf83/ (accessed Jan. 01, 2020).

[5] D. Van Tassel, “Comments in programming languages.” http://www.gavilan.edu/csis/languages/comments.html (accessed Jan. 01, 2020).

[6] GeeksForGeeks, “Comments in Java - GeeksforGeeks.” https://www.geeksforgeeks.org/comments-in-java/ (accessed Jan. 02, 2020).

[7] L. Moreno, J. Aponte, G. Sridhara, A. Marcus, L. Pollock, and K. Vijay-Shanker, “Automatic generation of natural language summaries for Java classes,” in *2013 21st International Conference on Program Comprehension (ICPC)*, San Francisco, CA, USA, May 2013, pp. 23–32. doi: 10.1109/ICPC.2013.6613830.

[8] E. Wong, J. Yang, and L. Tan, “Autocomment: Mining question and answer sites for automatic comment generation,” in *in Automated Software Engineering (ASE), 2013 IEEE/ACM 28th International Conference on, Nov 2013*, pp. 562–567.

[9] P. W. McBurney and C. McMillan, “Automatic documentation generation via source code summarization of method context,” in *Proceedings of the 22nd International Conference on Program Comprehension - ICPC 2014*, Hyderabad, India, 2014, pp. 279–290. doi: 10.1145/2597008.2597149.

[10] S. Haiduc, J. Aponte, L. Moreno, and A. Marcus, “On the Use of Automated Text Summarization Techniques for Summarizing Source Code,” in *2010 17th Working Conference on Reverse Engineering*, Beverly, MA, USA, Oct. 2010, pp. 35–44. doi: 10.1109/WCRE.2010.13.

[11] S. Reddy, “Vector space model,” *Wikipedia*. Oct. 18, 2021. Accessed: Nov. 07, 2021. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Vector_space_model&oldid=1050529441

[12] L. Europe, “Latent semantic analysis,” *Wikipedia*. Aug. 21, 2021. Accessed: Nov. 07, 2021. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Latent_semantic_analysis&oldid=1039966183

[13] S. Panichella, “CODES: mining source code Descriptions from develop ers discussions,” Jun. 2014.

[14] S. Panichella, J. Aponte, M. Di Penta, A. Marcus, and G. Canfora, “Mining source code descriptions from developer communications.” https://ieeexplore.ieee.org/document/6240510 (accessed Nov. 07, 2021).
[15] D. M. Blei, A. Y. Ng, and M. I. Jordan, “Latent Dirichlet Allocation,” *Journal of Machine Learning Research*, vol. 3, no. Jan, pp. 993–1022, 2003.
[16] D. Movshovitz-Attias and W. W. Cohen, “Natural Language Models for Predicting Programming Comments,” p. 6.
[17] M. M. Rahman, C. K. Roy, and I. Keivanloo, “Recommending Insightful Comments for Source Code using Crowdsourced Knowledge,” *2015 IEEE 15th International Working Conference on Source Code Analysis and Manipulation (SCAM)*, pp. 81–90, Sep. 2015, doi: 10.1109/SCAM.2015.7335404.
[18] Y. Oda *et al.*, “Learning to Generate Pseudo-Code from Source Code Using Statistical Machine Translation,” in *2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, Lincoln, NE, Nov. 2015, pp. 574–584. doi: 10.1109/ASE.2015.36.
[19] K. Cho, B. van Merrienboer, D. Bahdanau, and Y. Bengio, “On the Properties of Neural Machine Translation: Encoder-Decoder Approaches,” *arXiv:1409.1259 [cs, stat]*, Oct. 2014, Accessed: Nov. 07, 2021. [Online]. Available: http://arxiv.org/abs/1409.1259
[20] S. Iyer, I. Konstas, A. Cheung, and L. Zettlemoyer, “Summarizing Source Code using a Neural Attention Model,” in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Berlin, Germany, 2016, pp. 2073–2083. doi: 10.18653/v1/P16-1195.
[21] Wenhao Zheng, Hong-Yu Zhou, Ming Li, Jianxin Wu, “Code Attention: Translating Code to Comments by Exploiting Domain Features.” https://arxiv.org/pdf/1709.07642.pdf%7D.
[22] X. Hu, G. Li, X. Xia, D. Lo, and Z. Jin, “Deep code comment generation,” May 2018, pp. 200–210. doi: 10.1145/3196321.3196334.
[23] Y. Liang and K. Q. Zhu, “Automatic Generation of Text Descriptive Comments for Code Blocks,” p. 8.
[24] W. U. Ahmad, S. Chakraborty, B. Ray, and K.-W. Chang, “A Transformer-based Approach for Source Code Summarization,” *arXiv:2005.00653 [cs, stat]*, May 2020, Accessed: Nov. 07, 2021. [Online]. Available: http://arxiv.org/abs/2005.00653
[25] Y. Wan *et al.*, “Improving Automatic Source Code Summarization via Deep Reinforcement Learning,” *arXiv:1811.07234 [cs]*, Nov. 2018, Accessed: Nov. 07, 2021. [Online]. Available: http://arxiv.org/abs/1811.07234
[26] M. Allamanis, H. Peng, and C. Sutton, “A Convolutional Attention Network for Extreme Summarization of Source Code,” p. 10.
[27] L. Mou, G. Li, L. Zhang, T. Wang, and Z. Jin, “Convolutional Neural Networks over Tree Structures for Programming Language Processing,” *arXiv:1409.5718 [cs]*, Dec. 2015, Accessed: Nov. 07, 2021. [Online]. Available: http://arxiv.org/abs/1409.5718
[28] N. Khamis, R. Witte, and J. Rilling, *Automatic Quality Assessment of Source Code Comments: The JavadocMiner*, vol. 6177. 2010, p. 79. doi: 10.1007/978-3-642-13881-2_7.

[29] D. Steidl, B. Hummel, and E. Juergens, “Quality analysis of source code comments,” in *2013 21st International Conference on Program Comprehension (ICPC)*, San Francisco, CA, USA, May 2013, pp. 83–92. doi: 10.1109/ICPC.2013.6613836.

[30] X. Song, H. Sun, X. Wang, and J. Yan, “A Survey of Automatic Generation of Source Code Comments: Algorithms and Techniques,” *IEEE Access*, vol. PP, pp. 1–1, Jul. 2019, doi: 10.1109/ACCESS.2019.2931579.

[31] “Comparing Common Programming Languages to Parse Big XML File in Terms of Executing Time, Memory Usage, CPU Consumption and Line Number on Two Platforms,” *ResearchGate*. https://www.researchgate.net/publication/309022617_Comparing_Common_Programming_Languages_to_Parse_Big_XML_File_in_Terms_of_Executing_Time_Memory_Usage_CPU_Consumption_and_Line_Number_on_Two_Platforms (accessed Jan. 10, 2020).

[32] P. Niedringhaus, “Using Java to Read Really, Really Large Files,” *Medium*, Jan. 04, 2019. https://itnext.io/using-java-to-read-really-really-large-files-a6f8a3f44649 (accessed Jan. 10, 2020).

[33] C. Richardson, “Microservices Pattern: Microservice Architecture pattern,” *microservices.io*. http://microservices.io/patterns/microservices.html (accessed Apr. 05, 2021).

[34] J. P. Irudayaraj, “Adoption Advantages Of Micro-Service Architecture In Software Industries,” vol. 8, no. 09, p. 4, 2019.

[35] Paolo Di Francesco, Patricia Lago, and Ivano Malavolta, “(PDF) Research on Architecting Microservices: Trends, Focus, and Potential for Industrial Adoption.” https://www.researchgate.net/publication/317071768_Research_on_Architecting_Microservices_Trends_Focus_and_Potential_for_Industrial_Adoption (accessed Apr. 05, 2021).

[36] Jetinder Singh, “The What, why, and How of a Microservices Architecture | by Hashmap | HashmapInc | Medium.” https://medium.com/hashmapinc/the-what-why-and-how-of-a-microservices-architecture-4179579423a9 (accessed Apr. 05, 2021).

[37] C. M. Aderaldo, N. C. Mendonça, C. Pahl, and P. Jamshidi, “Benchmark Requirements for Microservices Architecture Research,” in *2017 IEEE/ACM 1st International Workshop on Establishing the Community-Wide Infrastructure for Architecture-Based Software Engineering (ECASE)*, May 2017, pp. 8–13. doi: 10.1109/ECASE.2017.4.

[38] B. Terzić, V. Dimitrieski, S. Kordić (Aleksić), and I. Luković, *A Model-Driven Approach to Microservice Software Architecture Establishment*. 2018, p. 80. doi: 10.15439/2018F370.

[39] S. Barakat, “Monitoring and Analysis of Microservices Performance,” *Journal of Computer Science and Control Systems*, vol. 10, pp. 19–22, May 2017.

[40] Docker, “Docker overview,” *Docker Documentation*, Oct. 22, 2021. https://docs.docker.com/get-started/overview/ (accessed Oct. 27, 2021).

[41] Docker, “Swarm mode key concepts,” *Docker Documentation*, Oct. 22, 2021. https://docs.docker.com/engine/swarm/key-concepts/ (accessed Oct. 27, 2021).

[42] Kubernetes, "What is Kubernetes?" *Kubernetes*. https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/ (accessed Oct. 27, 2021).
[43] Localstack, "Integrations," *Docs*. https://docs.localstack.cloud/integrations/ (accessed Oct. 27, 2021).
[44] Aqua, "Docker Containers vs. Virtual Machines," *Aqua*. https://www.aquasec.com/cloud-native-academy/docker-container/docker-containers-vs-virtual-machines/ (accessed Oct. 27, 2021).
[45] Molly Clancy, "Docker Containers vs. VMs: Pros and Cons of Containers and Virtual Machines," *Backblaze Blog | Cloud Storage & Cloud Backup*, Oct. 15, 2021. https://www.backblaze.com/blog/vm-vs-containers/ (accessed Oct. 27, 2021).
[46] Simran Arora, "Docker vs. Virtual Machines: Differences You Should Know - Cloud Academy." https://cloudacademy.com/blog/docker-vs-virtual-machines-differences-you-should-know/ (accessed Oct. 27, 2021).
[47] Weave, "Business Benefits of Kubernetes." https://www.weave.works/blog/business-benefits-of-kubernetes (accessed Oct. 27, 2021).
[48] D. Fontani, "Five good reasons for using Kubernetes," *Medium*, Nov. 15, 2020. https://towardsdatascience.com/5-reason-for-using-kubernetes-b7ade82eda90 (accessed Oct. 27, 2021).
[49] Localstack, *Overview*. LocalStack, 2021. Accessed: Oct. 27, 2021. [Online]. Available: https://github.com/localstack/localstack
[50] Nghia Le, "Architectural Driver," 17:06:23 UTC. Accessed: Aug. 08, 2021. [Online]. Available: https://www.slideshare.net/NghiaLe36/architectural-driver
[51] J. K. Bergey, M. J. Fisher, and L. G. Jones, "Use of the Architecture Tradeoff Analysis MethodSM (ATAMSM) in Source Selection of Software-Intensive Systems:" Defense Technical Information Center, Fort Belvoir, VA, Jun. 2002. doi: 10.21236/ADA403813.
[52] Software Engineering Institute, "Architecture Tradeoff Analysis Method Collection." https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=513908 (accessed Apr. 14, 2021).
[53] Dan Shewan, "How to Do a SWOT Analysis for Your Small Business (with Examples)." https://www.wordstream.com/blog/ws/2017/12/20/swot-analysis (accessed Oct. 27, 2021).
[54] Stephen J. Bigelow, "What is a SWOT Analysis?" *SearchCIO*. https://searchcio.techtarget.com/definition/SWOT-analysis-strengths-weaknesses-opportunities-and-threats-analysis (accessed Oct. 27, 2021).
[55] kalopsia, "Software Engineering | COCOMO Model," *GeeksforGeeks*, Apr. 13, 2018. https://www.geeksforgeeks.org/software-engineering-cocomo-model/ (accessed Oct. 27, 2021).
[56] Java t point, "Software Engineering | COCOMO Model - javatpoint," *www.javatpoint.com*. https://www.javatpoint.com/cocomo-model (accessed Oct. 27, 2021).
[57] mayankjtp, "Constructive Cost Model (COCOMO)," *Tutorial And Example*, Feb. 01, 2020. https://www.tutorialandexample.com/constructive-cost-model-cocomo/ (accessed Oct. 27, 2021).

[58] Nuwan Samarasinghe, “AWS Price Calculator saved.” [Online]. Available: https://calculator.aws/#/estimate?id=4250dd2e26f70070c6670de56e9eacd05277032f
[59] Torsten Mallée, “ROI of IT projects: 3-step calculation example.” https://www.aeb.com/intl-en/magazine/articles/3-step-roi-calculation-it-projects.php (accessed Oct. 27, 2021).
[60] Mona Lebied, “IT ROI Made Easy: How to Calculate The ROI For IT Projects,” *BI Blog | Data Visualization & Analytics Blog | datapine*, May 04, 2018. https://www.datapine.com/blog/how-to-calculate-it-roi/ (accessed Oct. 27, 2021).
[61] brightlineit, “How to Calculate ROI for IT Projects | Brightline IT | Planning for IT,” *Brightline Technologies*, Jun. 12, 2018. https://brightlineit.com/how-to-calculate-roi-for-it-projects/ (accessed Oct. 27, 2021).
[62] ANDREW BEATTIE, “How to Calculate Return on Investment (ROI),” *Investopedia*. https://www.investopedia.com/articles/basics/10/guide-to-calculating-roi.asp (accessed Oct. 27, 2021).
[63] Emily Guy Birken and Benjamin Curry, “Return on Investment (ROI) Definition – Forbes Advisor.” https://www.forbes.com/advisor/investing/roi-return-on-investment/ (accessed Aug. 08, 2021).
[64] Peter Landau, “Cost Benefits Analysis for Projects - A Step-by-Step Guide,” *ProjectManager.com*, Jun. 09, 2021. https://www.projectmanager.com/blog/cost-benefit-analysis-for-projects-a-step-by-step-guide (accessed Aug. 08, 2021).
[65] Matei Ripeanu, “Implementing the CBAM.” https://people.ece.ubc.ca/matei/EECE417/BASS/ch12lev1sec3.html (accessed Aug. 08, 2021).
[66] K. Rick, J. Asundi, and M. Klein, “(PDF) Quantifying the costs and benefits of architectural decisions.” https://www.researchgate.net/publication/3895392_Quantifying_the_costs_and_benefits_of_architectural_decisions (accessed Aug. 08, 2021).

11 APPENDIX

11.1 LIST OF FIGURES

11.2 SOFTWARE DEVELOPMENT TASK COST DOCUMENT

Task	Component	Point Estimation In Hours
Web Client		**40**
	Project initialization and basic structure creation	8
	Upload File component with service changers	16
	Show task output component with service changers	8
	how generated comment component with service changers	8
File Uploader Service		**106**
	Spring Boot Project initialization & basic structure creation	10
	S3 creation via terraform script	8
	File Uploader changes to support file upload and error handling	24
	Dockerization & creating docker hub project	8
	Check & test application through local docker	8
	Kubernetes YML creation for the service	8
	Deployment on the Kubernetes environment	16
	Monitoring the application	8
	Testing the application	16
Comment Generator Helper		**126**
	Spring Boot Project initialization & basic structure creation	10
	SQS creation via terraform script	8
	Redis Configuration	4
	comment generation helper changes to support file upload and error handling	24
	Dockerization & creating docker hub project	8
	Check & test application through local docker	8
	Kubernetes YML creation for the service	8
	Deployment on the Kubernetes environment	16
	Monitoring the application	8
	Testing the application	16
	Schedular changers to reinitiate the process if failed	16
Search Service		**106**
	Spring Boot Project initialization & basic structure creation	10
	SQS adding to the code	8

	Redis changes to the code	8
	comment generation helper changes to support file upload and error handling	24
	Dockerization & creating docker hub project	8
	Check & test application through local docker	8
	Kubernetes YML creation for the service	8
	Deployment on the Kubernetes environment	8
	Monitoring the application	8
	Testing the application	16
Comment Generator Engine		144
	Python Project Initialization	16
	SQS adding to the code	8
	Redis changes to the code	8
	Comment generation code changes to the existing generation algorithm	48
	Error handling of the code	8
	Dockerization & creating docker hub project	8
	Check & test application through local docker	8
	Kubernetes YML creation for the service	8
	Deployment on the Kubernetes environment	8
	Monitoring the application	8
	Testing the application	16
Total Hours with effort variance of 15% for the project management and other project related		**600.3**
Total Cost for the Development of the solution for 1 developer (1H will be 35 USD) 1 manager (1H will be 50 USD) 1 architect (1H will be 45 USD) 1 team lead (1H will be 40 USD)		**$22,511.25**

11.3 COST ANALYSIS FOR AWS RESOURCES

Item	Region	Description	Cost Monthly USD	Cost Yearly USD
Business support plan	All Regions	Supports 24/7 phone, chat, and email access to Cloud Support Engineers for unlimited contacts and a response time of less than 1 hour.	$100.00	$1,200.00
Amazon EC2	US East (N. Virginia)	Operating system (Linux), Quantity (2), Pricing strategy (EC2 Instance Savings Plans 1 Year No UpFront), Storage amount (80 GB), Instance type (t4g.2xlarge)	$262.16	$3,145.92
Amazon ElastiCache (Redis)	US East (N. Virginia)	Nodes (2 instances of type Redis Standard cache t2.micro OnDemand)	$24.82	$297.84
Amazon Simple Queue Service (SQS)	US East (N. Virginia)	DT Inbound: Not selected (0 TB per month), DT Outbound: Not selected (0 TB per month), Standard queue requests (2 million per month), FIFO queue requests (2 million per month)	$1.80	$21.60
Amazon Simple Storage Service (S3)	US East (N. Virginia)	S3 Standard storage (50 GB per month) DT Inbound: Not selected (0 TB per month), DT Outbound: Not selected (0 TB per month)	$6.68	$80.16
Total cost for resources			**$1,186.38**	**$14,236.56**

11.4 INDIRECT COST ANALYSIS

Project employee count (3 teams)	15	
Item	Monthly	Yearly
Per-person food cost per day : 2 USD	$40.00	$480.00
Per-person transportation cost per day : 10 USD	$200.00	$2,400.00
Per-person internet cost per day : 10 USD	$200.00	$2,400.00
Per person infrastructure cost: 10 USD	$200.00	$2,400.00
Total internet cost for the team	$3,000.00	$36,000.00
Total infastructure cost for the team	$3,000.00	$36,000.00
Total Indirect cost for the project	**$6,000.00**	**$72,000.00**

11.5 INTANGIBLE COST ANALYSIS

per day cost	$0.20
per-user cost for 24H trial	$0.20
expected uer cost for trial	$2,000.00

11.6 TOTAL CALCULATED COST

#	Item	Cost Yearly
1	Total Cost for resources	$14,236.56
2	Total Cost for development	$22,511.25
3	1st Year Cost : development + resources	$36,747.81
4	2nd Year to Other year Resource cost + Maintainance cost 40% of the development cost	$23,241.06
5	Total Indirect cost for the project (food + transeportation + internet + infastructure)	$72,000.00
6	Free tire cost for a user	$2,000.00
7	Grand total cost for the project 1st Year	$110,747.81
8	Grand Total cost for the project after 1st Year	$97,241.06