****

**YOUTUBE SPAM COMMENT DETECTION SYSTEM**

**By: -**

**Pranav Kolhe B228**

**Abhraneel Das B211**

**Dhairya Desai B212**

**Submitted to: -**

**Dr. Radhakrishna Rambola**

**INDEX: -**

* Project Overview
* Objectives
* Project Scope
* Milestones
* Constraints
* Deliverables
* Cost Estimation
* WBS
* Gantt Chart
* Activity Table
* PERT Chart
* Network Diagram
* Software Configuration Management
* Software Quality Management

**Project Overview:**

The YouTube spam comment detection system aims to detect and filter out spam comments from the YouTube platform using machine learning algorithms. The system will take in comments from various YouTube videos and classify them as either spam or non-spam based on their content.

The project will require a team of data scientists, machine learning engineers, and software developers. The team will work under the project manager, who will oversee the project and ensure that it is completed within the given timeline and budget.

**Objectives:**

The main objectives of this project are:

* Collect and preprocess the YouTube Spam Collection Data Set available on the UCI Machine Learning Repository.
* Evaluate the performance of different classification algorithms like SVM, Logistic Regression, Naive Bayes Classifier, and Random Forest.
* Use the trained model to predict whether a comment is spam or not for new comments.
* Develop a user-friendly interface to input comments and display the spam detection results.
* Create a detailed project report documenting the project's scope, objectives, methodology, and results.
* Conduct a project close-out meeting to discuss the project's outcomes, challenges, and recommendations for future improvements.

**Project Scope:**

The Project Scope of this project includes:

* Developing a machine learning model for identifying and filtering out spam comments on YouTube videos.
* Integrating the spam comment detection system with YouTube’s existing comment moderation tools.
* Testing and validating the accuracy of the spam comment detection system on a representative sample of YouTube videos.
* Deploying the spam comment detection system to YouTube’s production environment.

**Milestones:**

Project kickoff and requirements gathering: Week 1-3

Risk analysis and Cost Estimation: Week 4-5

Execution: Week 6-10

Quality Management: Week 11

Project completion and final report: Week 12-14

**Constraints:**

There are several constraints that can impact the implementation of the YouTube spam comment detection project, including:

* Data availability:

The availability of large amounts of high-quality data is crucial for training and evaluating machine learning models. Limited data availability can make it difficult to achieve high accuracy and reliability in detecting spam comments.

* Technical expertise:

Developing and implementing machine learning algorithms requires significant technical expertise in the fields of computer science and data analytics. The lack of qualified professionals with the necessary skills can be a major constraint.

* Computational resources:

Machine learning algorithms are computationally intensive and require powerful hardware resources such as GPUs and high-performance computing clusters. The availability and cost of these resources can be a major constraint.

* Regulatory compliance:

The use of personal data in machine learning models is subject to various regulatory requirements such as data privacy and security laws. Ensuring compliance with these regulations can be a significant constraint for organizations.

* Cost:

Developing and implementing machine learning algorithms can be expensive, requiring investment in hardware, software, and personnel. Limited budgets can be a significant constraint, particularly for small and medium-sized organizations.

**Deliverables:**

The deliverables for this topic could be:

* A well-documented dataset of YouTube comments labeled as spam or non-spam.
* A trained machine learning model capable of accurately detecting spam comments on YouTube videos.
* A report summarizing the project's findings and outlining recommendations for further research or application.
* Documentation on the preprocessing techniques, machine learning models, and evaluation metrics used in the project.
* Source code for the data preprocessing, machine learning model training, and testing process.

**Cost Estimation:**

Based on the requirements of the YouTube spam comment detection project, it could be classified as an organic project. Organic projects are relatively small, simple software projects with small teams, well-understood requirements, and a low degree of complexity. They typically involve well-understood applications where the software is not critical to the success of the overall system. In the case of the YouTube spam comment detection project, the requirements are clear and the project involves relatively simple software with a low degree of complexity. The project could likely be developed by a small team with a well-understood application. Therefore, it is reasonable to classify it as an organic project in terms of the COCOMO model.

Development Time (D) = 2.5 x (KLOC) ^ 0.38 months

Assuming a KLOC estimate of 50 KLOC, the development time for the project would be:

Estimate: 2.5 x (50) ^ 0.38 = 11.05 months

Therefore, based on the COCOMO model, the time to develop the YouTube spam comment detection system would 11.05.

Effort Required = a x (D) ^ b person-months

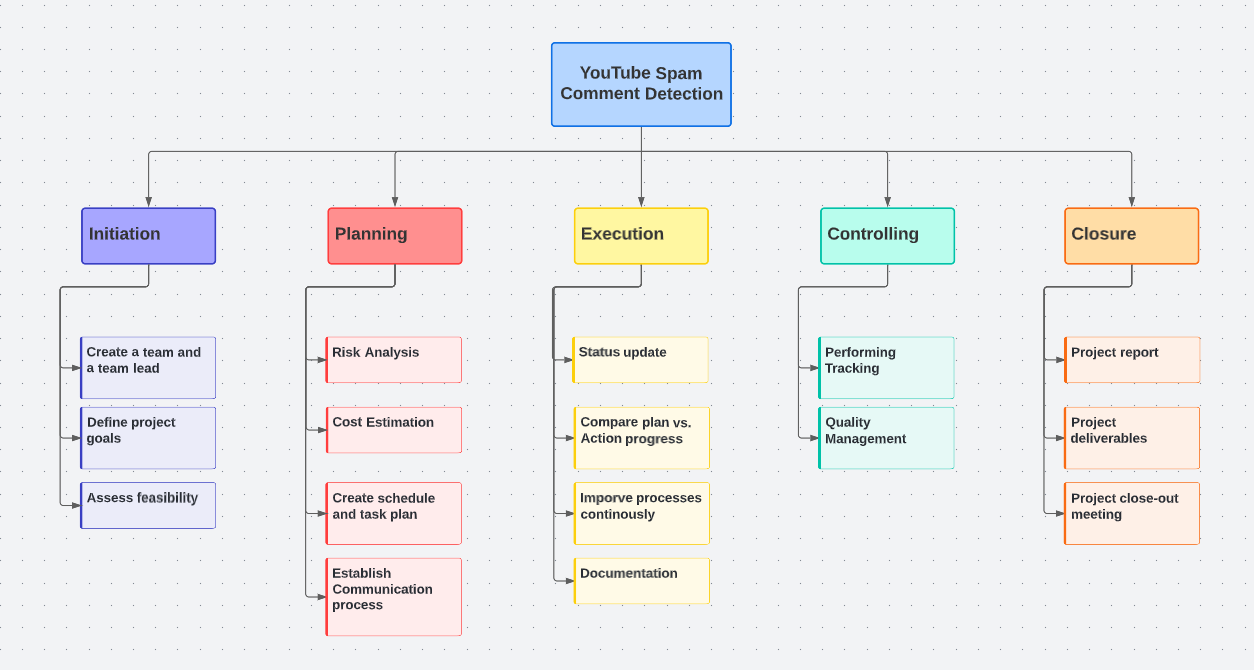
where a = 2.4 and b = 1.05 for an organic project.

Using the same KLOC estimate as above, we can calculate the effort as follows:

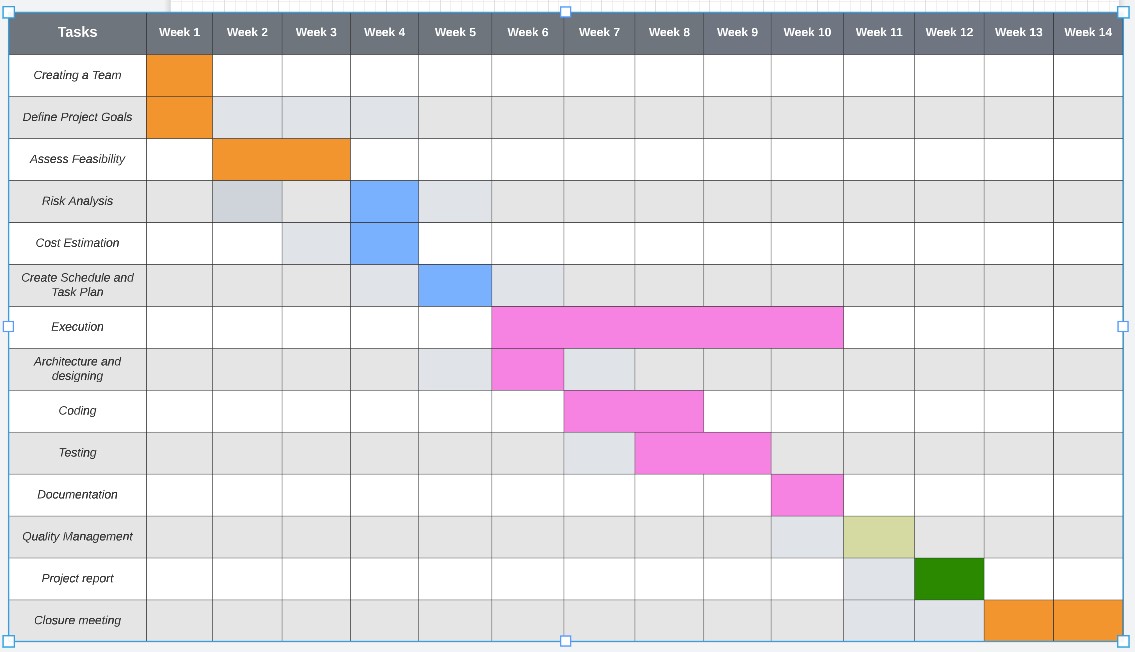
Estimate: 2.4 x (11.05) ^ 1.05 = 29.9 person-months

Therefore, based on the COCOMO model, the effort required to develop the YouTube spam comment detection system would be 29.9 person-months.

**Work Breakdown Structure:**



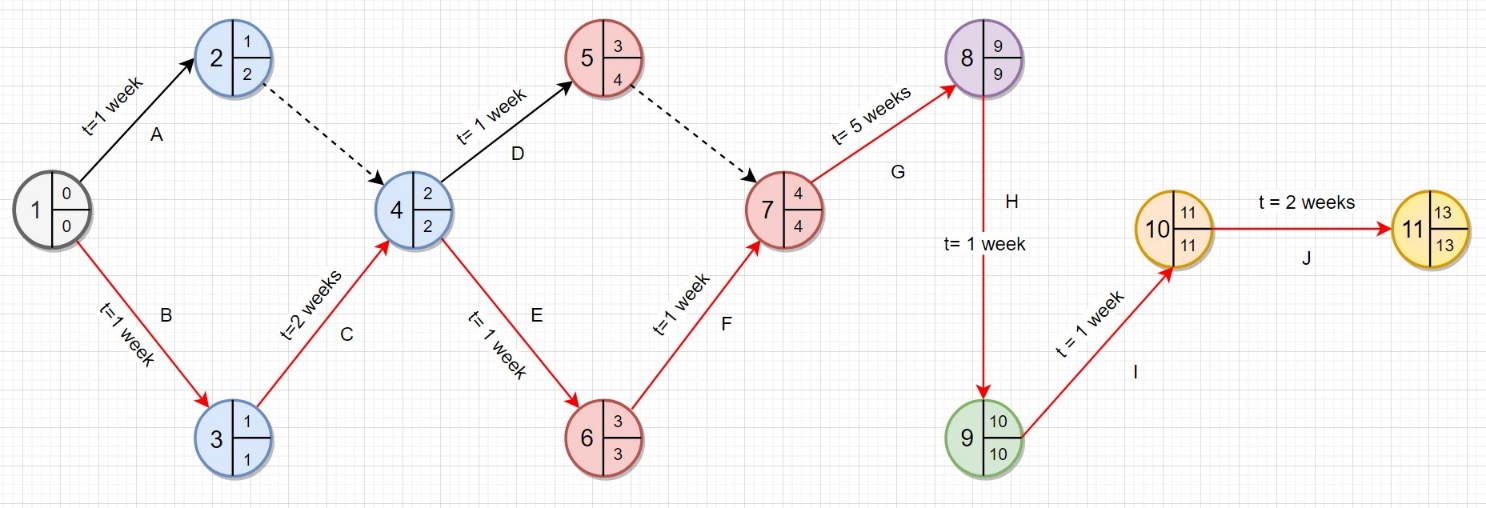
**Gantt Chart:**

****

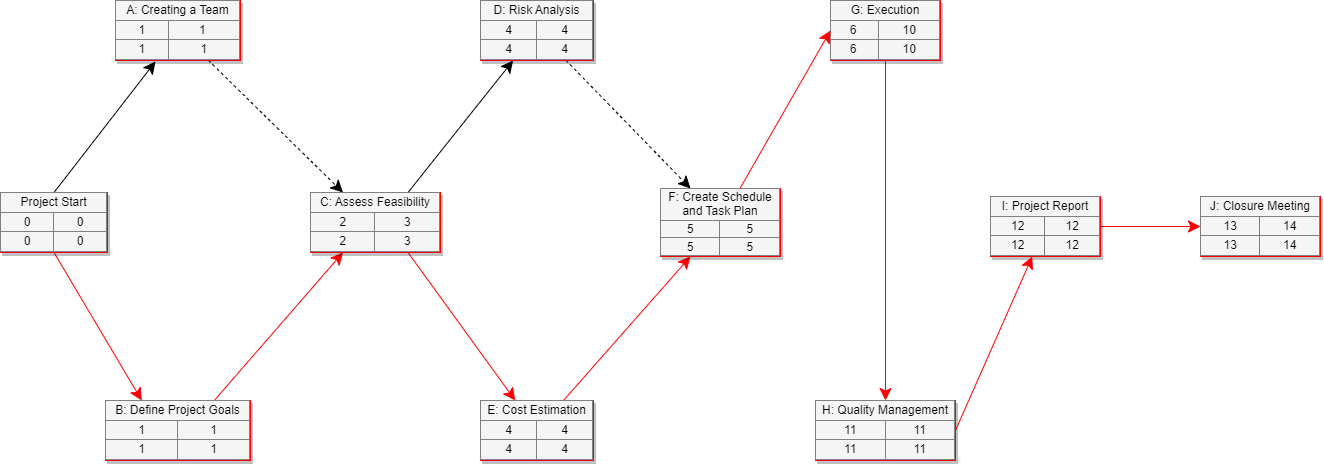
**Activity Table:**

****

**PERT Chart:**

****

**Network Diagram:**



**Software Configuration Management:**

Key activities that can be performed as part of SCM for this project:

* Version Control:

It involves tracking and managing changes to the source code and other project artifacts. Version control allows developers to keep track of changes made to the code, collaborate effectively, and roll back changes if necessary.

* Build and Release Management:

It involves automating the process of building and deploying the software to different environments. Build and release management ensures that the software is built consistently and deployed correctly.

* Configuration Management:

It involves managing the configuration of the software system, including hardware, software, and documentation. Configuration management ensures that the software system is consistent and reproducible.

* Change Management:

It involves managing changes to the software system. Change management ensures that changes are approved, tracked, and tested before being incorporated into the software system.

* Quality Assurance:

It involves ensuring that the software system meets the required quality standards. Quality assurance ensures that the software system is tested thoroughly and meets the functional and non-functional requirements.

**Software Quality Management:**

Software Quality Management would involve ensuring that the machine learning models and the code used for data preprocessing, model building, and testing are of high quality and meet the required standards. This includes establishing and adhering to coding standards, conducting thorough testing, implementing version control, tracking and managing issues, and ensuring documentation is up-to-date and accurate.

To ensure software quality, it is important to establish a set of quality objectives and metrics, such as accuracy and precision of the models, and to monitor and measure progress towards these objectives. Testing should be carried out at every stage of the development process to identify and correct defects early on. This includes unit testing, integration testing, and system testing.

A key aspect of software quality management is also the implementation of continuous improvement processes, such as using feedback from users and stakeholders to identify areas for improvement and implementing changes based on this feedback. This helps to ensure that the software meets the changing needs and requirements of users and remains of high quality over time.