Bahria University
Discovering Knowledge

Submitted by:
Natasha Altaf
01-134222-183
Qurat-ul-Ain Fatima
01-134222-121
Syed Muhammad Ali Raza Naqvi
01-134222-149

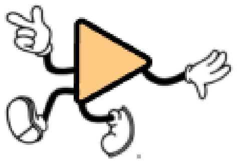
Software Engineering – Project Report “Peek-a-Tube”

Bachelor of Science in Computer Science – 5(B)
Submitted to: Ms. Sara Durrani

Department of Computer Science
Bahria University, Islamabad

Contents

Overview



This report presents **Peek-a-Tube**, a browser extension designed to enhance online safety for children aged 10–13 by filtering YouTube content. With the growing consumption of digital media, ensuring age-appropriate access has become a priority. Peek-a-Tube leverages advanced algorithms and AI-driven analysis to evaluate video metadata, ensuring unsuitable content is blocked. The project emphasizes user-friendly operation, scalability for future enhancements, and adherence to privacy standards. This document covers the project's objectives, technical design, implementation details, and evaluation, offering a comprehensive guide to its development and functionality.

Chapter 1: Introduction to Peek-a-Tube

Today, children watch a lot of videos online, especially on YouTube. However, not all videos are suitable for all ages. Our project, Peek-a-Tube, is creating a special tool that helps parents make sure their children can only watch videos that are right for their age. This tool is a browser extension, which means it works directly in the web browser, making it easy for parents to use while their kids are watching videos on YouTube.

Peek-a-Tube uses smart technology to check if videos are okay for kids aged 10 to 13. It blocks videos that are not suitable, ensuring children see only age-appropriate content. This tool is designed for parents who want to keep their children safe online and make sure they only watch videos that are good for them.

1.1 Background

Children and teenagers consume a large amount of digital content, making it challenging to ensure their online safety. YouTube, as one of the biggest video-sharing platforms, offers a variety of content, much of which may not be suitable for young viewers. Existing parental controls, including YouTube's restricted mode, often lack the precision to filter content based on specific age groups. This creates a need for advanced tools that can filter content accurately for different age ranges, such as 10–13 years. This project aims to improve children's online safety by providing a more effective and accurate content filtering solution for YouTube.

1.2 Project Scope

The project entails the development of a web extension specifically for the Google Chrome browser that filters YouTube video content based on the age category: 10-13 years. The extension will:

- Analyze video metadata to determine content appropriateness for each age group.

Limitations

- The extension will only function within the Chrome browser and is dependent on the accuracy and availability of video metadata provided by YouTube.
- It will not alter the video content itself but will block access to content deemed inappropriate.
- The effectiveness of the AI model in accurately classifying video content based on age appropriateness could vary based on the data it was trained on.

1.3 Project Objectives

1.3.1 Enhance Child Safety Online

- **Objective**: To minimize children's exposure to harmful or inappropriate content on YouTube, especially videos that may contain violence, adult themes, or misinformation.
- **Significance**: By tailoring content filtering specifically for age group (10–13), the tool ensures that children encounter content aligned with their developmental needs and emotional maturity.
- **Methodology**: Using advanced algorithms to analyze video titles, descriptions, and metadata, the system can identify, and block videos flagged as unsuitable for the specific age category.
- **Impact**: This reduces the risk of children being influenced by harmful material while fostering a safer and more constructive online environment.

1.3.2 Improve Content Filtering Accuracy

- **Objective**: To achieve high precision and recall in filtering YouTube content using sophisticated AI models.
- **Significance**: Current filtering systems often result in false positives (blocking appropriate videos) or false negatives (allowing inappropriate content). The project aims to reduce such errors, ensuring reliable and effective filtering.
- **Methodology**: The extension leverages metadata analysis and AI-driven decision-making to determine the age-appropriateness of videos. Factors like keywords, channel credibility, and audience ratings are considered.
- **Impact**: Improved accuracy builds trust among users and ensures the filtering system adapts effectively to changing trends in video content.

1.3.3 Ensure User-Friendly Experience

- **Objective**: To make the extension easy to install and operate, minimizing complexity for users.
- **Significance**: Many parents may lack technical expertise. A simple, intuitive interface ensures wider adoption and satisfaction.
- **Methodology**:
  - Streamlined installation process with clear instructions.
  - Automated content analysis in the background, reducing the need for user intervention.
- **Impact**: Ensuring ease of use increases the likelihood of parents using the tool consistently, protecting more children from unsuitable content.

1.3.4 Scalability and Adaptability

- **Objective**: To design the extension for future growth, supporting additional browsers and enhancing content filtering for more specific age bracket.
- **Significance**: As the demand for parental control tools grows, the ability to expand the extension’s functionality will meet evolving user needs.
- **Methodology**:
  - Developing the extension with modular code architecture to simplify updates and enhancements.
  - Designing for cross-browser compatibility, starting with Chrome but with potential for Firefox, Safari, and Edge support.
  - Building an adaptive AI model capable of incorporating feedback and expanding age-specific filtering rules.
- **Impact**: Scalability ensures the extension remains relevant and versatile, accommodating a broader user base and refining its filtering capabilities over time.

Chapter 2: Requirements Analysis

2.1 Functional Requirements

Table 2 - 1 Functional Requirements

S.No.	Functional Requirements	Type	Status
1	The extension should monitor and store the current YouTube URL, updating it during SPA navigation.	Core	Complete
2	It should extract the video ID and fetch metadata (title, description, tags, content rating) using the YouTube Data API.	Core	Complete
3	The extension must analyze video metadata and block age-restricted or unsuitable videos based on predefined rules.	Core	Complete
4	It should integrate with the Llama3 API to analyze metadata and make a final decision regarding video suitability.	Core	Complete
5	Users must be able to enable or disable the extension via stored preferences in chrome.storage.local.	Core	Complete
6	The extension should redirect users to the YouTube homepage if a video is flagged as unsuitable.	Core	Complete
7	Errors encountered during API calls must be logged, and fallback mechanisms should prevent complete functionality failure.	Core	Complete
8	A small pop-up should appear to inform the user whenever a video is blocked, providing an explanation for the action.	Core	Complete

2.2 Non-Functional Requirements

Table 2 - 2 Functional Requirements

S.No.	Non-Functional Requirements	Category
1	The extension should perform content analysis and provide results within a second to ensure a smooth experience.	Performance
2	Must ensure the secure handling of both video metadata and user data to comply with data privacy laws.	Security
3	The extension should only be compatible with Google Chrome, ensuring optimal performance and compatibility.	Compatibility
4	User interactions, such as logging in, enabling/disabling the extension, must be intuitive and easy to perform.	Usability
5	Error messages should be clear and guide users on how to resolve issues.	Usability
6	The system must be available 24/7 without interruptions once installed on the user's browser.	Availability
7	The design of the extension should allow easy addition of future features, such as support for other platforms or advanced moderation rules.	Extensibility

2.3 Use Case Modeling and Workflow Representation

2.3.1 Workflow Representation

Fig 2.1 represents an activity diagram of the project's workflow.

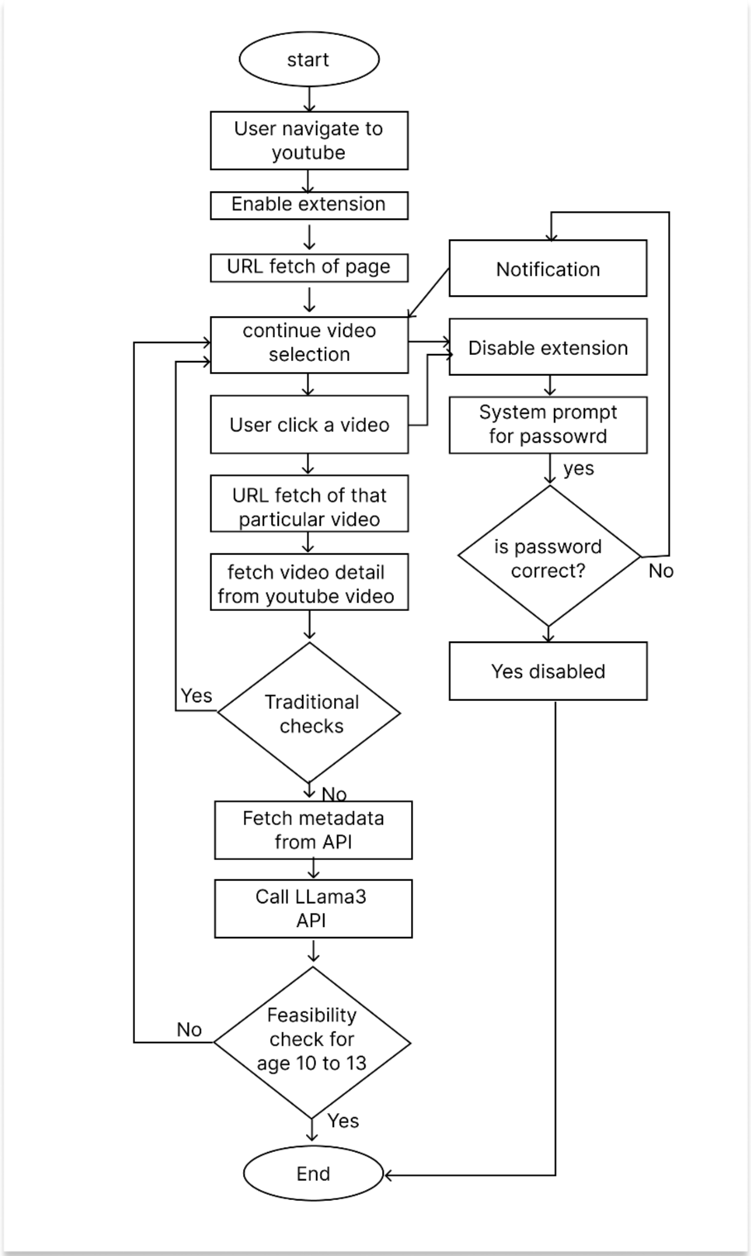


Fig 2-2 Workflow Representation

2.3.1 Use Case Modeling

Fig 2.2 represents a use case diagram of the Parent and a Child.

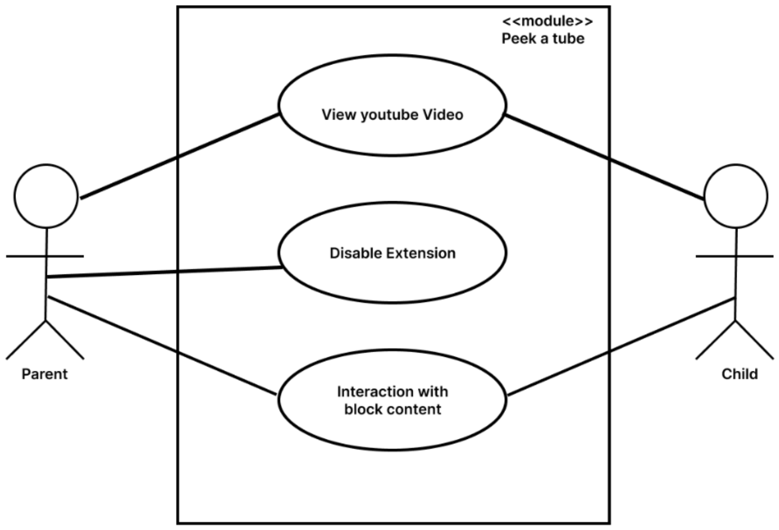


Fig 2-2 Use Case Diagram

Use Case Description Table

1. *View Youtube Video*

Table 2 - 3 Use Case Description Table for "View Youtube Video"

Attribute	Details
Use Case ID	002
Use Case Name	View YouTube Video
Actors	Child
Description	Child views YouTube videos through the filtered interface.
Trigger	Child selects a video to watch.
Pre-Conditions	Extension is active and configured for content filtering.
Post Conditions	Video plays if appropriate, or blocked message is shown.
Normal Flow	Child selects a video.
	System checks video against filters.
	Video is played or blocked based on content.

2. *Disable Extension*

Table 2 - 4 Use Case Description Table for "Disable Extension"

Attribute	Details
Use Case ID	003
Use Case Name	Disable Extension
Actors	Parent
Description	Parent disables the content filtering extension.
Trigger	Parent selects "Disable Extension."
Pre-Conditions	Parent is logged in and at the settings page.
Post Conditions	Extension is disabled, and content is no longer filtered.
Normal Flow	Parent navigates to settings.
	Parent selects "Disable Extension."
	Parent confirms the action.

	Extension is disabled.

3. *Interaction with Blocked Content/Popup*

Table 2 - 5 Use Case Description Table for "Interaction with Blocked Content/Popup"

Attribute	Details
Use Case ID	004
Use Case Name	Interaction with Blocked Content/Popup
Actors	Child
Description	Child interacts with a popup notification for blocked content.
Trigger	Child attempts to view a blocked video.
Pre-Conditions	Child is using the extension, and a video is blocked.
Post Conditions	Child reads the notification and cannot watch the video.
Normal Flow	Child selects a blocked video.
	System displays a popup explaining the block.
	Child acknowledges the popup.

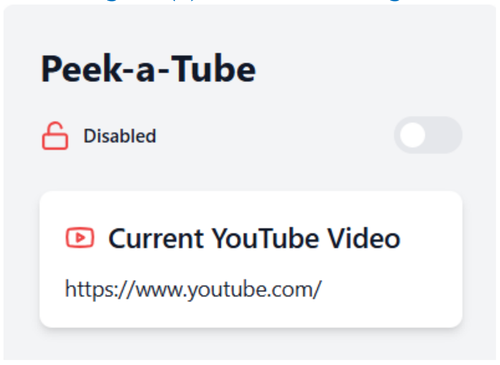
Chapter 3: Software Design Documentation

This chapter provides a detailed design documentation for *Peek-a-Tube*. It outlines the structure, components, workflows, and system interactions essential for implementing the web extension.

3.1 Front-End Interface Design

Figs 3.1 shows the Front-End Interface of "Peek-a-Tube" web extension.

Fig 3-1(a) Peek-a-Tube logo



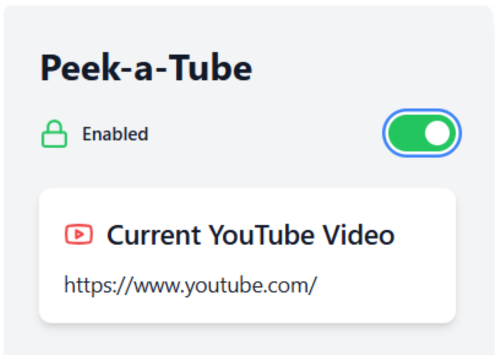
Peek-a-Tube

Disabled

Current YouTube Video

https://www.youtube.com/

Fig 3-1(b) Default Window/Extension Disabled



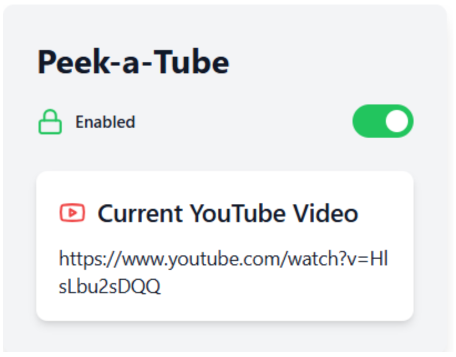
Peek-a-Tube

Enabled

Current YouTube Video

https://www.youtube.com/

Fig 3-1(c) Extension Enabled

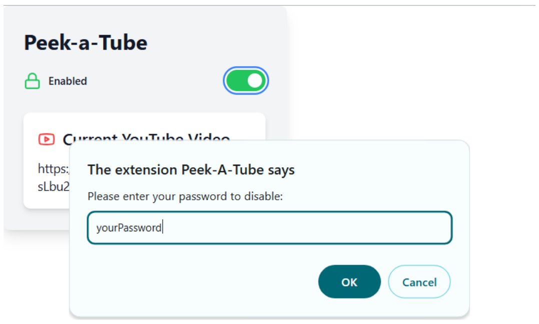

Peek-a-Tube

Enabled

Current YouTube Video

https://www.youtube.com/watch?v=HlsLbu2sDQQ

Fig 3-1(d) Real-time Youtube Video URL fetched


The extension Peek-A-Tube says

Please enter your password to disable:

yourPassword

Fig 3-1(e) Enter Password to disable the extension

3.2 Sequence Design

Fig 3.2 represents sequence diagram of “Peek-a-Tube”

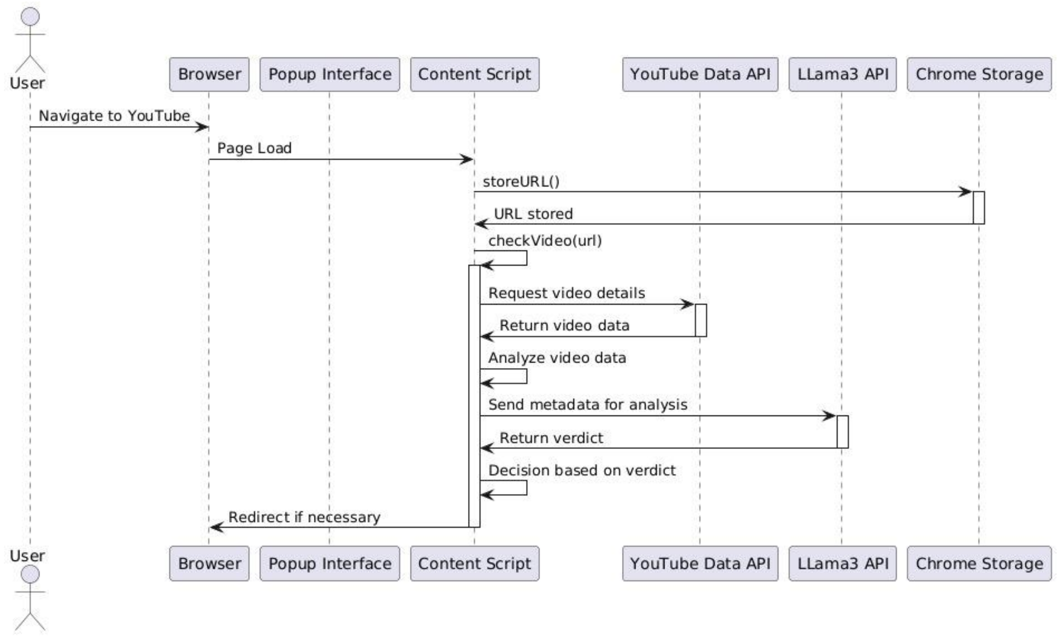


Fig 3-2 Sequence Diagram

3.3 Activity Diagram

Fig 3.3 represents an activity diagram of the project's workflow.

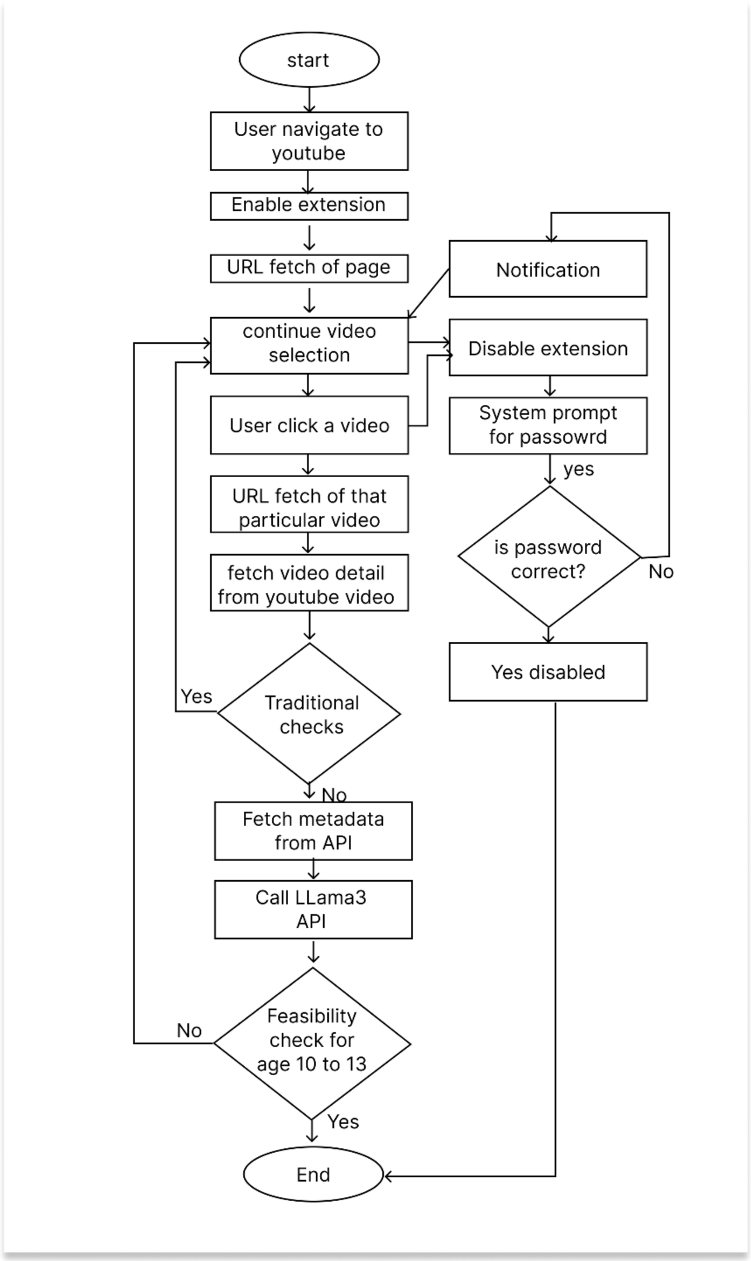


Fig 3-3 Activity Diagram

3.4 System Architecture Design

Fig 3.4 represents the system architecture of the project.

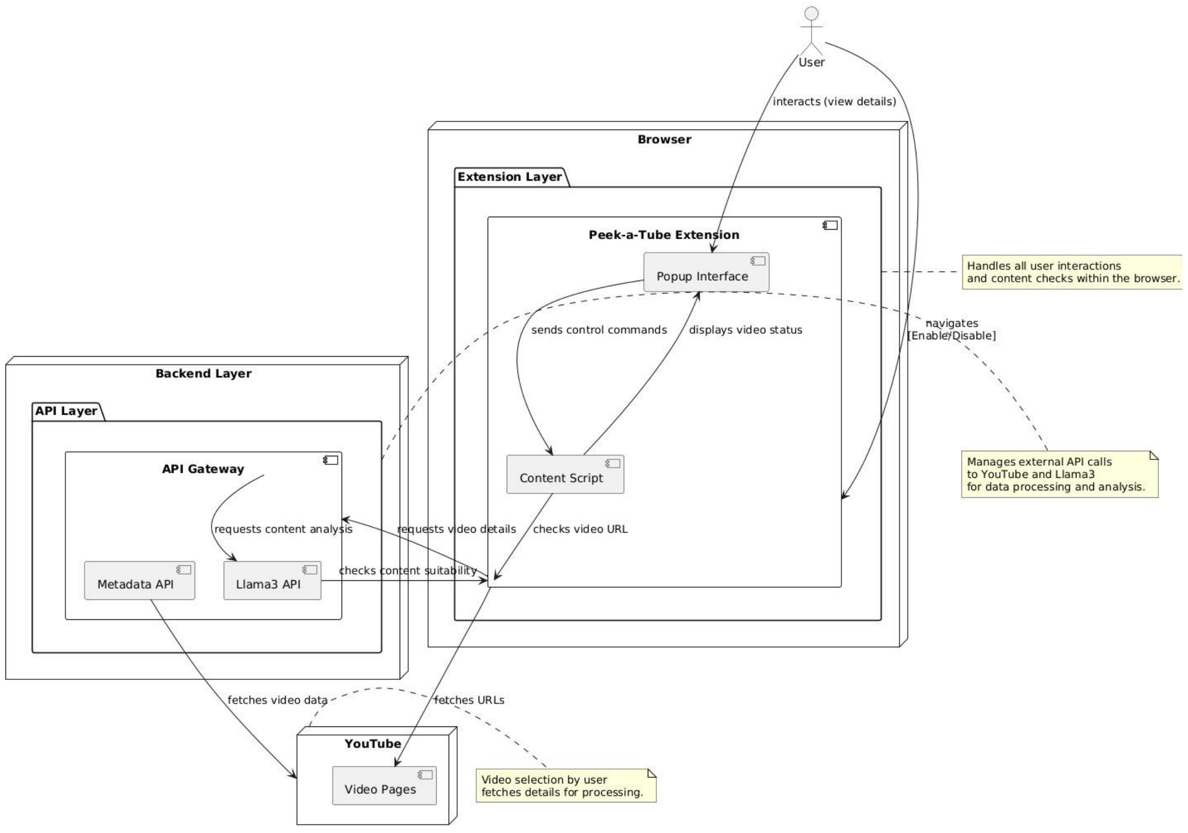


Fig 3-4 System Architecture Diagram

Chapter 4: Implementation

4.1 Objective

Develop the system according to the design specifications, ensuring all components work seamlessly to meet the project's objectives.

4.2 Deliverables

4.2.1 Source Code

- The source code is well-documented and adheres to industry standards for maintainability and scalability.
- All functionality, including URL monitoring, metadata analysis, and API integrations, is implemented as per the specifications.
- The complete source code is available on the GitHub repository: Peek-a-Tube GitHub Repository.

4.2.2 Version Control

- Git is used for version control to ensure systematic tracking of changes and collaboration among team members.
- The repository includes clear commit messages, a README file for instructions, and branches for feature development and testing.
- Repository Link: GitHub - Peek-a-Tube.

This phase ensures the foundational components of Peek-a-Tube are robustly developed and managed systematically for future enhancements**.**

Chapter 5: Testing

5.1 Objective

Verify that the system meets the specified requirements and is free of defects.

5.2 Deliverables

5.2.1 Test Plan

- As we are working on the extension, so we did unit testing only.
- A document outlining the testing strategy, types of tests to be performed, and test cases.
- Includes functional, integration, and end-to-end testing strategies.

5.2.2 Test Report

- A summary of testing activities, including test cases executed, defects found, and their resolution status.

5.3 Test Cases

Test Cases Table

1. *Test Case – 01*

Table 5 - 1 Test Case – 01

Use Case ID	01
Use Case Name	URL Monitoring
Actors	Browser, Extension
Description	This use case verifies that the extension accurately monitors and updates the current YouTube URL in real time during SPA navigation.
Trigger	User navigates to a different video on YouTube without refreshing the page.
Preconditions	The extension is installed and enabled.
Postconditions	The extension correctly detects and stores the updated URL.
Normal Flow Postconditions	1. User navigates to a new video.
Normal Flow Postconditions	2. The system detects the change and updates the URL being monitored.
	The extension correctly detects and stores the updated URL.

2. *Test Case – 02*

Table 5 - 2 Test Case – 02

Use Case ID	**02**
Use Case Name	Video Blocking
Actors	Browser, Extension
Description	This use case verifies that the extension blocks inappropriate videos based on metadata analysis.
Trigger	User attempts to access a video flagged as inappropriate.
Preconditions	The extension is active, and content filtering rules are configured.
Postconditions	The video is blocked, and the user is redirected to the YouTube homepage.
Normal Flow	User clicks on a video.
	The system prevents access and redirects to the homepage.

3. *Test Case – 03*

Table 5 - 3 Test Case – 03

Use Case ID	**03**
Use Case Name	Extension Enable/Disable
Actors	User
Description	This use case verifies that users can enable or disable the extension through preferences stored locally.
Trigger	User toggles the extension settings.
Preconditions	The extension is installed and preferences are accessible.
Postconditions	The extension is toggled on/off as per user selection.
Normal Flow	User accesses settings.
	User toggles the extension.
	The system updates preferences.

Chapter 6: Deployment and Maintenance

6.1 Objective

Deploy the Peek-a-Tube extension in a Chrome environment for testing and user interaction while preparing a comprehensive plan for long-term maintenance and updates.

6.2 Deliverables

6.2.1 Deployment Plan

6.2.1.1 Environment Setup

- Ensure the Chrome browser is installed and up to date.
- Install required developer tools and enable "Developer Mode" in Chrome extensions.

6.2.1.2 Extension Packaging

- Package the source code files (including contentScript.ts) into a .zip file.
- Ensure that the manifest.json file is correctly configured with permissions and script files.

6.2.1.3 Testing in Chrome

- Load the unpacked extension in Chrome by navigating to chrome://extensions/ and clicking on "Load Unpacked."
- Select the folder containing the extension files.
- Verify that the extension loads without errors and runs the specified scripts (e.g., URL monitoring, metadata checks, Llama3 API integration).

6.2.1.4 API Key Management

- Use placeholder API keys for testing.
- Ensure keys (YOUTUBE_API_KEY, LLAMA3_API_KEY) are securely stored and not exposed publicly during testing.

6.2.2 User Manual

6.2.2.1 Installation Instructions

- Open Chrome and navigate to chrome://extensions/.
- Enable "Developer Mode."
- Load the unpacked extension from the packaged folder.

6.2.2.2 Using the Extension

- Enable the extension from the Chrome toolbar.
- Navigate to YouTube, and the extension will start monitoring URLs and filtering content automatically.

6.2.2.3 Features

- Blocks videos flagged as age-restricted or containing banned keywords.
- Uses the Llama3 API for additional checks on metadata.
- Provides popup notifications for blocked content.

6.2.2.4 Troubleshooting

- If the extension does not load, ensure all required permissions are included in manifest.json.
- Check for errors in the Chrome Developer Tools console.
- Verify API keys and endpoints are correct and active.

6.2.3 Maintenance

6.2.3.1 Regular Updates

- Periodically review and update API keys to ensure uninterrupted service.
- Refactor the code to maintain compatibility with YouTube updates and Chrome browser versions.

6.2.3.2 Error Monitoring

- Monitor Chrome console logs for errors or warnings.
- Address user-reported issues promptly.

6.2.3.3 Feature Expansion

- Incorporate additional filtering mechanisms (e.g., AI-driven sentiment analysis).
- Add support for other browsers (e.g., Firefox, Safari) based on demand.

6.2.3.4 Security Audits

- Regularly audit the extension’s code for potential security vulnerabilities.
- Ensure sensitive information, such as API keys, is securely handled and stored.