YOUTUBE BOOKMARK EXTENSION: ENHANCING YOUR VIDEO EXPERIENCE

A Summer Internship Project Report Submitted in partial fulfillment of the requirements for the award of the degree of

#### BACHELOR OF TECHNOLOGY CSE-(CYS, DS) & AI&DS

Submitted by

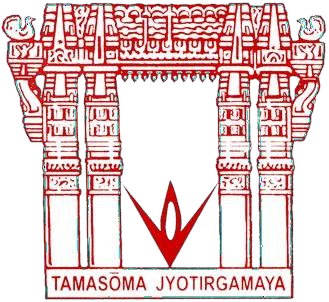
J. Abhilash 21071A7228

M. Abdul Raouf 21071A7242

D. Anil 22075A7202

Under the guidance of

#### Ms. P. DEVIKA

**Assistant Professor, Department of CSE-(CyS, DS) and AI&DS**

#### DEPARTMENT OF CSE–(CyS, DS) and AI&DS VALLURUPALLI NAGESWARA RAO VIGNANA JYOTHI INSTITUTE OF ENGINEERING AND TECHNOLOGY

An Autonomous Institute, NAAC Accredited with ‘A++’ Grade NBA Accredited for CE, EEE, ME, ECE, CSE, EIE, IT B. Tech Courses Approved by AICTE, New Delhi, Affiliated to JNTUH Recognized as “College with Potential for Excellence” by UGC ISO 9001:2015 Certified, QS I GUAGE Diamond Rated

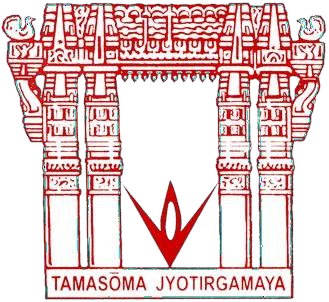
Vignana Jyothi Nagar, Pragathi Nagar, Nizampet (S.O), Hyderabad – 500 090

#### VALLURUPALLI NAGESWARA RAO VIGNANA JYOTHI INSTITUTE OF ENGINEERING AND TECHNOLOGY

An Autonomous Institute, NAAC Accredited with ‘A++’ Grade NBA Accredited for CE, EEE, ME, ECE, CSE, EIE, IT B. Tech Courses Approved by AICTE, New Delhi, Affiliated to JNTUH Recognized as “College with Potential for Excellence” by UGC ISO 9001:2015 Certified, QS I GUAGE Diamond Rated

Vignana Jyothi Nagar, Pragathi Nagar, Nizampet (S.O), Hyderabad – 500 090

#### DEPARTMENT OF CSE–(CyS, DS) and AI&DS



**CERTIFICATE**

This is to certify that the project report entitled **YOUTUBE BOOKMARK EXTENSION: ENHANCING YOUR VIDEO EXPERIENCE** is a bonafide work done under our supervision and is being submitted by J. Abhilash (21071A7228), M. Abdul Raouf (21071A7242), D. Anil (22075A7202), in partial fulfillment for the award of the degree of Bachelor of Technology in CSE-(CYS, DS) & AI&DS, of the VNRVJIET, Hyderabad during the academic year 2023-2024. Certified further that to the best of our knowledge the work presented in this project has not been submitted to any other University or Institute for the award of any Degree or Diploma.

**Ms. P. Devika**  **Dr. M. Raja Sekar**

#### Assistant Professor Professor and Head of the

#### Dept. of CSE-(CyS, DS) and AI&DS Dept. of CSE-(CyS, DS) and AI&DS

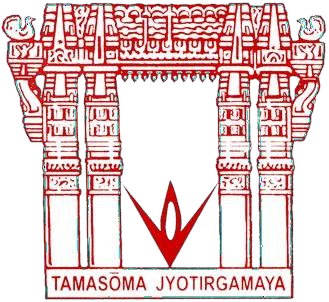
#### VNR VJIET VNR VJIET

#### VALLURUPALLI NAGESWARA RAO VIGNANA JYOTHI INSTITUTE OF ENGINEERING AND TECHNOLOGY

An Autonomous Institute, NAAC Accredited with ‘A++’ Grade NBA Accredited for CE, EEE, ME, ECE, CSE, EIE, IT B. Tech Courses Approved by AICTE, New Delhi, Affiliated to JNTUH Recognized as “College with Potential for Excellence” by UGC ISO 9001:2015 Certified, QS I GUAGE Diamond Rated

Vignana Jyothi Nagar, Pragathi Nagar, Nizampet (S.O), Hyderabad – 500 090

#### DEPARTMENT OF CSE – (CyS, DS) and AI&DS



**DECLARATION**

We declare that the Summer Internship project work entitled **YOUTUBE BOOKMARK EXTENSION: ENHANCING YOUR VIDEO EXPERIENCE** submitted in the department of Information Technology, Vallurupalli Nageswara Rao Vignana Jyothi Institute of Engineering and Technology, Hyderabad, in partial fulfillment of the requirement for the award of the degree of **Bachelor of Technology** in **Artificial Intelligence and Data Science** is a Bonafide record of our own work carried out under the supervision of Mrs. P. Devika**,** Assistant Professor**, Department of CSE-(CyS, DS) and AI&DS, VNRVJIET**. Also, we declare that the matter embodied in this thesis has not been submitted by us in full or in any part thereof for the award of any degree of any other institution or university previously.

Place: Hyderabad

|  |  |  |  |
| --- | --- | --- | --- |
| **J. Abhilash** | **M. Abdul Raouf** | **D. Anil** |  |
| (21071A7228) | (21071A7242) | (22075A7202) |  |

#### ACKNOWLEDGEMENT

Firstly, we would like to express our immense gratitude towards our institution VNR Vignana Jyothi Institute of Engineering and Technology, which created a great platform to attain profound technical skills in the field of Information Technology, thereby fulfilling our most cherished goal.

We are very much thankful to our Principal, **Dr. Challa Dhanunjaya Naidu,** and our Head of Department, **Dr. M. Raja Sekar**, for extending their cooperation in doing this project within the stipulated time.

We extend our heartfelt thanks to our guide, **Mrs. P. Devika,** and the project coordinators **Mr. Manmath Nath Das,** and **Mrs. E. Lalitha** for their enthusiastic guidance throughout the course of our project.

Last but not least, our appreciable obligation also goes to all the staff members of the Information Technology department and to our fellow classmates who directly or indirectly helped us.

J. Abhilash 21071A7228

M. Abdul Raouf 21071A7242

D. Anil 22075A7202

#### ABSTRACT

This project centers on the development of a user-friendly Google Chrome extension designed to simplify and optimize the bookmarking of YouTube videos for enhanced user convenience. The primary goal is to offer users a straightforward and efficient means of saving their favorite videos, allowing easy retrieval without the need to navigate through browsing history or the YouTube interface. The extension seamlessly integrates with the YouTube website, facilitating one-click video bookmarking directly from the YouTube player page or video listing page. Upon bookmarking a video, essential information such as the title and thumbnail is stored locally within the extension, ensuring a quick and reliable reference.

Leveraging standard web development technologies like HTML, CSS, and JavaScript, the project focuses on creating an intuitive user interface and functionality. Chrome's storage capabilities are harnessed to efficiently manage the storage and retrieval of bookmarked video data.By delivering a dedicated extension for YouTube video bookmarking, the project aims to provide users with a convenient solution for saving and accessing their preferred content. The extension's minimalist design prioritizes core bookmarking functionality, promoting a seamless and efficient user experience. Through this initiative, the project seeks to enhance the overall enjoyment of using YouTube, empowering users to effortlessly manage and revisit their curated collection of favorite videos.

**INDEX**

1. [Introduction](#_bookmark0) 10
2. [Literature Survey/ Existing System 1](#_bookmark1)1
   1. [Feasibility Study 1](#_bookmark2)1
      1. [Organizational Feasibility 1](#_bookmark3)1
      2. [Economic Feasibility 1](#_bookmark4)1
      3. [Technical Feasibility 1](#_bookmark5)1
      4. [Behavioural Feasibility 1](#_bookmark6)1
   2. [Literature Review 1](#_bookmark7)2
   3. [Existing System 1](#_bookmark8)3
   4. [Drawbacks Of the Existing System 1](#_bookmark9)3
3. [Software Requirement Analysis 1](#_bookmark10)3
   1. [Introduction 1](#_bookmark11)3
      1. [Document Purpose 1](#_bookmark12)3
      2. [Definitions 1](#_bookmark13)4
   2. [System Architecture 1](#_bookmark14)4
   3. [Functional Requirements 1](#_bookmark15)5
   4. [System Analysis 1](#_bookmark16)5
   5. [Non-Functional Requirements](#_bookmark17) 16
   6. [Software Requirement Specification](#_bookmark18) 17
   7. [Software Requirements](#_bookmark19) 17
   8. [Hardware Requirements](#_bookmark20) 17
4. [Software Design 18](#_bookmark21)
   1. [UML Diagrams](#_bookmark22) 19
      1. [Use Case Diagram 1](#_bookmark23)9
      2. [Sequence Diagram 2](#_bookmark24)3
      3. [Activity Diagram](#_bookmark25) 27
5. [Proposed System 2](#_bookmark26)9
   1. [Methodology 2](#_bookmark27)9
   2. [Functionalities](#_bookmark28) 30
   3. [Advantages Of Proposed System](#_bookmark29) 30
6. Coding/Implementation 32
   1. [Code](#_bookmark30) and Implementation 32
7. [Testing](#_bookmark31) 46
   1. [Types Of Testing](#_bookmark32) 46
      1. [Manual Testing](#_bookmark33) 46
      2. [Automated Testing](#_bookmark34) 46
   2. [Software Testing Methods](#_bookmark35) 47
      1. [Black Box Testing](#_bookmark36) 47
      2. [Gray Box Testing](#_bookmark37) 47
      3. [White Box Testing](#_bookmark38) 48
   3. [Testing Levels](#_bookmark39) 49
      1. Test Cases 49
8. Output 50
9. Conclusion and Future work 52
10. References 53

**List of Tables**

**Table Page No**

Table. 7.3.1 Test cases 47

**List of Figures**

**Figure Page No**

Fig 3.2.1 Youtube Extension 13

Fig 3.4.1 Analysis of the system 14

Fig 4.1.1.1 Use Case diagram for the application 21

Fig 4.1.2.1 Sequence diagram for the system 25

Fig 4.1.3.1 Activity Diagram for the system 27

Fig 6.1.1 Extension 31

Fig 6.1.2 manifest.json 32

Fig 6.1.3 Directory Structure 35

Fig 6.1.4 popup.js 37

Fig 6.1.5 popup.js 38

Fig 6.1.6 popup.js 38

Fig 6.1.7 background.js 38

Fig 6.1.8 Time format into seconds 38

Fig 6.1.9 content.js 39

Fig 6.1.10 content.js 40

Fig 6.1.11 content.js 43

Fig 6.1.12 popup.css 44

Fig 6.1.13 styles.css 44

Fig 7.2.1.1 Black Box Testing 46

Fig 7.2.1.2 Grey Box Testing 48

Fig 7.2.3.1 White Box Testing 48

Fig 8.1 Final Output 51

# INTRODUCTION

The project at hand focuses on the creation of a Google Chrome extension tailored to enhance the user experience of managing and revisiting favorite YouTube videos. In the ever-expanding digital landscape, where YouTube serves as a primary source of entertainment and information, the need for a streamlined and efficient bookmarking solution becomes increasingly apparent. This project addresses this need by developing a simple yet powerful extension that integrates seamlessly with the YouTube platform. The core objective of the extension is to provide users with a hassle-free method of bookmarking YouTube videos, eliminating the need to navigate through complex interfaces or sift through extensive browsing histories. By leveraging standard web development technologies like HTML, CSS, and JavaScript, the project aims to construct an intuitive user interface that complements the extension's functionality. The seamless integration of the extension with the YouTube website enables users to bookmark their preferred videos directly from the YouTube player page or video listing page with a single click. Crucially, the extension stores essential information about the bookmarked videos, such as titles and thumbnails, locally within the extension for quick and reliable retrieval. Emphasizing a minimalist design, the project aims to prioritize core bookmarking functionality, ensuring a straightforward and efficient user experience. Chrome's storage capabilities are harnessed to optimize the storage and retrieval of bookmarked video data, further enhancing the extension's performance.

# LITERATURE SURVEY/ EXISTING SYSTEM

## FEASIBILITY STUDY

A Google Chrome extension for YouTube aims to enhance the user experience by providing a streamlined and efficient solution for bookmarking favorite videos. YouTube, as a widely-used platform for video content, can become overwhelming for users to manage their preferred videos without an effective bookmarking system. This extension project seeks to address this need by creating a dedicated tool that seamlessly integrates with YouTube.

### ORGANIZATIONAL FEASIBILITY

Organizational feasibility assessment evaluates whether a proposed initiative or project aligns with the existing organizational structure, capabilities, and resources. It determines whether the organization has the necessary resources, skills, and expertise to successfully implement and sustain the proposed spam alert system.

### ECONOMIC FEASIBILITY

The economic feasibility of the proposed YouTube extension project is promising. The estimated budget for development, which encompasses resources, tools, and potential licensing fees, is considered reasonable and aligned with the project's scope. Revenue streams, including potential premium versions or affiliate marketing opportunities, present avenues for financial sustainability. The extension's impact on user satisfaction and efficiency can contribute to a positive return on investment by attracting a broad user base.

### TECHNICAL FEASIBILITY

Integration with YouTube's platform is feasible, utilizing Chrome's storage capabilities for effective data management. The straightforward technical requirements and compatibility with existing technologies make the development and deployment of the extension technically viable.

## EXISTING SYSTEM

There are some existing systems but which are very choppy and are made in the old Manifest V2 Version and don’t provide the liberty to edit the bookmarks and make it compatible with the existing webpage.

## DRAWBACKS OF THE EXISTING SYSTEM

* + **Limited Bookmarking Features**: Many existing systems lack dedicated bookmarking features within the YouTube environment, requiring users to rely on generic browser bookmarking or external tools.
  + **Inefficient Navigation**: Users often face challenges in efficiently navigating through YouTube's extensive content library to locate their bookmarked videos, leading to a less-than-optimal user experience.
  + **Lack of Integration**: Current systems may not seamlessly integrate with YouTube's interface, making the bookmarking process cumbersome and disconnected from the natural flow of user interaction on the platform.
  + **Disconnection from YouTube Interface**: Traditional browser bookmark systems lack integration with the YouTube interface, requiring users to switch between the browser and YouTube, disrupting the seamless viewing experience.
  + **Limited Contextual Information**: Generic browser bookmarks often provide minimal information about the bookmarked content, making it challenging for users to quickly identify and recall specific YouTube videos from their bookmarked list.
  + **Ineffective Organization**: Existing bookmark systems may not offer specialized organization features for YouTube content, leading to a cluttered bookmark list that is difficult for users to manage, particularly as the number of saved videos grows.

# SOFTWARE REQUIREMENT ANALYSIS

* 1. **INTRODUCTION**

This Software Requirements Specification (SRS) outlines the functional and non-functional requirements for a standalone spam alert system designed to effectively identify and classify incoming messages as either spam or not spam. The system will operate independently, analyzing provided input messages and providing an output indicating the message's classification.

### DOCUMENT PURPOSE

The purpose of this document is to define the requirements for a Youtube extension which enhances the productivity of our work by removing all the non-needed parts.

* + 1. **DEFINITIONS**

**Extension**: A software module that enhances the functionality of a web browser by adding features or modifying existing ones. In the context of the project, it refers to the additional capabilities and functionalities added to Google Chrome for improved YouTube bookmarking.

**Bookmarking**: The action of saving a webpage or, in this case, a YouTube video, for quick access later. It involves creating a reference or link to the content to facilitate easy retrieval.

**Integration**: The process of combining or coordinating different systems or components to work together seamlessly. In the project, it refers to the smooth collaboration between the extension and the YouTube platform.

**User Interface (UI):** The visual elements and interactive components through which users interact with a software application. In the project, UI design involves creating an intuitive and user-friendly interface for the extension.

## SYSTEM ARCHITECTURE

## 

## *Fig 3.2.1 Youtube Extension*

## The system architecture for the YouTube bookmark extension comprises a client-side component and a server-side backend. On the client side, a browser extension is responsible for interacting with the user interface of the YouTube website and handling bookmark functionalities. This includes capturing video URLs, managing user preferences, and displaying bookmarked videos. The extension communicates with the server-side backend, which is responsible for storing and retrieving user-specific data such as bookmarks and preferences. The backend, built on a scalable and reliable infrastructure, may use a database to store user data securely. APIs facilitate communication between the extension and the server, allowing seamless synchronization of bookmarks across devices. This modular architecture ensures efficient performance, flexibility, and easy scalability for future enhancements or additional features.

## FUNCTIONAL REQUIREMENTS

## Bookmarking Capability

## User Account Management

## Cross-Device Sync

## User Preferences

## SYSTEM ANALYSIS

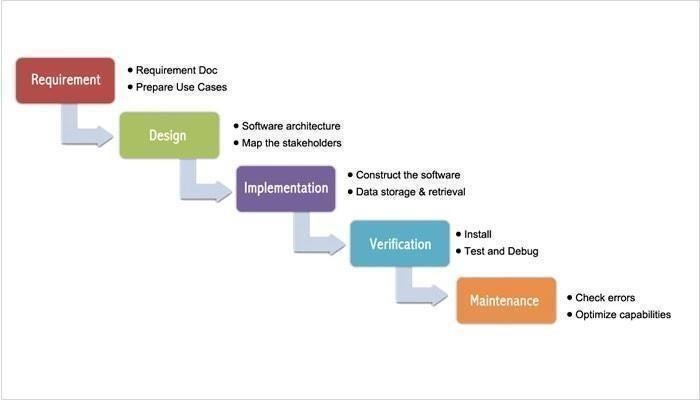


Fig:3.4.1 Analysis of the system

It was the first Process Model to be introduced. A linear sequential life cycle model is another name for it. It's quite simple to use and understand. Phases do not overlap in this paradigm, and each phase must be finished before the next one begins. The first SDLC the approach used during software development was the waterfall model.

The model shows that the development of software is linear and is a sequential process. Only after one phase of the development is completed, we can go to the next phase. In this waterfall paradigm, the phases do not overlap.

The steps in the waterfall model are explained below.

**Requirements:** The search has become more intense and concentrated on the software's requirements at this time. To comprehend the nature of the programs to be developed, the software engineer must first comprehend the software's information domain, which includes the required functionalities, user interface, and so on. The customer must be informed about the second activity, which must be recorded and presented.

**Design:** This step is used to transform the above criteria as a representation in the form of "blueprint" software before coding begins. The design must be able to meet the criteria laid out in the previous stage.

**Implementation**: The design was converted into a machine-readable format for it to be interpreted by a computer in some circumstances, i.e., through the coding process into a programming language. This was the stage in which the programmer will put the technical design phase into action.

**Verification:** It, like anything else constructed, must first be put to the test. The same may be said for software. To ensure that the application is error-free, all functions must be checked, and the results must closely comply to the previously specified requirements. **Maintenance:** Software maintenance, including development, is essential since the software that is being generated is not always exactly like that. It may still have minor faults that were not identified previously when it runs, or it may require additional capabilities that were not previously available in the software.

**Useful factors:** The Waterfall model has its advantages like it is simple to use. Additionally, while using the model all the system requirements can be defined as a whole, explicitly and at the start the product can run without many issues.

It is economic to make changes in the early stages of the project when there are problems with system requirements then when the problems which arise in later stages.

## NON-FUNCTIONAL REQUIREMENTS

#### Real time

* The system should provide the analysis result so accurately when it is unnoticeable time for its users.

#### Accuracy

* Should not be confused and the system should predict approximately.

#### Environment

* The system should provide real time recognition with high accuracy in low light conditions as well.

#### Usability

* The system should provide natural interaction to its users.

## SOFTWARE REQUIREMENT SPECIFICATIONS

**3.6.1 Manifest V3 Compatibility:**

**Description**: The extension must adhere to Google Chrome's Manifest V3 specifications, the latest version of the extension manifest format. This includes transitioning from Manifest V2 to V3 for compatibility with the latest Chrome browser updates.

**Acceptance Criteria**: The extension's manifest file should be updated to comply with the specifications outlined in Manifest V3. All deprecated features of Manifest V2 should be replaced with the corresponding features of Manifest V3 to ensure proper functionality.

**3.6.2 Content Security Policy (CSP) Implementation:**

**Description**: The extension must implement a secure Content Security Policy (CSP) to mitigate the risk of code injection and enhance overall security. The CSP should define trusted sources for scripts, styles, and other resources.

**Acceptance Criteria**: The extension's manifest and content scripts should include a well-defined CSP, restricting the allowed sources for scripts, styles, and other resources. This implementation should be tested to ensure it does not interfere with the extension's functionality.

**3.4.3 Chrome Storage Utilization:**

**Description**: The extension should efficiently utilize Chrome Storage for storing bookmarked video data locally. This includes specifying the storage permissions in the manifest file and implementing proper data handling procedures.

**Acceptance Criteria**: The extension should successfully store and retrieve bookmarked video data using Chrome Storage. The implementation should be tested to ensure data consistency and security.

## SOFTWARE REQUIREMENTS

* Browser : Google, Brave, etc..
* Version : Google Chrome Extension Manifest V3
* Technology : Web Technology, Storage API, IDE

## HARDWARE REQUIREMENTS

* Any Laptop
* Storage Space
* Processor and RAM

# SOFTWARE DESIGN

## 4.1 UML DIAGRAMS

The Device Architecture Manual describes the application requirements, operating state, application and subsystem functionality, documents and repository setup, input locations, yield types, human-machine interfaces, management reasoning, and external interfaces. The Unified Modeling Language (UML) assists software developers in expressing an analysis model through documents that contain a plethora of syntactic and semantic instructions. A UML context is defined as five distinct viewpoints that present the system from a particularly different point of view.

The components are similar to modules that can be combined in a variety of ways to create a complete UML diagram. As a result, comprehension of the various diagrams is essential for utilizing the knowledge in real-world systems. The best method to understand any complex system is to draw diagrams or images of it. These designs have a bigger influence on our understanding. Looking around, we can see that info-graphics are not a new concept, but they are frequently utilized in a variety of businesses in various ways.

#### User Model View

The perspective refers to the system from the clients' point of view. The exam's depiction depicts a situation of utilization from the perspective of end-clients. The user view provides a window into the system from the perspective of the user, with the system's operation defined in light of the user and what the user wants from it.

#### Structural model view

This layout represents the details and functionality of the device. This software design maps out the static structures. This view includes activity diagrams, sequence diagrams and state machine diagrams

#### Behavioral Model View

It refers to the social dynamics as framework components, delineating the assortment cooperation between various auxiliary components depicted in the client model and basic model view. UML Behavioral Diagrams illustrate time-dependent aspects of a system and communicate the system's dynamics and how they interact. Behavioral diagrams include interaction diagrams, use case diagrams, activity diagrams and state–chart diagrams.

#### Implementation Model View

The essential and actions as frame pieces are discussed in this when they are to be

manufactured. This is also referred to as the implementation view. It uses the UML Component diagram to describe system components. One of the UML diagrams used to illustrate the development view is the Package diagram.

#### Environmental Model View

The systemic and functional component of the world where the program is to be introduced was expressed within this. The diagram in the environmental view explains the software model's after-deployment behavior. This diagram typically explains user interactions and the effects of software on the system. The following diagrams are included in the environmental model: Diagram of deployment.

The UML model is made up of two separate domains:

* Demonstration of UML Analysis, with a focus on the client model and auxiliary model perspectives on the framework.
* UML configuration presenting, which focuses on demonstrations, usage, and natural model perspectives.

### USE CASE DIAGRAM

The objective of a use case diagram is to portray the dynamic nature of a system. However, because the aim of the other four pictures is the same, this description is too broad to characterize the purpose. We'll look into a specific purpose that distinguishes it from the other four diagrams.

The needs of a system, including various factors, are collected using use case diagrams. Most of these specifications are design specifications. As a result, use cases are constructed and actors are identified when examining a system to gather its functions.

Use case diagrams are made to represent the outside view once the primary task is completed.

In conclusion, use case diagrams are useful for the following purposes:

* + - * Used to collect a system's requirements.
      * Used to get a bird's-eye view of a system.
      * Determine various factors that are influencing the system.
      * Display the interaction of the requirements as actors.

Use case diagrams are used to analyze a system's high-level requirements. The functionality of a system is recorded in use cases when the requirements are examined. Use cases can be defined as "system functionalities written in a logical order." The actors are the second pillar of use cases that is important. Any entities that interact with the system are referred to as actors.

Internal applications, human users and external applications can all be actors. The

following factors should be kept in mind when constructing a use case diagram.

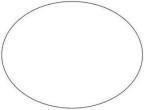
* + - * As a use case, functionalities will be represented.
      * Actors.
      * Relationships between use cases and actors.

#### Use Cases

A use case is a written depiction of how visitors will execute tasks on your website. From the standpoint of a user, it defines how a system responds to a request. Each use case is characterized by a sequence of basic actions that start with the user's goal and finish when that goal is achieved.

#### Graphical Representation

Use cases are represented by an oval shape.



The following is a more precise analysis of a use case:

* + - * A pattern of behavior displayed by the system.
      * A series of related transactions performed by an actor as well as the system.
      * Delivering something useful to the actor.

You can utilize use cases to document system requirements, connect with top users and domain experts, and test the system. Looking at the actors and defining what they can accomplish with the system is the greatest way to uncover use cases.

#### Flow of events

A sequence of times can be thought of as a collection of interactions (or opportunities) carried out by the system. They provide daily point-by-point details, published in terms of what the framework can do rather than whether the framework performs the task.

* + - * When and how the employment case begins and ends.
      * Interactions between the use case and the actor.
      * Information required by the employment case.
      * The employment case's normal sequence of events.
      * A different or exceptional flow.

#### Construction of Use case

The behavior of the framework is graphically illustrated in use-case outlines. These graphs show how the framework is utilized at a high level, when seen through the perspective of an untouchable (actor). A utilization case graph can depict all or some of a framework's work instances.

A use-case diagram may include the following elements:

* + - * Actors.
      * Use cases.

#### Relationships in use cases

Active relationships, also known as behavioral relationships, are a type of interaction that is frequently shown in use case diagrams. The four main types of behavioral relationships are inclusion, communication, generalization and extension.

#### Communicates

The behavioral relationship communicates connects an actor to a use case. Remember that the purpose of the use case is to provide some sort of benefit to the system's actor. As a result, it is critical to document these interactions between actors and use cases. A line with no arrowheads connects an actor to a use case.

#### Includes

The includes relationship (also known as the uses relationship) describes a situation in which a use case contains behavior that is shared by multiple use cases. To put it another way, the common use case is included in the other use cases. The included relationship is indicated by a dotted arrow pointing to the common use case.

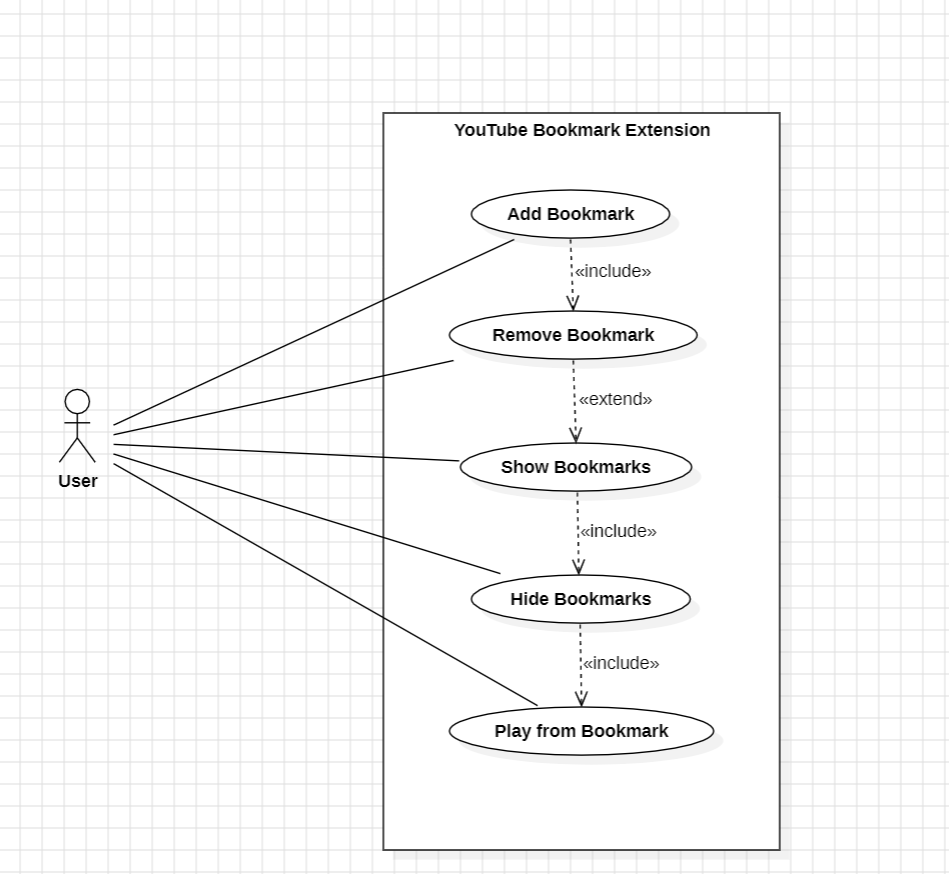
#### Extends

The extended connection describes when one use case contains behavior that allows a new use case to handle a variant or exception to the basic use case. A distinct use case handles exceptions to the basic use case. The arrow connects the basic and extended use cases.

#### Generalizes

The generalized relationship indicates that one thing is more prevalent than another. This link could be between two actors or between two use cases. The arrow points to a "thing" in UML that is more general than another "thing."

In this system Use Case diagram:



*Figure 4.1.1.1 Use Case Diagram for Youtube Extension*

#### Actors:

* + User

#### Use Cases:

* + Add Bookmark
  + Remove Bookmark
  + Show, Hide Bookmark
  + Play from Bookmark

#### Connections:

#### User can Add a bookmark to any youtube video he chooses

#### After adding the bookmark he can remove it

#### He can click on show to let him show how many bookmarks are there

### SEQUENCE DIAGRAM

Because it illustrates how a group of items interact with one another, a sequence diagram is a form of interaction diagram. These diagrams are used by software engineers and businesspeople to comprehend the requirements for a new system or to document a current process. Sequence diagrams are sometimes known as event diagrams or event scenarios. Sequence diagrams can be useful as a reference for businesses and other organizations. Make the diagram to show:

* Describe the specifics of a UML use case.
* Create a model of the logic of a complex procedure, function, or operation.
* Examine how objects and components interact with one another in order to complete a process.
* Plan and comprehend the specific functionality of a current or future scenario. The following scenarios lend themselves well to the use of a sequence diagram:

A usage scenario is a diagram that shows how your technology might be utilized in the future. It's an excellent approach to make sure you've thought through every possible system usage situation.

#### Method logic:

A UML sequence diagram can be used to study the logic of any function, method, or complex process, just as it can be used to examine the rationale of a use case. If you view a service to be a high-level method utilized by several customers, a sequence diagram is a fantastic approach to map out service logic.

#### Object:

An object has a state, a lead, and a personality. The structure and direction of objects that are, for all intents and purposes, indistinguishable are depicted in their fundamental class. Each object in a diagram represents a specific instance of a class. An order case is an object that is not named.

#### Message:

A message is the exchange of information between two articles that causes an event to occur. A message transmits information from the source point of control convergence to the objective point of control convergence.

#### Link:

An existing association between two objects, including class, implying that there is an association between their opposing classes. If an object associates with itself, use the image's hover adjustment.

#### Lifeline:

It reflects the passage of time as it goes downward. The events that occur consecutively to an object during the monitored process are depicted by this dashed vertical line. A designated rectangle shape or an actor symbol could be the starting point for a lifeline.

#### Actor:

Entities that interact with the system or are external to it are shown.

#### Synchronous message:

This is represented by a solid line with a solid arrowhead. This symbol is used when a sender must wait for a response to queries before proceeding. Both the call and the response should be depicted in the diagram.

#### Asynchronous message:

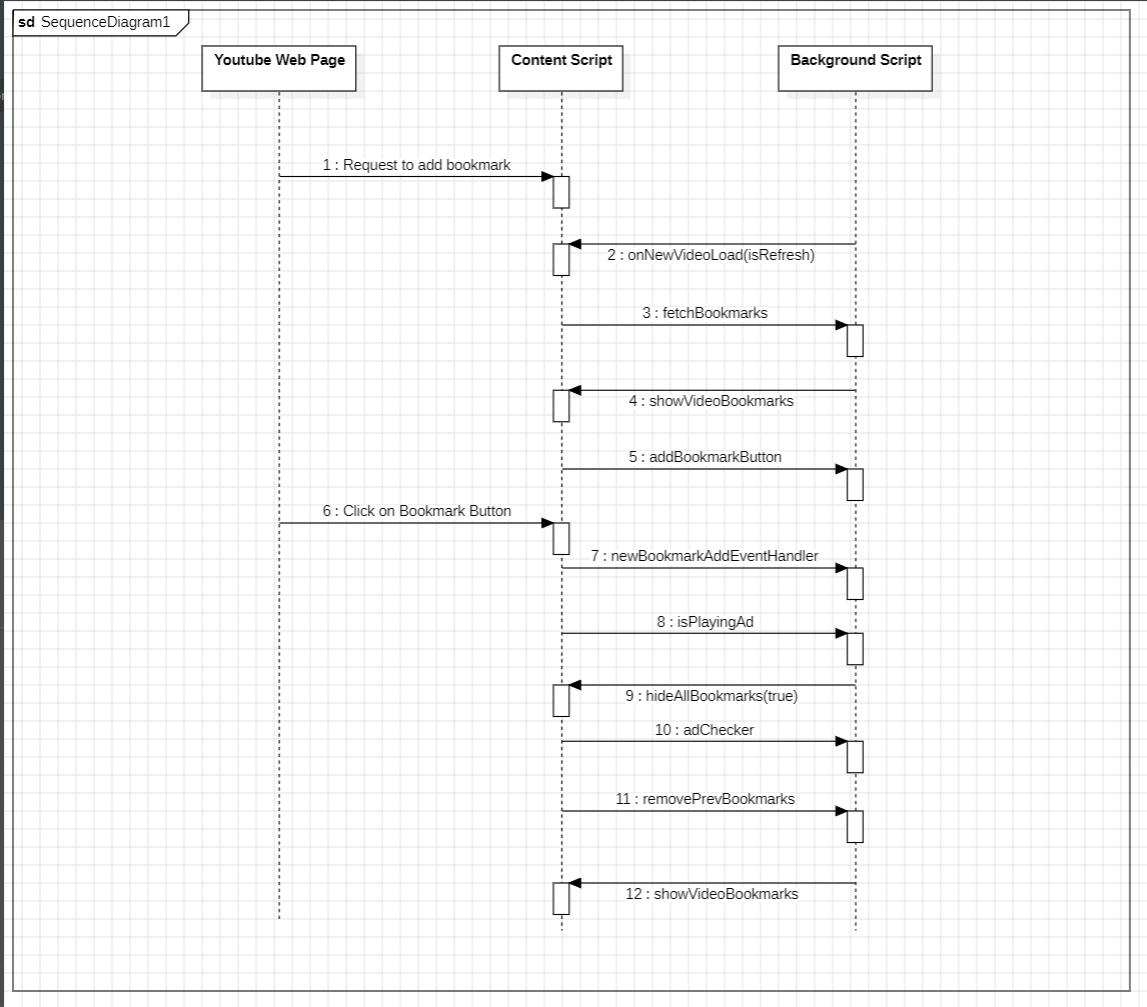
A solid line with a lined arrowhead is used to represent this. Asynchronous messages do not necessitate a response before the sender can proceed. The diagram should only include the call.

#### Delete message:

An X follows a solid line with a solid arrowhead. This message has the effect of causing an object to be destroyed.

**Sequence diagram components:**

|  |  |  |
| --- | --- | --- |
| **Name** | **Description** | **Symbol** |
| Object symbol | Represents a class or object in UML. The object symbol demonstrates how an object will behave in the context of the system. Class attributes should not be listed in this shape. | UML-Object-Diagram |
| The activation box | Represents the time needed for an object to complete a task. The longer the task will take, the longer the activation box becomes. |  |
| Actor symbol | Shows entities that interact with or are external to the system. |  |
| Lifeline symbol | Represents the passage of time as it extends downward. This dashed vertical line shows the sequential events that occur to an object during the charted process. Lifelines may begin with a labelled rectangle shape or an actor symbol. |  |
| Alternative symbol | Symbolises a choice (that is usually mutually exclusive) between two or more message sequences. To represent alternatives, use the labelled rectangle shape with a dashed line inside. |  |
| Message  symbol | This symbol is used when a sender needs to send a message. |  |
| Reply message symbol | Represented by a dashed line with a lined arrowhead, these messages are replies to calls. |  |



*Fig 4.1.2.1 Sequence Diagram for the Youtube Extension*

### ACTIVITY DIAGRAM

An activity diagram is a flowchart that displays the movement of information from one action to the next. A system operation can be used to describe the activity.

From one operation to the next, the control flow is guided. In nature, this flow might be sequential, branching, or concurrent. By employing numerous parts such as join, fork and so on, activity diagrams cope with all sorts of flow control.

Activity diagrams provide the same basic functions as the other four diagrams. It captures the dynamic behavior of the system. The other four diagrams depict message flow from one item to the next, whereas the activity diagram depict. A specific system operation is referred to as an activity. It doesn't show any communication flow from one activity to the next. The phrases activity diagrams and flowcharts are often used interchangeably. Although the diagrams resemble flowcharts, they are not.

#### Notations

**Initial point or start point**

A small, filled circle, followed by an arrow, represents the beginning action state or starting point for any activity diagram. Make sure the start point of an activity diagram with swimlanes is in the top left corner of the first column.

#### Activity or Action state

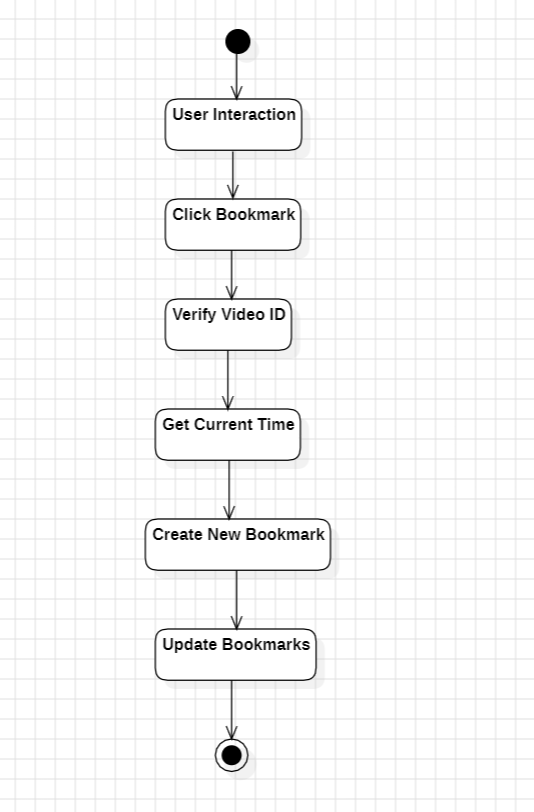
An action state is a representation of an object's non-interruptible action. You can make an action state in SmartDraw by sketching a rectangle with rounded corners.

#### Action flow

Transitions from one action state to another are depicted by action flows, also known as edges and routes. An arrowed line is commonly used to depict them.

#### Decisions and branching

A diamond signifies a multiple-choice decision. Place a diamond between the two activities when one requires a decision before moving on to the next. A condition or guard expression should be used to label the outgoing alternates. One of the paths can also be labeled "else."



*Fig 4.1.3.1: Activity Diagram for the Youtube Bookmark*

# PROPOSED SYSTEM

## METHODOLOGY

**Google Chrome Browser:**

Version 80 and above for compatibility with the latest extension features.

**HTML, CSS, and JavaScript:**

Development languages for creating the user interface and functionality of the extension.

**Google Chrome Extension Manifest V3:**

Adherence to the specifications outlined in Manifest V3 for proper compatibility.

**Chrome Storage API:**

Utilization for efficient storage and retrieval of bookmarked video data.

**Content Security Policy (CSP):**

Implementation to enhance security and mitigate code injection risks.

**Integrated Development Environment (IDE):**

Such as Visual Studio Code or any preferred IDE for coding and debugging.

## FUNCTIONALITIES

## The YouTube bookmark extension offers a range of user-friendly functionalities to enhance the video-watching experience. Users can effortlessly bookmark their favorite YouTube videos directly from the browser extension, capturing crucial details like video URL, title, and thumbnail for quick reference. The extension provides a user account management system, allowing users to create accounts, log in, and log out, enabling personalized features such as cross-device synchronization of bookmarks. With customizable user preferences, individuals can tailor their experience, setting default bookmark folders, managing notification preferences, and adjusting display options. The extension's search and filter functionalities empower users to efficiently navigate through their bookmarked content, offering options to search by keywords, filter by date, and organize bookmarks based on various criteria. Additionally, a notification system keeps users informed about new features and updates, ensuring they stay connected with the extension's evolving capabilities. Overall, these functionalities collectively contribute to a seamless and personalized YouTube bookmarking experience.

## ADVANTAGES OF PROPOSED SYSTEM:

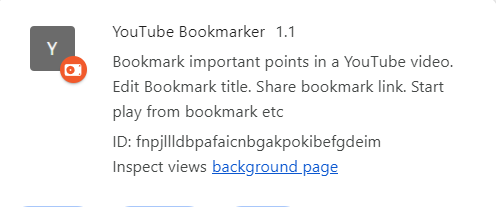
* Users can conveniently bookmark YouTube videos with a single click directly from the YouTube player or video listing page, streamlining the process of saving favorite content.
* The extension seamlessly integrates with the YouTube platform, ensuring a cohesive user experience without the need to navigate away from the familiar YouTube interface.
* Users can easily access their bookmarked videos directly from the extension, eliminating the need to search through browsing history or external bookmarking systems.
* The extension prioritizes a minimalist design, focusing on essential bookmarking functionality to provide an intuitive and clutter-free user interface.
* Bookmarking includes storing basic information like video titles and thumbnails locally, offering users a visual reference for quickly identifying and retrieving bookmarked content.
* Unlike generic browser bookmarks, the extension is tailored specifically for YouTube, offering users a dedicated tool for efficiently managing and revisiting their preferred videos.
* The proposed system contributes to an overall improved user experience on YouTube by addressing common challenges related to video bookmarking and content management.
* Users save time by avoiding cumbersome navigation through YouTube's vast content library, allowing for quick access to their curated collection of favorite videos.
* The extension can be designed to adapt to changes in YouTube's interface, ensuring continued functionality and a positive user experience despite potential updates to the platform.
* The extension provides opportunities for monetization, such as offering premium features or integrating affiliate marketing, contributing to the financial sustainability of the project.

# CODING AND IMPLEMENTATIONS

#### CODE AND IMPLEMENTATION:

To test and debug our extension during the development phase , we will follow the below steps to get a basic extension up and running.

1. Create the manifest.json file and the project structure as shown above or download them from the link at the top of the article.
2. Go to chrome://extensions from your browser. <ss here>
3. Toggle the ‘Developer Mode’ to ON.
4. Click on the ‘Load Unpacked’ option to load the Extension folder for test, debug and further development.
5. At this point, you should see your Extension on the right side of the address bar of the chrome.



*Fig 6.1.1 Extension*

Now that we have loaded our Extension folder on chrome, we are ready to build a basic chrome extension and test it out.

1**.manifest.json** – To hold information about the Chrome extension.

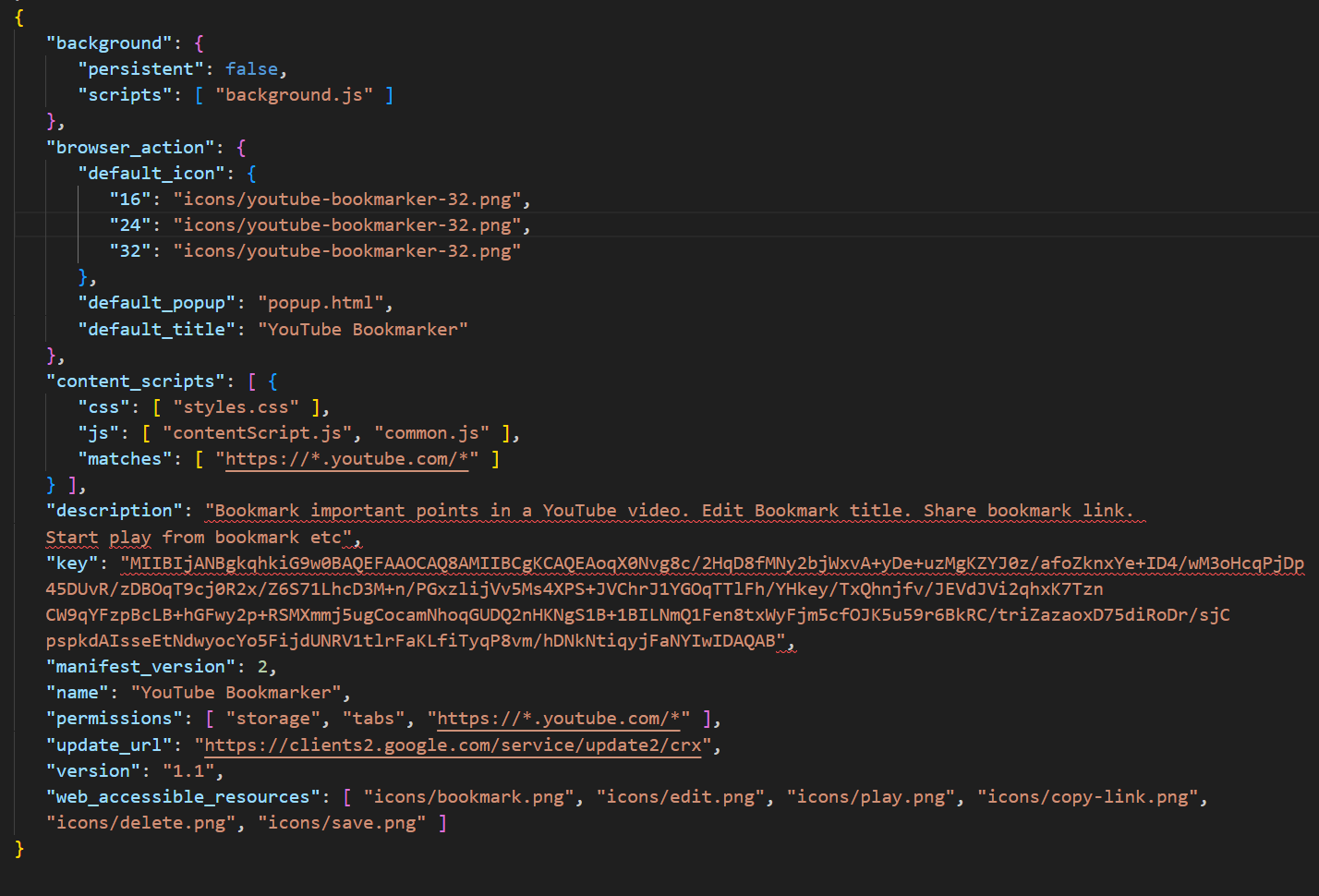
2.**popup.html** – To say, when the user clicks on the  extension icon.

3.**popup.js** – No significant work of this file at this point. Although, we’ll keep it here.

4.**background.js** – To load up the extension with the basic backend functionality. Not required for the Extension.

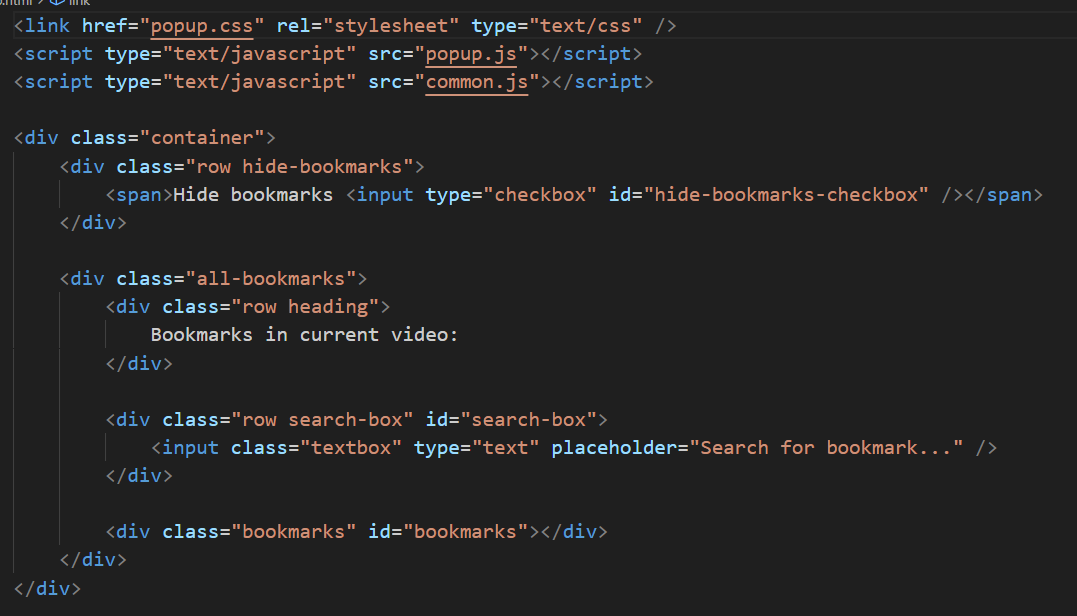
5.**jQuery.js** – We’ll include jQuery to help with development.

**1.manifest.json**:



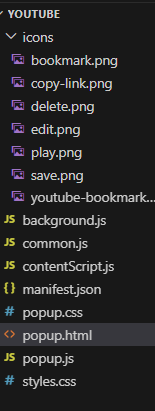
*Fig 6.1.2 manifest.json*

**2. popup.html:**



*Fig 6.1.2 popup.html*

**Project Directory structure:**



*Fig 6.1.3 Directory Structure*

**Source Code:**

1. As the extension would access and store the web addresses, i.e., youtube video addresses, it becomes imperative to have a look at the different kinds of timestamps that we can access from the DOM.

**popup.js:**





*Fig 6.1.4 popup.js*





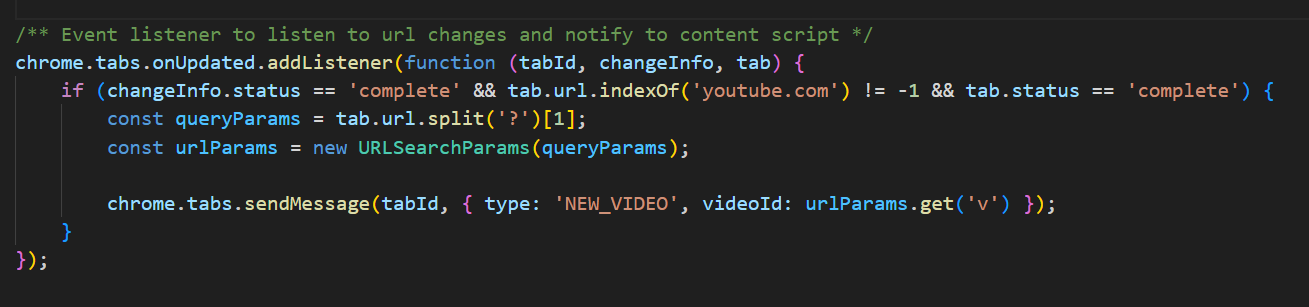
*Fig 6.1.5 popup.js*





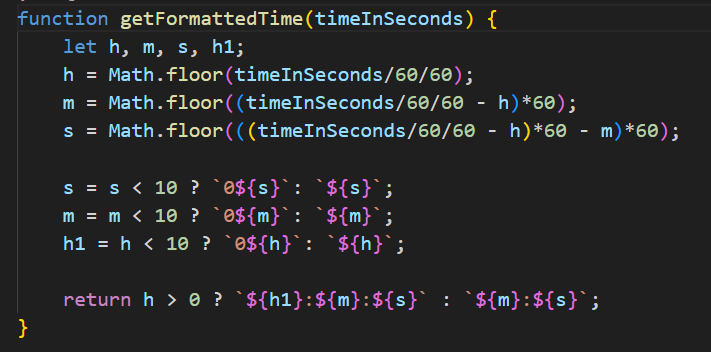
*Fig 6.1.6 popup.js*

**Background.js**



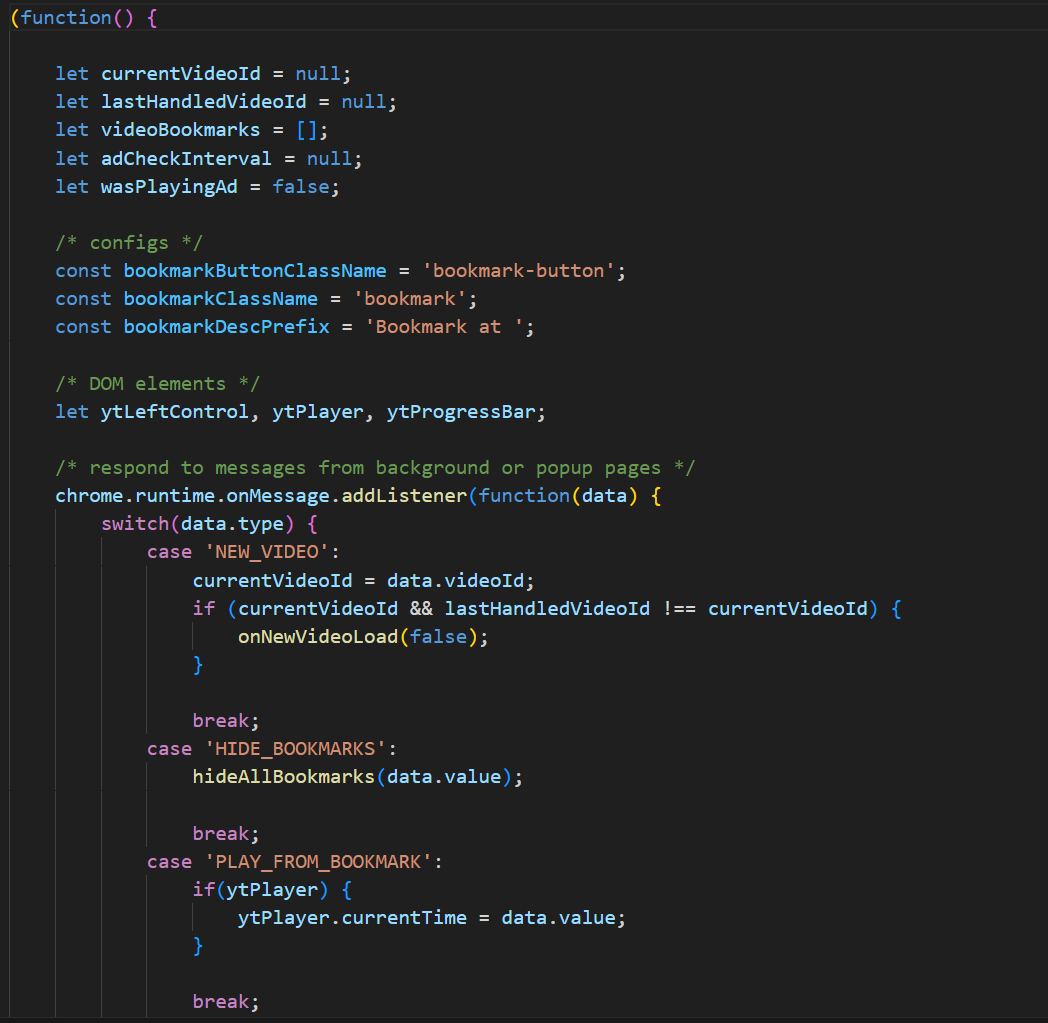
*Fig 6.1.7 background.js*

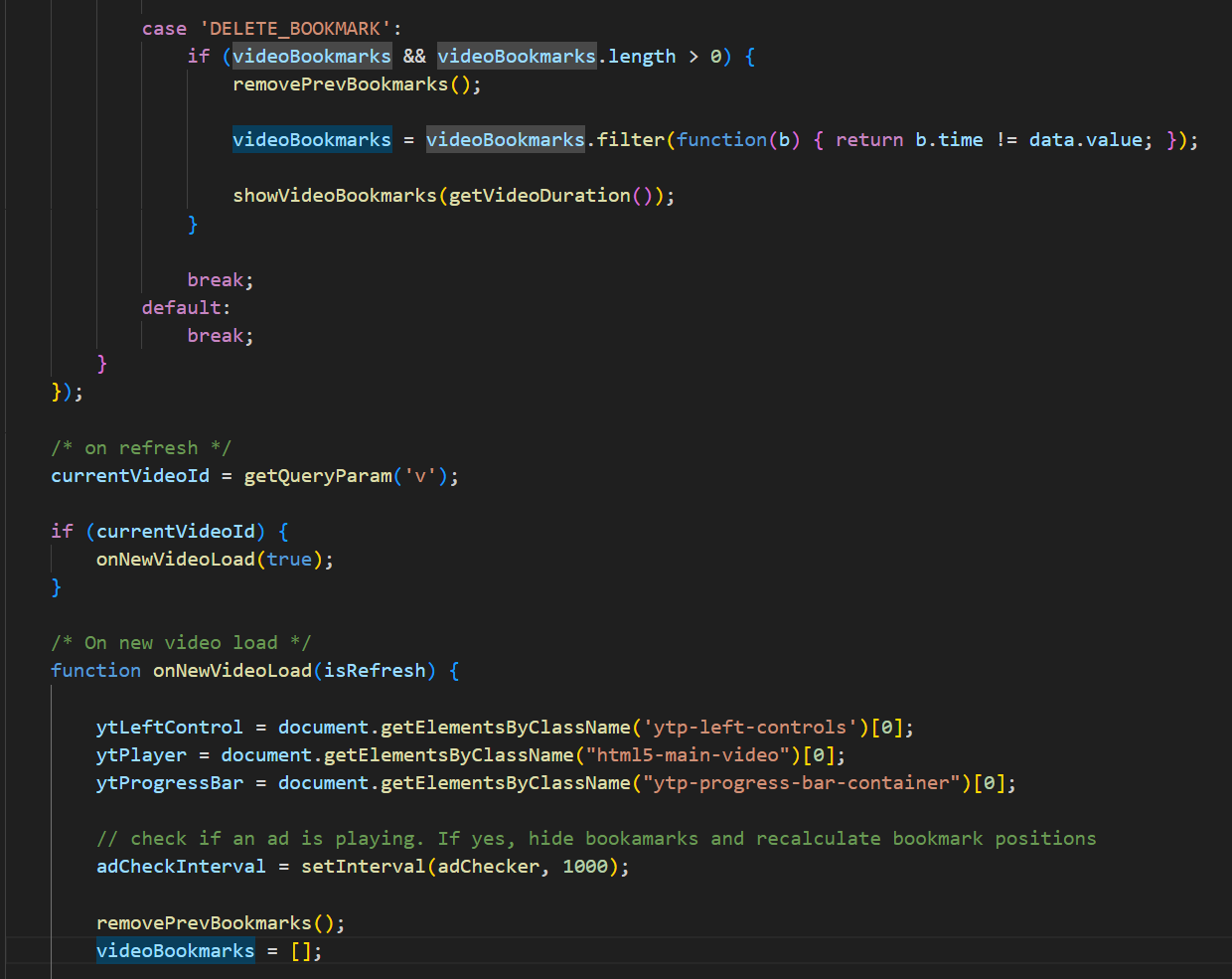
**Common.js:**



*Fig 6.1.8 Time Formatting into seconds*

**Content.js:**



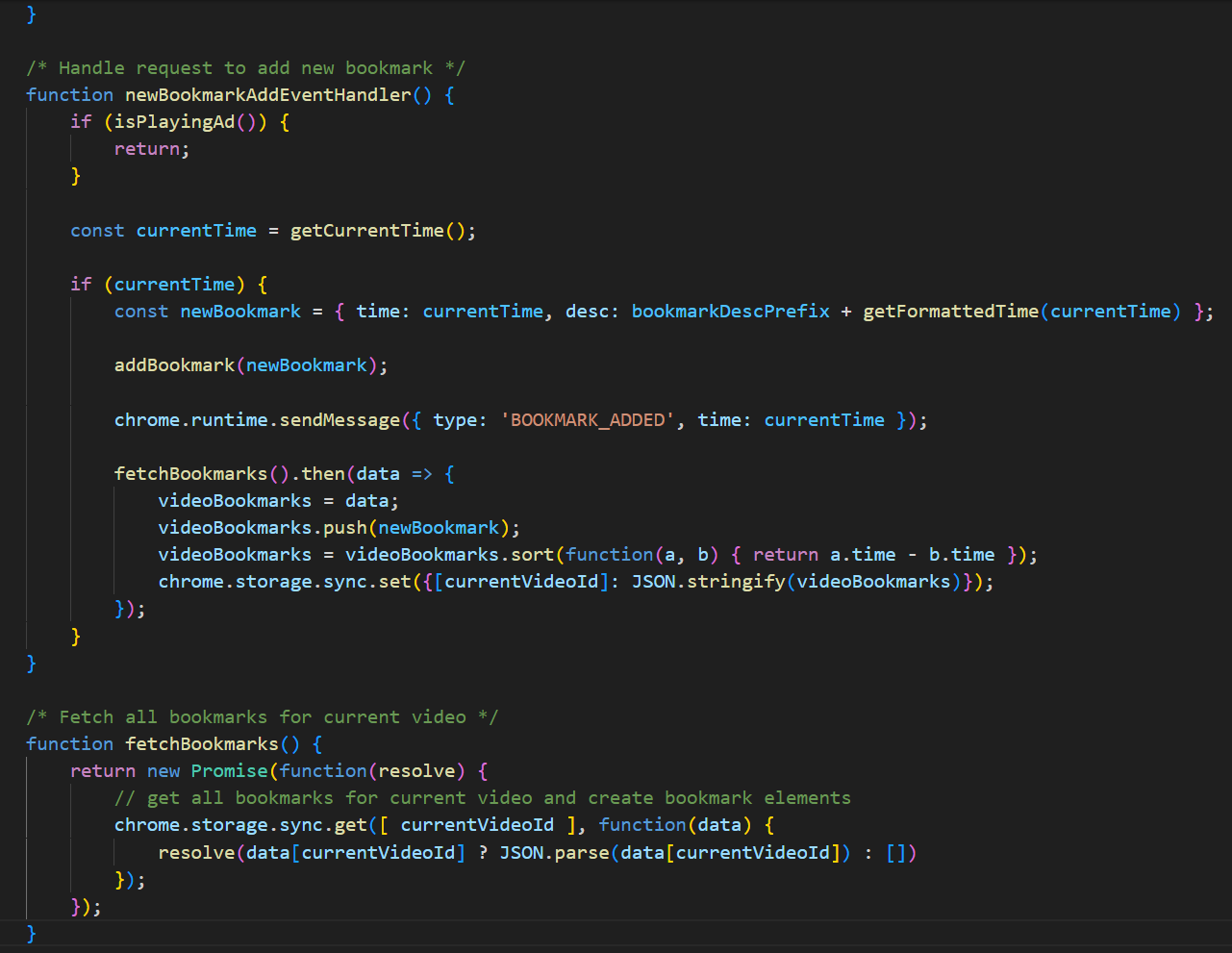


*Fig 6.1.9 Content.js*





*Fig 6.1.10 Content.js*





*Fig 6.1.11 Content.js*



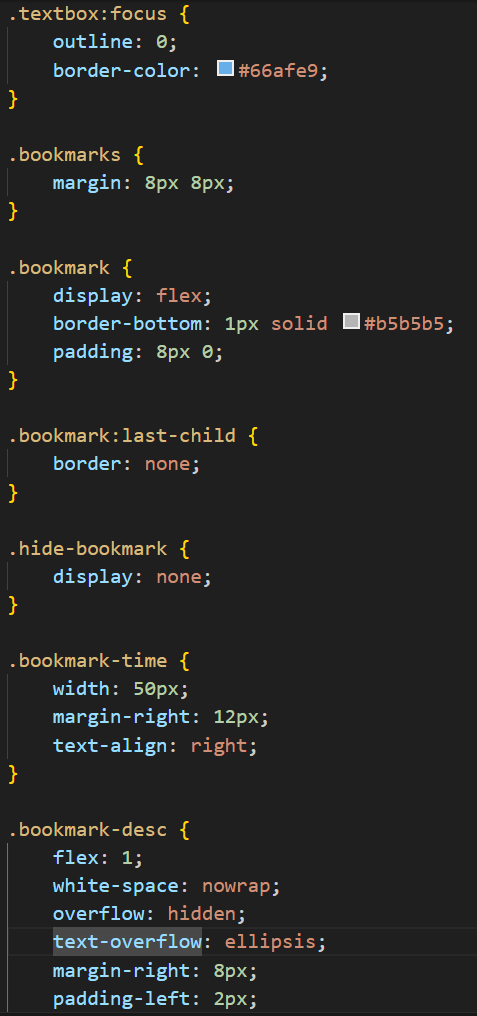


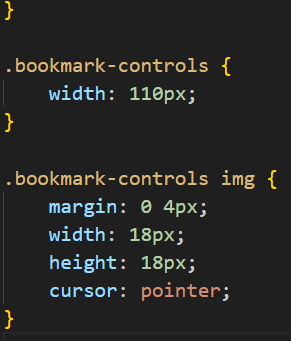
*Fig 6.1.12 Content.js*

**Popup.css:**



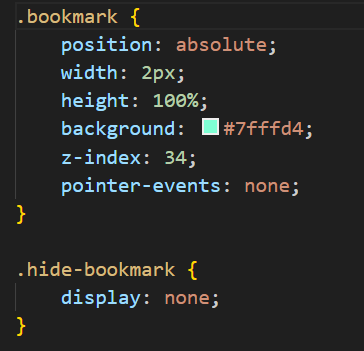
*Fig 6.1.13 popup.css*





*Fig 6.1.14 popup.css*

**Styles.css:**



*Fig 6.1.15 styles.css*

**7. TESTING**

In the domain of the YouTube extension project, testing serves as a critical phase to uphold quality assurance and identify potential bugs or flaws in the system.

## TYPES OF TESTING

### MANUAL TESTING

**Manual Testing** is a type of software testing in which test cases are executed manually by a tester without using any automated tools. The purpose of Manual Testing is to identify the bugs, issues, and defects in the software application. Manual software testing is the most primitive technique of all testing types and it helps to find critical bugs in the software application.

Any new application must be manually tested before its testing can be automated. Manual Software Testing requires more effort but is necessary to check automation feasibility. Manual Testing concepts does not require knowledge of any testing tool. One of the Software Testing Fundamental is “**100% Automation is not possible “**. This makes Manual Testing imperative.

### AUTOMATED TESTING

**Automation Testing** is a software testing technique that performs using special automated testing software tools to execute a test case suite. On the contrary, Manual Testing is performed by a human sitting in front of a computer carefully executing the test steps.

The automation testing software can also enter test data into the System Under Test, compare expected and actual results and generate detailed test reports. Software Test Automation demands considerable investments of money and resources.

## SOFTWARE TESTING METHODS

### BLACK BOX TESTING

Black box testing is a technique of software testing which examines the functionality of software without peering into its internal structure or coding. The primary source of black box testing is a specification of requirements that is stated by the customer. In this method, tester selects a function and gives input value to examine its functionality, and checks whether the function is giving expected output or not. If the function produces correct output, then it is passed in testing, otherwise failed. The test team reports the result to the development team and then tests the next function. After completing testing of all functions if there are severe problems, then it is given back to the development team for correction.

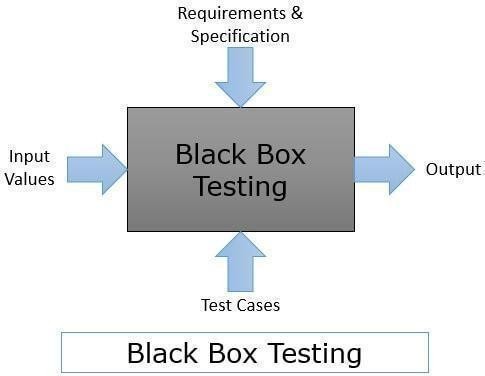


Fig. 7.2.1.1 Black Box Testing

#### GRAY BOX TESTING

Gray box testing is a software testing method to test the software application with partial knowledge of the internal working structure. It is a **combination of black box and white box testing** because it involves access to internal coding to design test cases as white box testing and testing practices are done at functionality level as black box testing.

Gray box testing commonly identifies context-specific errors that belong to web systems. For example; while testing, if tester encounters any defect, then he makes changes in code to resolve the defect and then test it again in real time. It concentrates on all the layers of any complex software system to increase testing coverage. It gives the ability to test both presentation layer as well as internal coding structure. It is primarily used in integration testing and penetration testing.

This testing technique is a combination of Black box testing and White box testing. In Black box testing, the tester does not have any knowledge about the code. They have information for what will be the output for the given input. In White box testing, the tester has complete knowledge about the code. Grey box testers have knowledge of the code, but not completely.



Fig. 7.2.2.1 Grey Box Testing

#### WHITE BOX TESTING

White box testing is an approach that allows testers to inspect and verify the inner workings of a software system—its code, infrastructure, and integrations with external systems. White box testing is an essential part of automated build processes in a modern Continuous Integration/Continuous Delivery (CI/CD) development pipeline. White box testing is often referenced in the context of Static Application Security Testing (SAST), an approach that checks source code or binaries automatically and provides feedback on bugs and possible vulnerabilities. White box testing is a testing technique, that examines the program structure and derives test data from the program logic/code. The other names of glass box testing are clear box testing, open box testing, logic driven testing or path driven testing or structural testing.

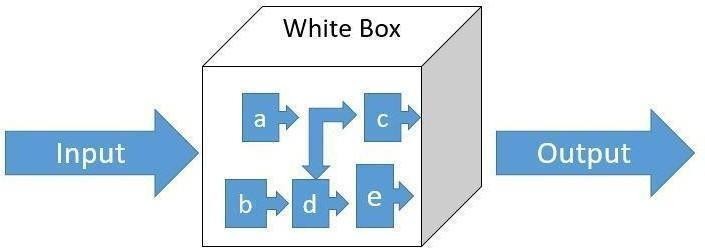


Fig. 7.2.3.1 White Box Testing

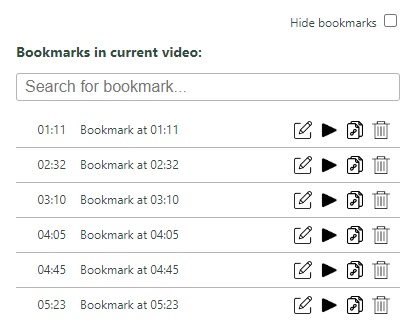
## TESTING

## TEST CASES

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sl.no** | **Testcase** | **Expected Result** | **Actual Result** | **Pass/ Fail** |
| 1. | Logo is present in the Youtube page | Present | Present | PASS |
| 2. | Showing the extension in the browser | Present | Present | PASS |
| 3. | Adding the Bookmark | Yes | Yes | PASS |
| 4. | Showing Timelines in the bookmark page | Yes | Yes | PASS |
| 5. | Videos start from Bookmark | Yes | Yes | PASS |
| 6. | Deleting the Bookmark after watching | Yes | Yes | PASS |
| 7. | Rename the Bookmark | Yes | Yes | PASS |
| 8. | Only Youtube.com videos | Yes | Yes | PASS |

*Table 7.3.1: Testcases*

**8. OUTPUT:**

****

*Fig 8.1 Final Output of the Extension.*

# 9. CONCLUSION AND FURTHER WORK

In conclusion, the development of the Google Chrome extension for YouTube bookmarking presents a valuable solution for users seeking a streamlined and efficient way to save and revisit their favorite videos. The extension's core functionality, including one-click bookmarking and seamless integration with the YouTube platform, contributes to an enhanced user experience. The minimalist design and focus on intuitive interactions aim to simplify the bookmarking process, providing users with a dedicated tool tailored specifically for YouTube content management.

* + - **User Feedback Integration:**

Implementing a feedback mechanism for continuous improvement based on user experiences.

* + - **Advanced Bookmarking Options:**

Exploring additional features like categorization, tagging, or playlist creation for advanced content organization.

* + - **Enhanced UI/UX Design:**

Continuosly refining the user interface and experience based on design trends and user feedback.

* + - **Browser Compatibility Expansion:**

Extending compatibility to other popular browsers for a consistent experience across platforms.

# REFERENCES

1. <https://developer.chrome.com/docs/extensions>
2. <https://developers.google.com/youtube/v3>
3. <https://developer.mozilla.org/en-US/docs/Learn>