S.P. Mandali’s
Ramnarain Ruia Autonomous College
Matunga, Mumbai

Project Report
On
Multi-Functional Chrome Extension for Productivity and Content Management

Project Guide
Prof. Ms. Kiran Prajapati

Project By
Mr. Sanskar Bhushankar
TYBSC
Department of Computer Science
2023 - 2024

Acknowledgement

First and foremost, I extend my gratitude to the Almighty for His blessings throughout this project.
I am deeply thankful to Ramnarain Ruia Autonomous College for granting me the opportunity to embark on this journey of discovery.
My sincere appreciation goes to our esteemed Principal, Dr. Anushree Lokur, whose consistent encouragement and support have propelled me forward.
Special thanks are due to Prof. Ms. Aarya Tawde, Head of the Computer Science Department, whose guidance and motivation have empowered me to explore new frontiers.
I owe a profound debt of gratitude to Prof. Ms. Kiran Prajapati, my dedicated project guide, for her timely guidance, supervision, and invaluable suggestions, which have significantly shaped the development of my project. Her unwavering support and encouragement have been a constant source of inspiration.
I am also immensely thankful to Prof. Ms. Rasika Mundhe, and Prof. Mr. Mahavir Advaya, for their invaluable guidance during challenging phases of the project. Their wisdom and encouragement have been instrumental in overcoming obstacles.
Heartfelt gratitude to Prof. Ms. Edith Juni and Prof. Ms. Kiran Prajapati for their insights and advice, which have broadened my perspective and enriched the project.
I am deeply indebted to my family for their unwavering support, guidance, and motivation. Their love and encouragement have been my anchor throughout this journey.
Lastly, I extend my gratitude to all my friends for their invaluable suggestions and guidance, which have played a significant role in shaping this project.
Thank you to everyone who has been a part of this rewarding journey.

Thanking,
Sanskar Ramesh Bhushankar

INDEX

A. Preliminary Investigation

Description of Present System

The purpose of the project is to develop a versatile tool for web browsers aimed at enhancing their performance and productivity, particularly tailored for online tasks and students.

This innovative tool, presented as a Chrome extension, aims to revolutionize the browsing experience by offering a comprehensive suite of features designed to streamline various online activities and cater to the needs of students and internet users alike.

At its core, the tool serves as a multifunctional extension for the Chrome browser, providing users with an array of services and functionalities across different websites.

Its primary objective is to augment productivity and efficiency during online tasks, making it an indispensable asset for students, professionals, and anyone seeking to optimize their browsing experience.

The tool is designed to cater to various types of websites, including popular platforms like YouTube, news websites, tutorial websites, and more. By seamlessly integrating with these platforms, the extension extends its capabilities to enhance user interactions, simplify content management, and facilitate learning and information retrieval.

Limitations of the Present System

Fragmented Workflow: Users face challenges in managing tasks efficiently as they need to switch between various tools and tabs, leading to a fragmented workflow and reduced productivity.

Manual Content Organization: The absence of automated content organization tools requires users to manually categorize and store information, consuming valuable time and effort.

Accessibility Issues: Essential functionalities such as text summarization or distraction-free modes are inaccessible, hindering users' ability to efficiently consume content and focus on tasks.

Contextual Understanding: Without note-making and annotation capabilities, users may struggle to capture and retain important information within the browsing context, leading to a loss of context and decreased productivity.

Proposed System

Your project entails the development of a sophisticated Chrome extension aimed at revolutionizing the browsing experience by addressing the inherent limitations of the current system. By seamlessly integrating a plethora of innovative features directly into the Chrome browser, the extension serves as a comprehensive solution to enhance workflow efficiency, automate content management, and improve accessibility.

Critical to overcoming accessibility challenges, essential features such as text summarization and distraction-free modes are readily accessible within the extension, promoting heightened productivity and focus during online tasks. Furthermore, the inclusion of note-making and annotation capabilities allows users to capture and retain crucial information within the browsing context, facilitating contextual understanding and information retention.

In summary, my project offers a holistic solution to surmount the limitations of the present system, providing users with a versatile Chrome extension that enhances workflow efficiency, automates content management, fosters collaboration, and improves accessibility. By seamlessly integrating these features into the browsing experience.

Advantages of Proposed System

Streamlined Workflow: Unlike the fragmented workflow caused by switching between multiple tools and tabs, your system offers a cohesive and integrated solution within the Chrome browser. This streamlines users' workflow by consolidating essential features and functionalities into a single extension, enabling seamless navigation and task management.

Automated Content Management: Your system addresses the manual effort required for content organization by incorporating automated tools for capturing and categorizing information. By automating tasks such as content capture, storage, and organization, users can save valuable time and focus more on their core tasks.

Improved Accessibility: Your system ensures accessibility to essential features such as text summarization and distraction-free modes, enhancing users' ability to consume content efficiently and focus on tasks without distractions. By providing easy access to these features directly within the browser interface, your system promotes a more productive and focused browsing experience.

Contextual Understanding and Retention: Through note-making and annotation capabilities, your system enables users to capture and retain important information within the browsing context. This promotes contextual understanding and retention of key insights, helping users stay organized and productive while browsing the web.

Features and functionalities

NotesMakerPDF:
The extension empowers users to strategically mark and capture screenshots for any given Tab and store it in chrome storage and the user will be able to save all screenshots in form of pdf.
Also user will be able to take a quick note of anything which the user don’t need to save all the time it gets autosave in the chrome storage and can be downloaded to pdf as well.

Web Page Content Summarization:

Leveraging advanced textual analysis techniques, the extension distills lengthy web content into minimal summaries. This feature expedites comprehension by encapsulating core information, facilitating efficient consumption of voluminous material.

YouTube Video Summarization:
Facilitating to summarize the lengthy youtube videos into short and highlighting the major parts from the video which saves time from the user to watch entire video.

Distraction-Free Mode:
Enhancing user focus, this feature allows for the elimination of extraneous elements such as suggestions, likes, dislikes, and comments on YouTube videos. By activating a 'movie mode,' users can enjoy a clutter-free viewing experience akin to dimming lights in a theater. This promotes a seamless, undistracted engagement with the content at hand.

Programming languages:

For programing of the chrome extension javascript ,flask and python will be used as backend as it provides proper api availability usage for manifest.json file used in the extension .
Use of gemini ai api will be done with the help of python

Feasibility Study:

a. Technical Feasibility:

Availability of APIs: The availability and functionality of APIs youube and gemini ai
.
Data Extraction and Analysis: The feasibility of extracting and analyzing data from different platforms like youtube and websites

System Integration: The technical feasibility of integrating data from various sources and presenting it in a unified interface.

b. Financial Feasibility:

Cost Analysis: The costs associated with developing the extension , including software development, hosting infrastructure chrome webstore payment , and any necessary licenses or APIs.

Revenue Generation: Potential revenue streams, such as offering the tool as a subscription service, licensing the software to businesses, or providing premium features to users.

c. Operational Feasibility:

User Acceptance: Assess the potential users' interest and demand for such a tool by conducting surveys or gathering feedback.

User-Friendly Interface: Ensure that the tool has an intuitive and user-friendly interface to facilitate ease of use and adoption.

Technical Support and Maintenance: Consider the resources and support required to provide ongoing technical support, bug fixes, and updates for the tool.

e. Schedule Feasibility:

Project Timeline: Evaluate the estimated time required for development, testing, and deployment

Resource Allocation: Determine the availability of skilled developers, designers, and testers required for the project.

Stake holders

1. Users:
The primary stakeholders of the Chrome Extension project will be the users of the extension. These could be individuals, content consumers, students, professionals, or anyone seeking a streamlined browsing experience with enhanced features.

2. Project Team:
The project team, including developers, designers, testers, project managers, and other team members, are stakeholders who contribute to the development, implementation, and maintenance of the Chrome Extension.

3. Management:
Management or decision-makers within your organization or institution are stakeholders responsible for providing resources, guidance, and strategic decisions that impact the project's direction and success.

4. Browser Platforms:
Browser platforms like Google Chrome, Mozilla Firefox, and others are indirect stakeholders. Ensuring compatibility and compliance with browser guidelines and policies is essential for the extension's distribution and use.

5. Legal Advisors:
Legal advisors or consultants with expertise in data privacy, intellectual property, and other relevant areas are stakeholders involved in ensuring compliance, minimizing legal risks, and addressing potential legal concerns.

Project Requirements

Hardware Requirements (Computer) for developing:

- **Operating System:** Windows 10 or later versions (64-bit).
- **RAM:** 8 GB or more (recommended).
- **Disk Space:** 50 GB (minimum).
- **PC Screen Resolution:** 1366 x 770p.

Technologies Used

1. JavaScript
2. HTML
3. CSS
4. python
5. Chrome api
6. Visual Studio Code

Gantt Chart

🟩 **Actual Completion**
🟦 **Expected Time of Completion**
🟥 **Break**

Phase Title	June					July					August				
PROJECT APPROVAL	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
			🟩	🟩											
			🟦	🟦											
	🟥	🟥													
PRELIMINARY INVESTIGATION	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
					🟩	🟩									
					🟦	🟦									
SYSTEM ANALYSIS															
PROJECT DEVELOPMENT METHODOLOGY	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
							🟩	🟩							
							🟦	🟦							
EVENT TABLE	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
							🟩	🟩							
							🟦	🟦							
USE CASE DIAGRAM	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
									🟩						
									🟦						
USE CASE SCENARIO	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
										🟩					
										🟦					
ACTIVITY DIAGRAM	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
											🟥	🟥	🟩	🟩	
													🟦	🟦	
Phase	September					October					November				
STATE-CHART DIAGRAM	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
	🟩														
	🟦														
PACKAGE DIAGRAM	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
		🟩													
		🟦													
DEPLOYMENT DIAGRAM	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
			🟩												
			🟦												
SYSTEM CODING	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
				🟩	🟩							🟩	🟩		
				🟦	🟦							🟦	🟦		
						🟥	🟥	🟥	🟥	🟥					
Phase	December					January					February				
SYSTEM CODING	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩			🟩	🟩		
	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦			🟦	🟦		
SUBMISSION	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
														🟩	🟩
														🟦	🟦

B.SYSTEM ANALYSIS

Project Development

For the development of my project, we can consider iterative and incremental development methodologies such as Agile. Agile methodologies promote adaptive planning, evolutionary development, and rapid delivery. Here's an example of a project development methodology for my study buddy:

1. Project Initiation:

- Define the project scope, objectives, and deliverables.
- Identify the key stakeholders and gather their requirements and expectations.
- Conduct a feasibility study to assess the project's technical and financial viability.

2. Requirements Gathering:

- Collaborate with stakeholders, including students, teachers, and parents, to gather detailed requirements and understand their needs.
- Identify the essential features and functionalities of the study buddy.
- Prioritize requirements based on their importance and potential impact.

3. Agile Planning:

- Create a product backlog consisting of user stories or tasks that need to be implemented.
- Break down the user stories into smaller, manageable tasks.
- Estimate the effort and complexity of each task using techniques like story points or planning poker.
- Create a prioritized list of tasks for each iteration or sprint.

4. Iterative Development:

- Plan and execute short iterations or sprints, typically lasting 1-4 weeks.
- Develop and implement the features identified for each iteration.
- Conduct regular meetings, such as daily stand-ups, to discuss progress, address challenges, and synchronize the team's efforts.

- Continuously review and refine the product backlog based on feedback and changing requirements.

5. Continuous Testing and Quality Assurance:

- Perform continuous testing throughout the development process to ensure the functionality and reliability of the study buddy.

- Conduct unit testing, integration testing, and user acceptance testing (UAT).

- Implement an automated testing framework to streamline the testing process.

- Incorporate feedback from testing to improve the system's performance and usability.

6. Incremental Deployment:

- Deploy the developed features and functionalities in increments, ensuring continuous delivery and feedback.

- Regularly release updates and new versions of the study buddy to address user needs and incorporate improvements.

- Monitor the system's performance and gather user feedback to drive further enhancements.

7. Collaboration and Communication:

- Foster a collaborative environment by promoting open communication and regular interaction among team members.

- Encourage continuous feedback and engage stakeholders in the development process.

- Conduct regular review meetings to evaluate progress, address issues, and make necessary adjustments.

8. Documentation and Knowledge Sharing:

- Maintain proper documentation of the project, including requirements, design, development, and testing processes.

- Create user manuals and documentation to assist users in utilizing the study buddy effectively.

- Encourage knowledge sharing among team members through code reviews, technical discussions, and sharing best practices.

9. Project Monitoring and Control:

- Regularly monitor project progress against the defined timeline, budget, and quality targets.

- Identify and address any deviations or risks promptly.

- Continuously assess and adjust the project plan based on changing requirements or emerging challenges.

10. Project Closure:

- Conduct a final review to assess the project's success and achievement of objectives.

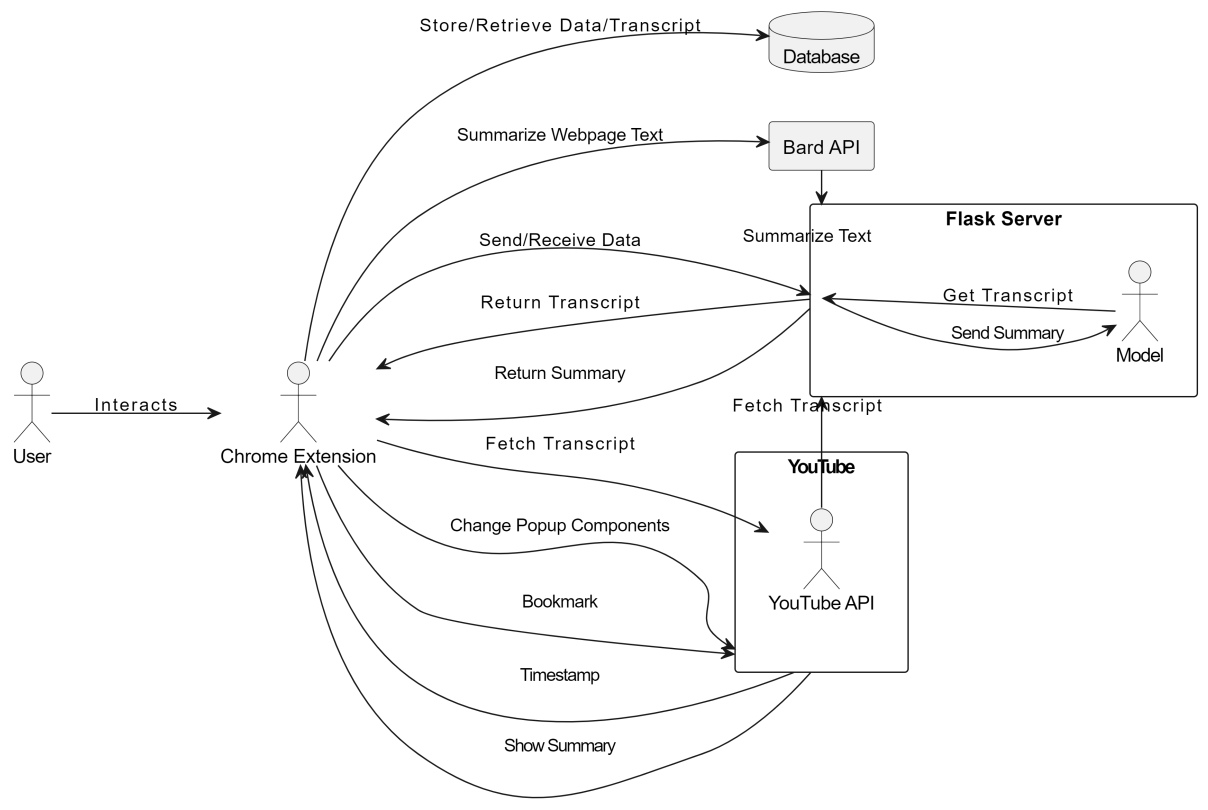
- Celebrate the project completion and recognize the efforts of the team.

- Perform post-implementation reviews and gather feedback from users to identify areas for improvement.

EVENT TABLE

Event	Trigger	Source	Use Case	Response	Destination
User clicks 'Take Screenshot' button	User action	Chrome Extension	Capture Screenshot	Capture screenshot of the current tab	Chrome Extension
User selects text on a webpage	User action	Chrome Extension	Add Text	Copy selected text to clipboard	Chrome Extension
User clicks 'Download PDF' button	User action	Chrome Extension	Generate PDF	Convert screenshots and text to PDF	Chrome Extension
User visits a YouTube video	User action	Chrome Extension	Summarize YouTube Video	Fetch transcript and summarize video content	Chrome Extension
User clicks 'Capture Webpage Text' button	User action	Chrome Extension	Summarize Webpage Text	Extract webpage text and generate summary	Chrome Extension
User triggers the focus mode on YouTube	User action	Chrome Extension	Enable Focus Mode on YouTube	Hide distractions like comments and videos	YouTube
User clicks 'Bookmark Video' button in YouTube tab	User action	Chrome Extension	Bookmark YouTube Video Timestamp	Prompt user to enter bookmark name	Chrome Extension
User confirms the bookmark name	User action	Chrome Extension	Bookmark YouTube Video Timestamp	Store bookmarked timestamp in Chrome storage	Chrome Extension

Use case diagram



Use Case Scenario

Use Case Scenario 1: Capture Screenshot and Save Text to Database

Primary Actor: User

Main Flow:

The user navigates to a webpage they want to capture.
The user clicks on the Chrome extension button to capture a screenshot.
The extension captures the screenshot of the current webpage.
Optionally, the user selects and copies some text from the webpage.

The user clicks on the extension button to save the captured screenshot and copied text to the database.
The extension stores the screenshot and text data in the database for future reference.

Use Case Scenario 2: Summarize YouTube Video and Add to PDF

Primary Actor: User

Main Flow:

The user visits a YouTube video page.
The user clicks on the Chrome extension button to initiate the summarization process.
The extension fetches the transcript of the YouTube video.
The extension sends the transcript to the Flask server running the Summary Model.
The Summary Model generates a summary of the video based on the transcript.
The summary is returned to the extension by the Flask server.
The extension adds the video summary to an existing or new PDF document.

Use Case Scenario 3: Summarize Webpage Text Using AI and Add to PDF

Primary Actor: User

Main Flow:

The user visits a webpage with substantial text content.
The user activates the text summarization feature through the Chrome extension.
The extension analyzes the text content of the webpage using AI algorithms.

The extension identifies the major points and key information from the webpage.
The extension generates a concise summary of the webpage content.
The summarized content is added to an existing or new PDF document by the extension.

Use Case Scenario 4: Enable Focus Mode on YouTube

Primary Actor: User

Main Flow:

The user opens a YouTube video page in their browser.
The user clicks on the Chrome extension button to activate the focus mode.
The extension modifies the YouTube page's components to hide distractions such as comments and suggested videos.
The user can now watch the video without any other elements distracting them.
Optionally, the user can disable the focus mode by clicking on the extension button again.
These use case scenarios demonstrate how the Chrome extension interacts with the user and performs various functionalities like capturing screenshots, summarizing content, and enhancing the YouTube viewing experience.

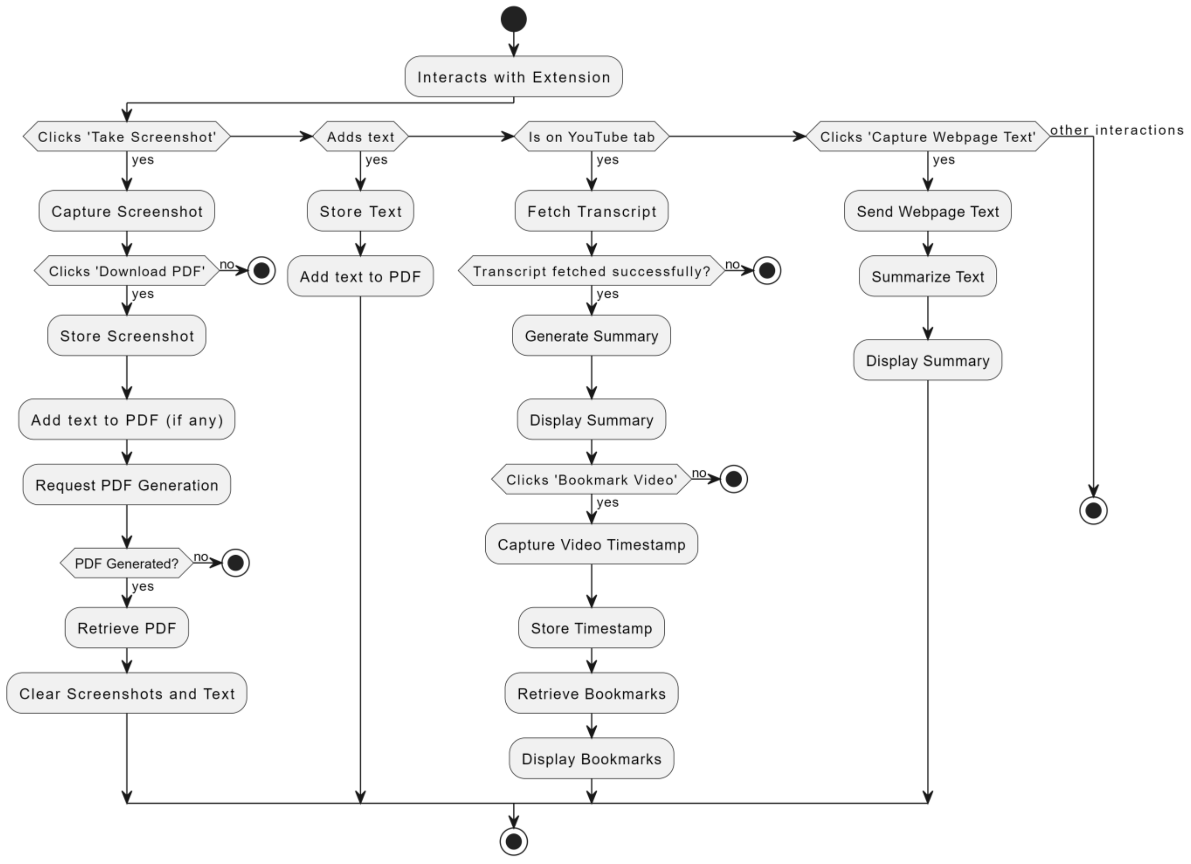
Use Case Scenario 4: Bookmark YouTube Video Timestamp

Primary Actor: User

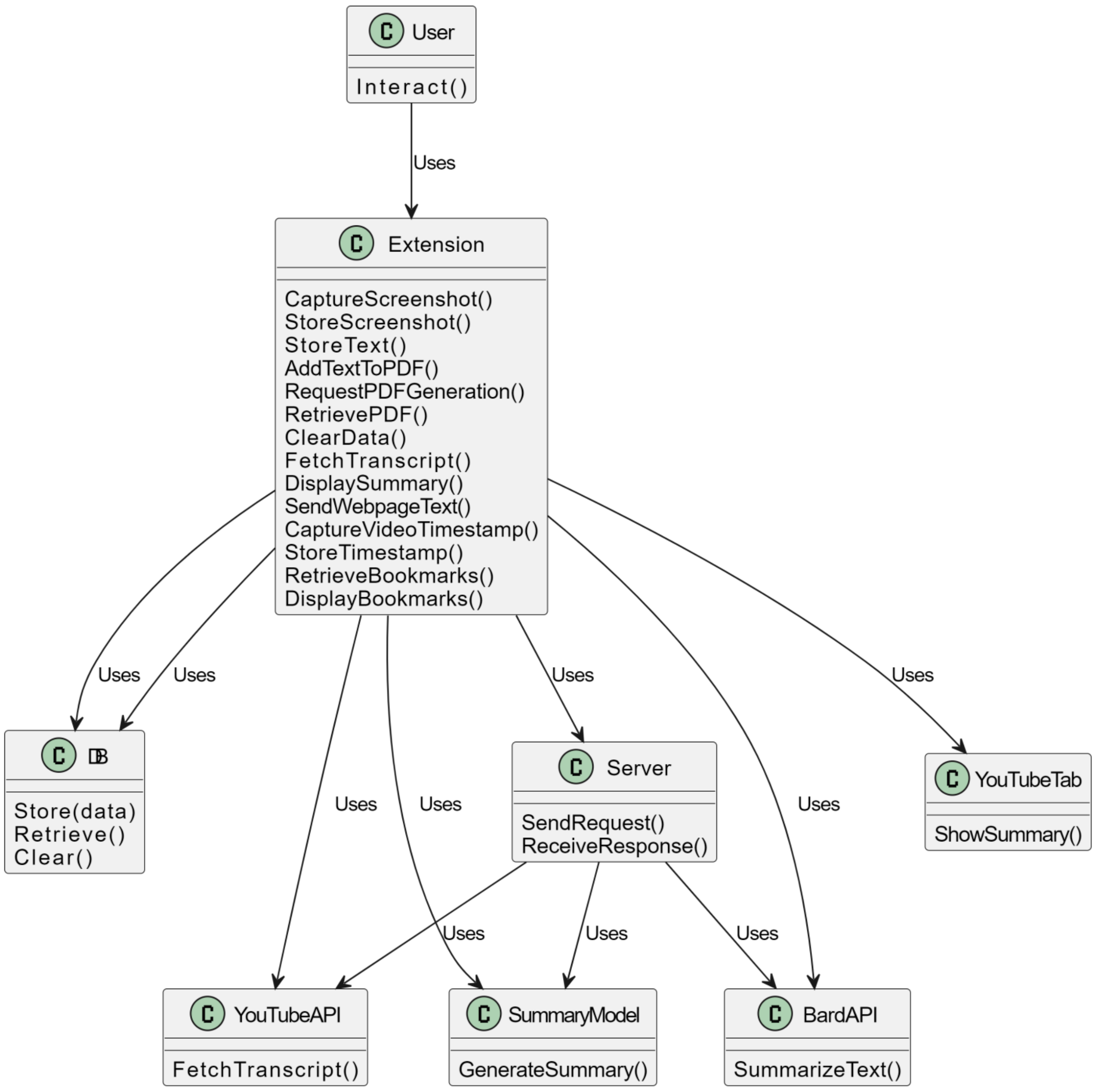
Main Flow:

The user watches a YouTube video and finds a specific timestamp they want to bookmark.
The user clicks on the injected bookmark button added by the Chrome extension on the YouTube video tab.
The extension prompts the user with an alert box to enter a name for the timestamp.
The user enters the name for the timestamp and clicks "OK".
The extension captures the current timestamp of the video along with the provided name.
The extension stores the timestamp and its associated name in the Chrome storage.
The user can see the bookmarked timestamp in the extension's interface.
When the user revisits the same YouTube video in the future, the extension retrieves and displays the saved timestamps for that video from the Chrome storage.
The user can click on a bookmarked timestamp to navigate directly to the saved timestamp in the video.
This use case scenario describes how the Chrome extension allows users to easily bookmark specific timestamps in YouTube videos and access them later for reference.

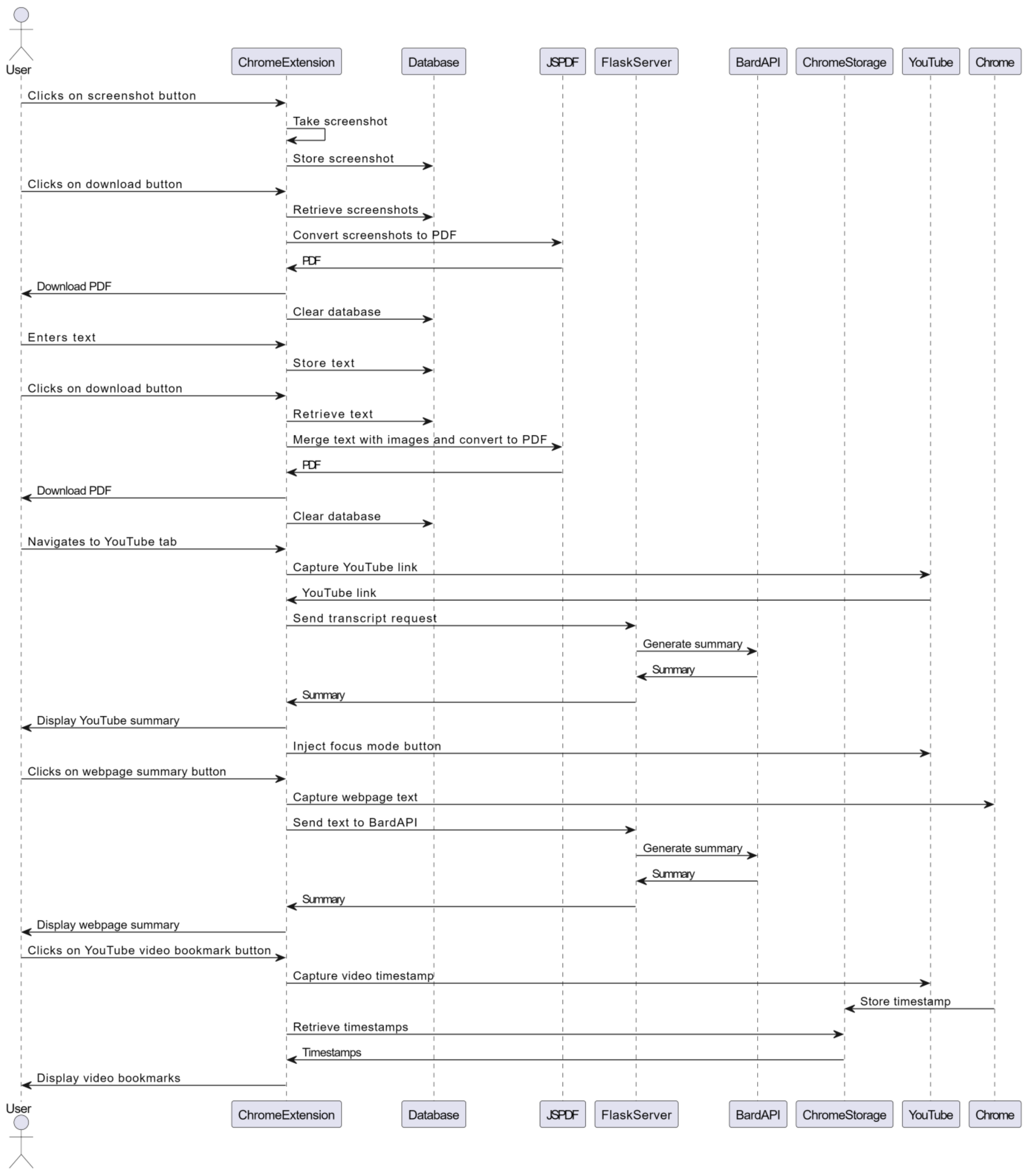
ACTIVITY DIAGRAM



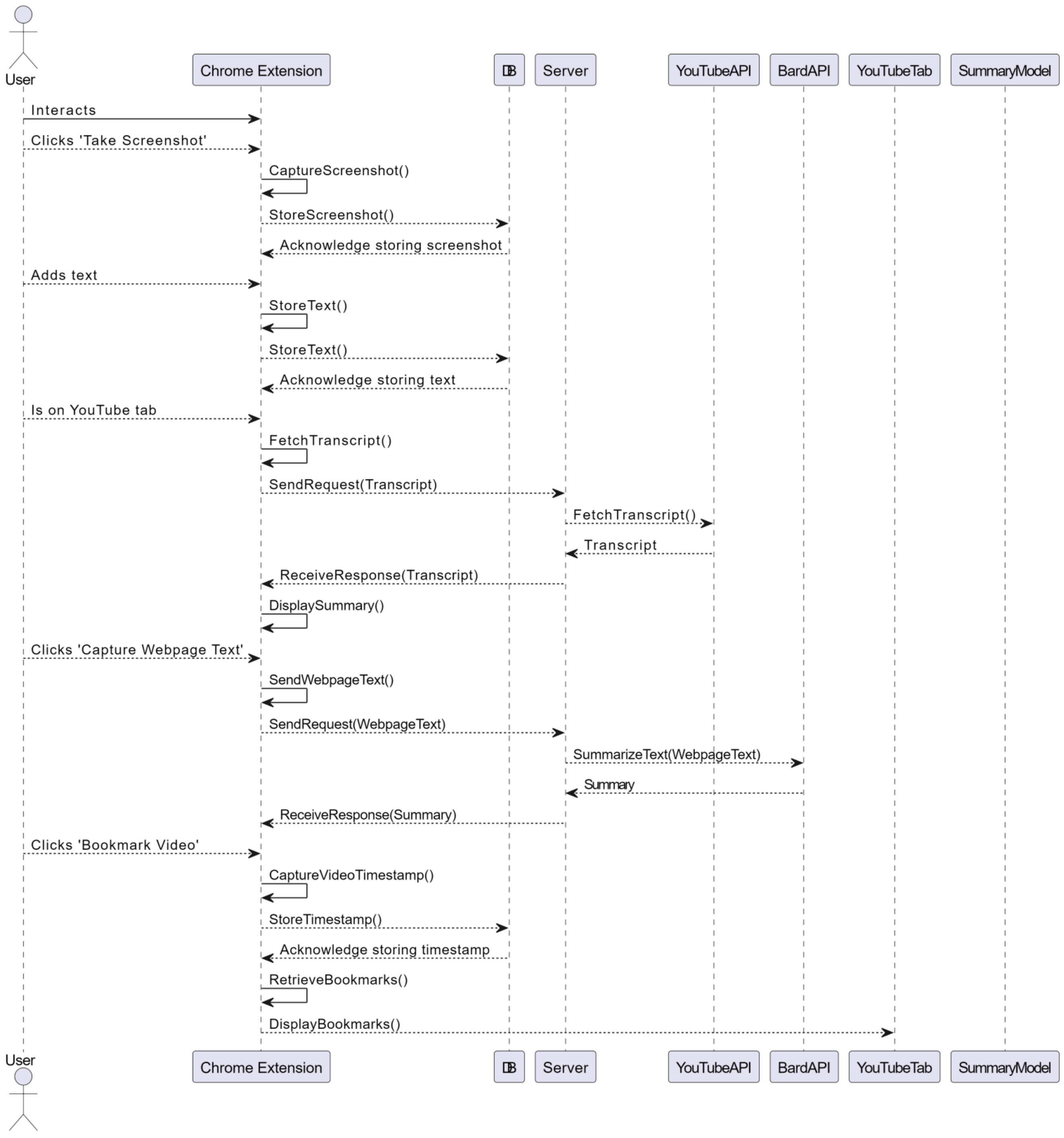
Class diagram



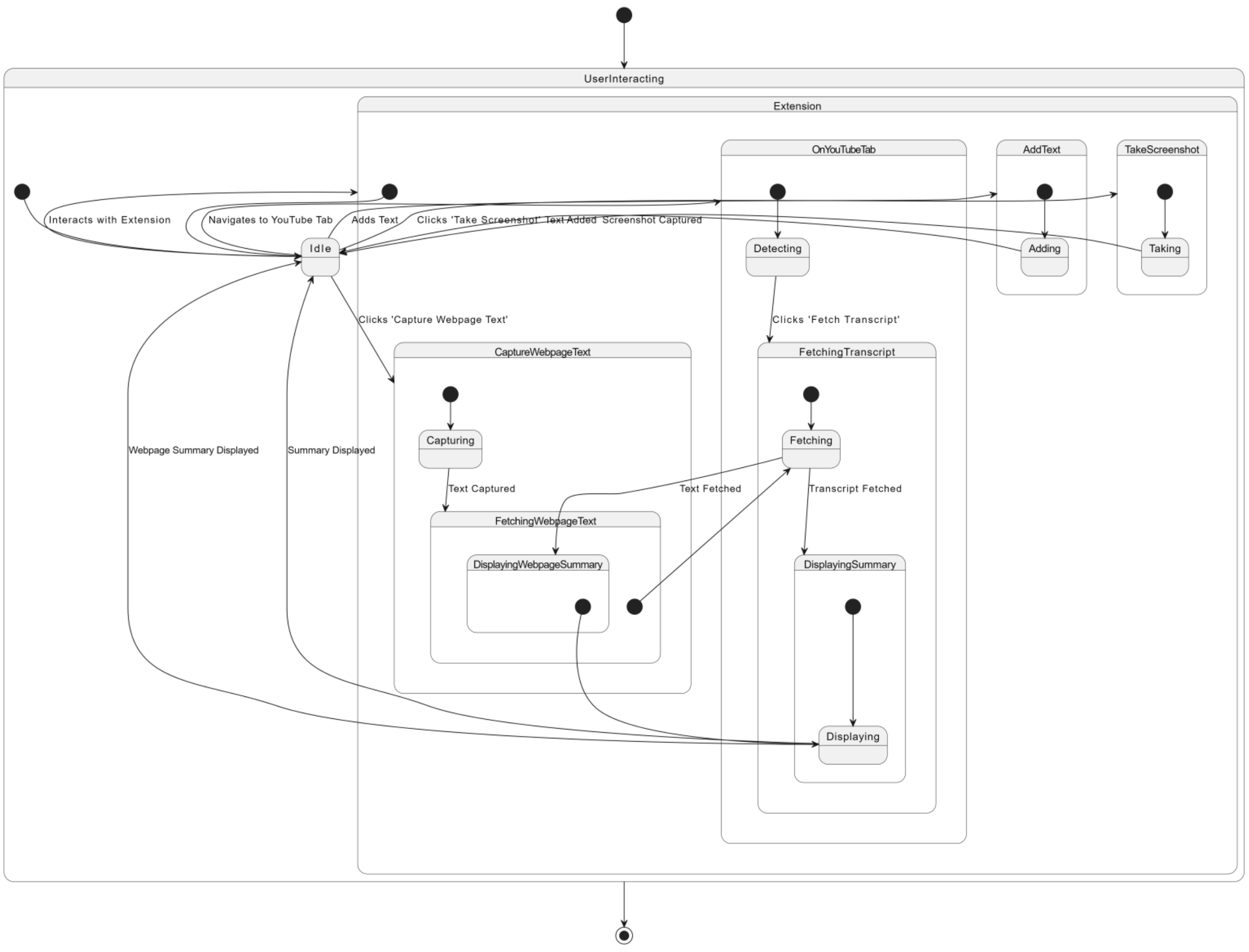
Sequence diagram



Collaboration Diagram

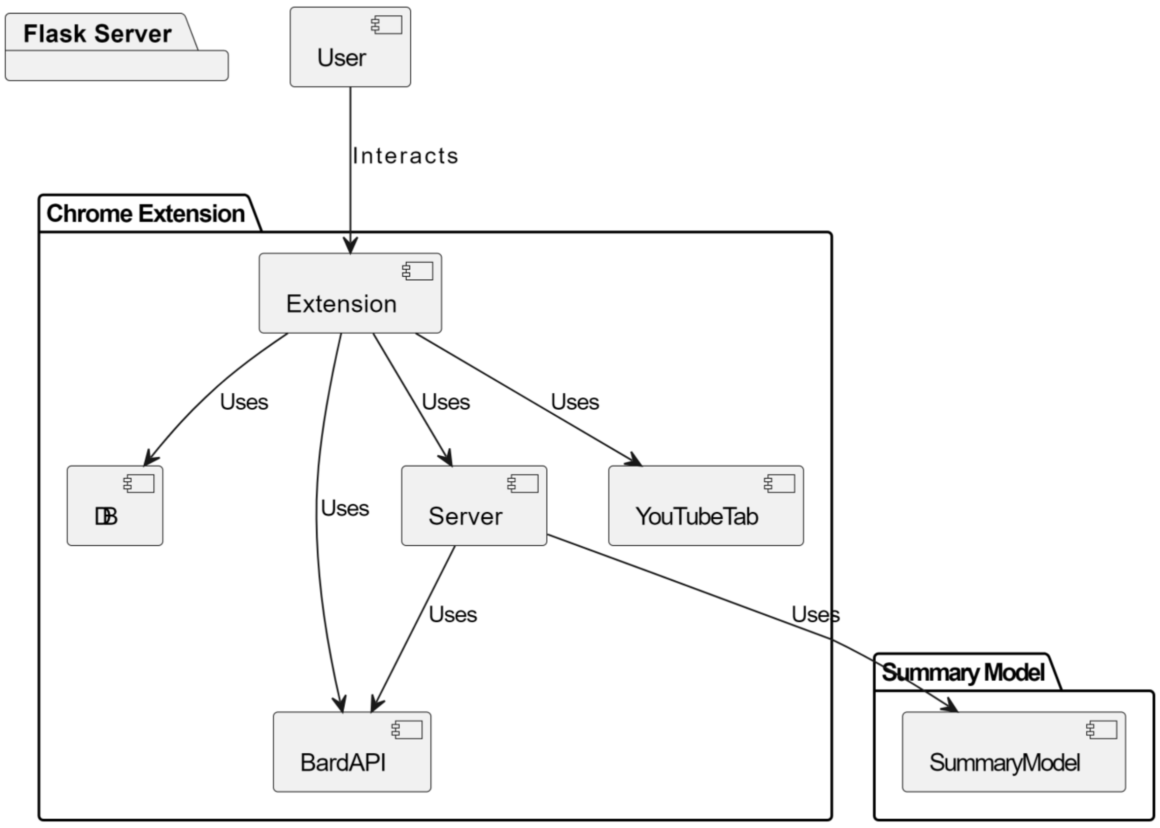


State-Chart Diagram

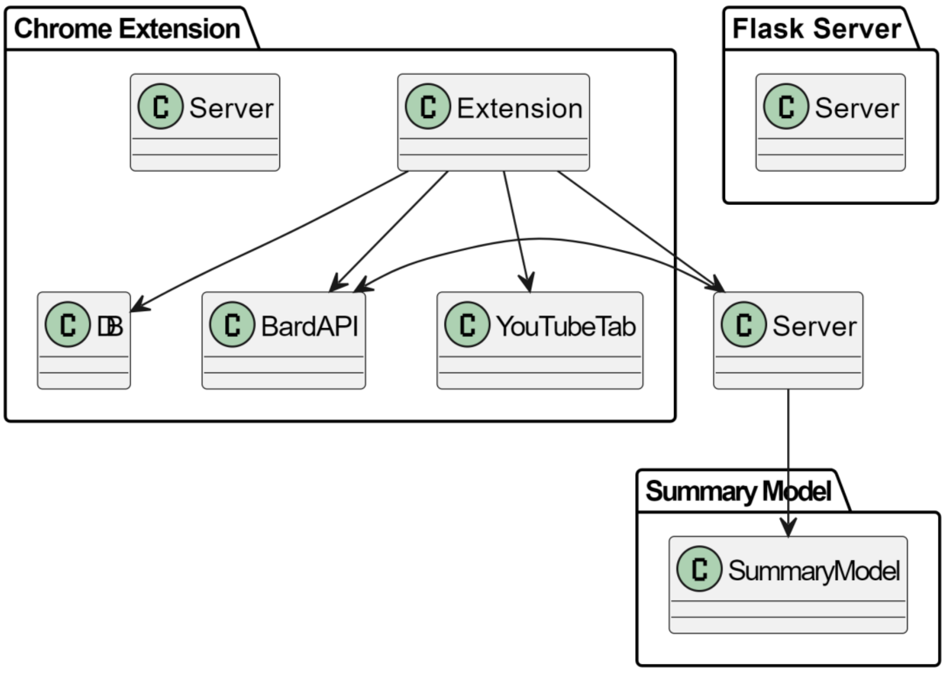


C.System design

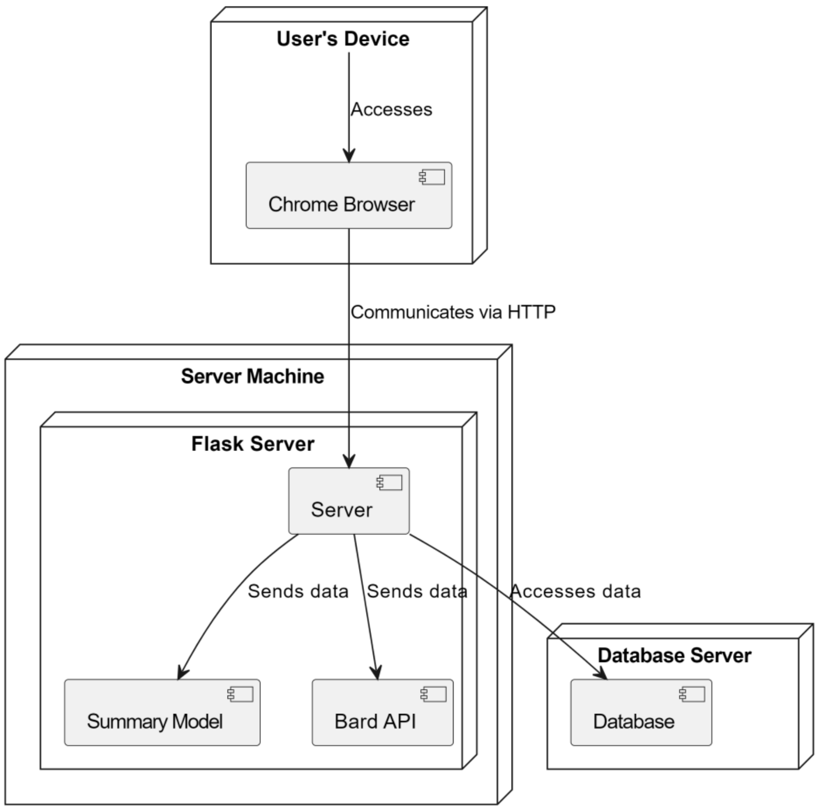
Component diagram



Package Diagram



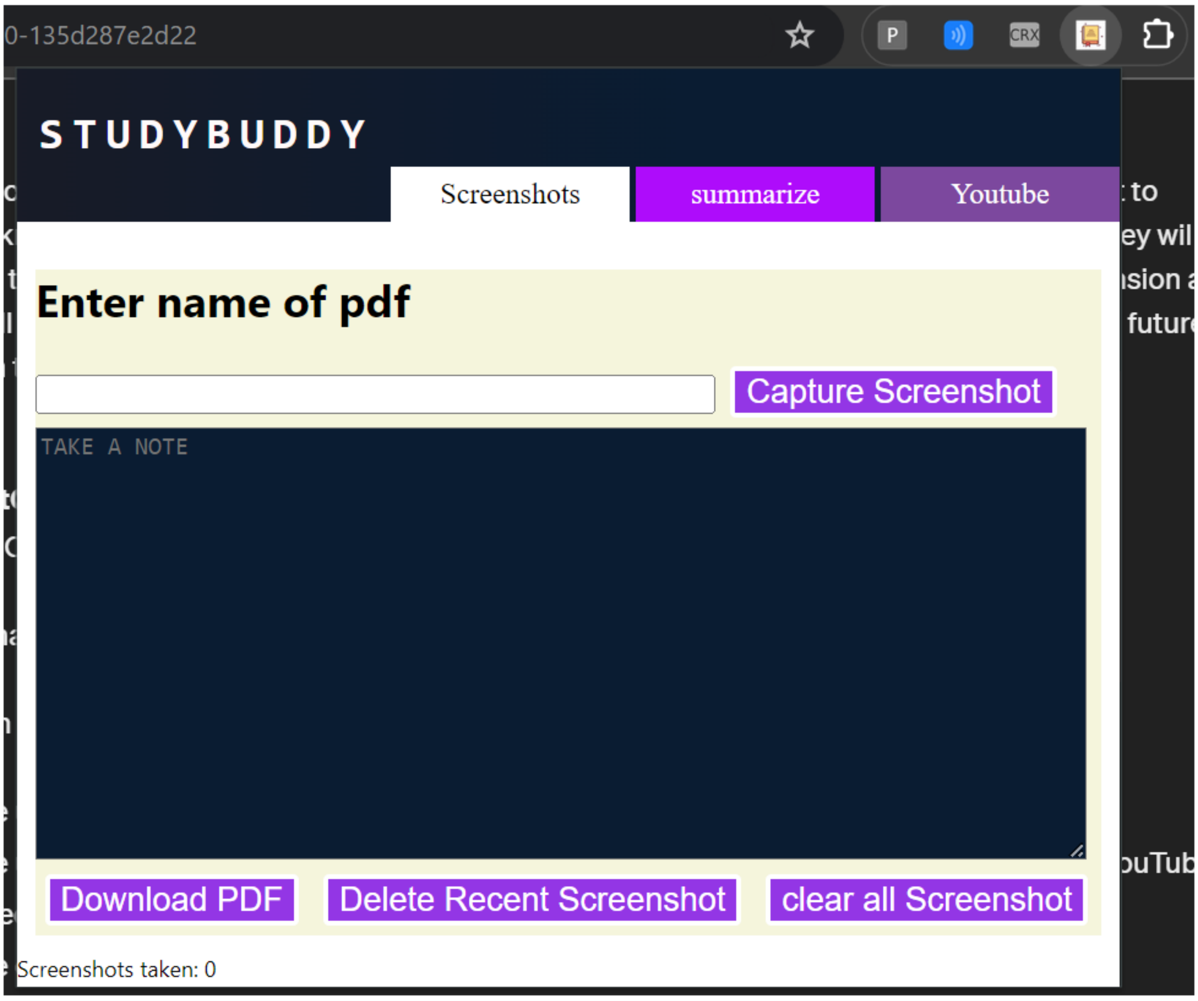
Deployment Diagram



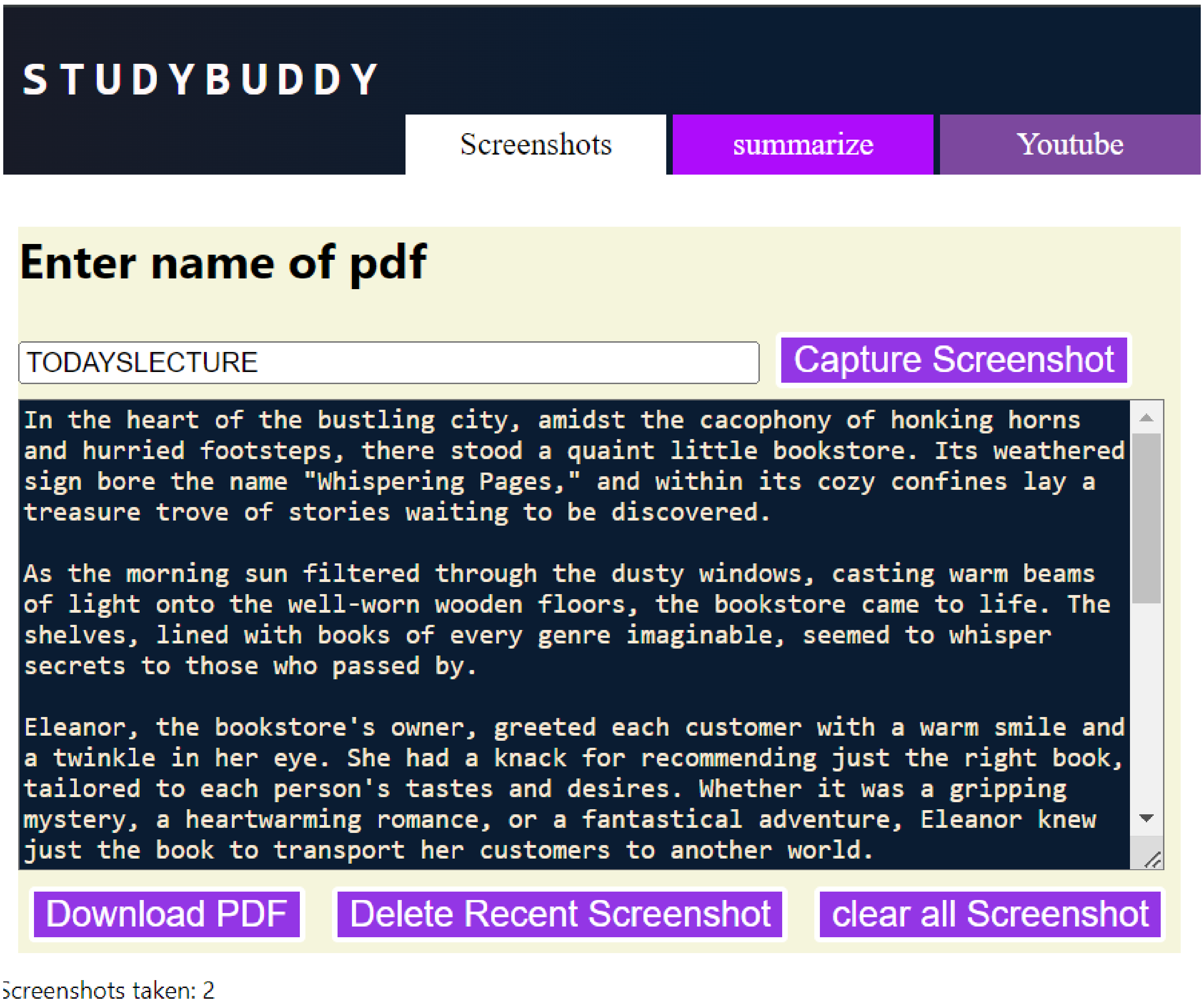
D.SYSTEM CODING/Testing

Screen Layout

HomePage



We can add text to the pdf and and store it as notes for future reference also can capture screenshot of any tab



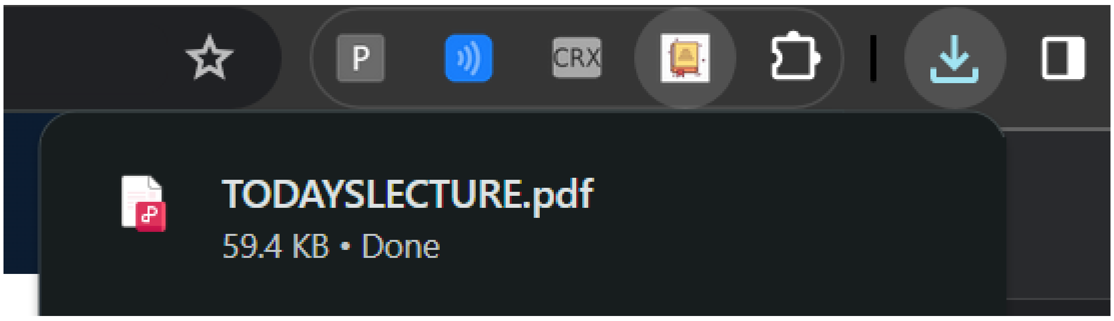
Enter name of pdf

TODAYSLECTURE

```
In the heart of the bustling city, amidst the cacophony of honking horns
and hurried footsteps, there stood a quaint little bookstore. Its weathered
sign bore the name "Whispering Pages," and within its cozy confines lay a
treasure trove of stories waiting to be discovered.

As the morning sun filtered through the dusty windows, casting warm beams
of light onto the well-worn wooden floors, the bookstore came to life. The
shelves, lined with books of every genre imaginable, seemed to whisper
secrets to those who passed by.

Eleanor, the bookstore's owner, greeted each customer with a warm smile and
a twinkle in her eye. She had a knack for recommending just the right book,
tailored to each person's tastes and desires. Whether it was a gripping
mystery, a heartwarming romance, or a fantastical adventure, Eleanor knew
just the book to transport her customers to another world.
```

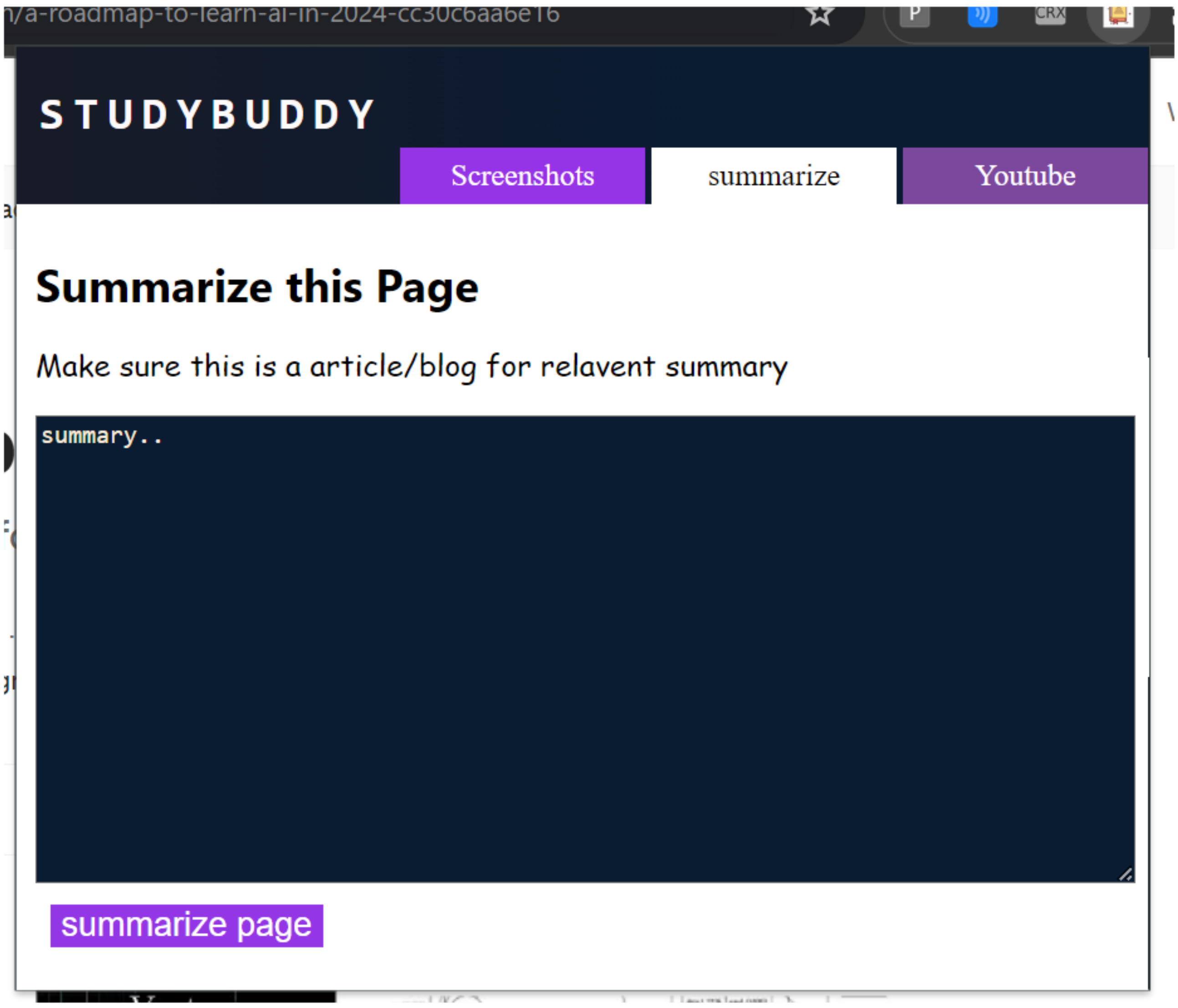
Screenshots taken: 2

Output we get when download pdf



TODAYSLECTURE.pdf
59.4 KB • Done

Summarize page

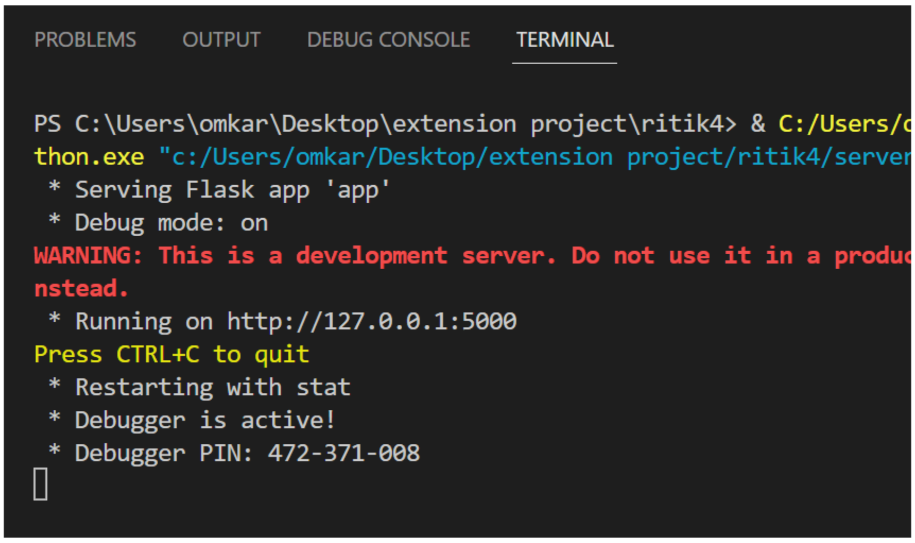


Summarize this Page

Make sure this is a article/blog for relavent summary

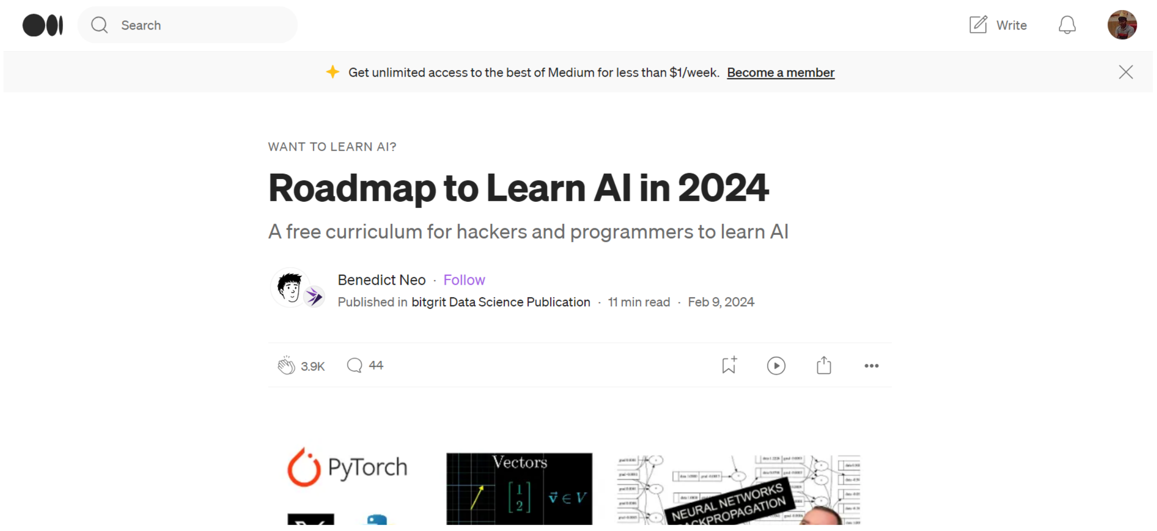
```
summary..
```

Server running in background



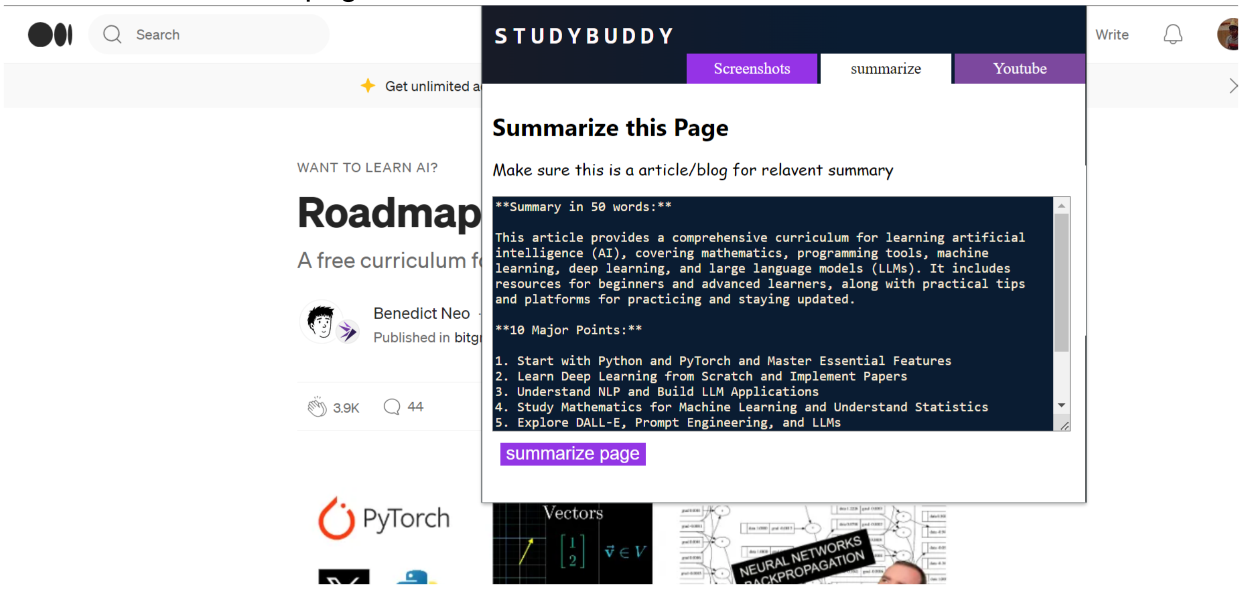
```
PS C:\Users\omkar\Desktop\extension project\ritik4> & C:/Users/o
thon.exe "c:/Users/omkar/Desktop/extension project/ritik4/server
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a produc
nstead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 472-371-008
```

Open any blog page or any page with relevant information



Search
Write
Get unlimited access to the best of Medium for less than $1/week. Become a member
WANT TO LEARN AI?
Roadmap to Learn AI in 2024
A free curriculum for hackers and programmers to learn AI
Benedict Neo · Follow
Published in bitgrit Data Science Publication · 11 min read · Feb 9, 2024
3.9K
44
PyTorch
Vectors
NEURAL NETWORKS
BACKPROPAGATION

Clicks on summarize page



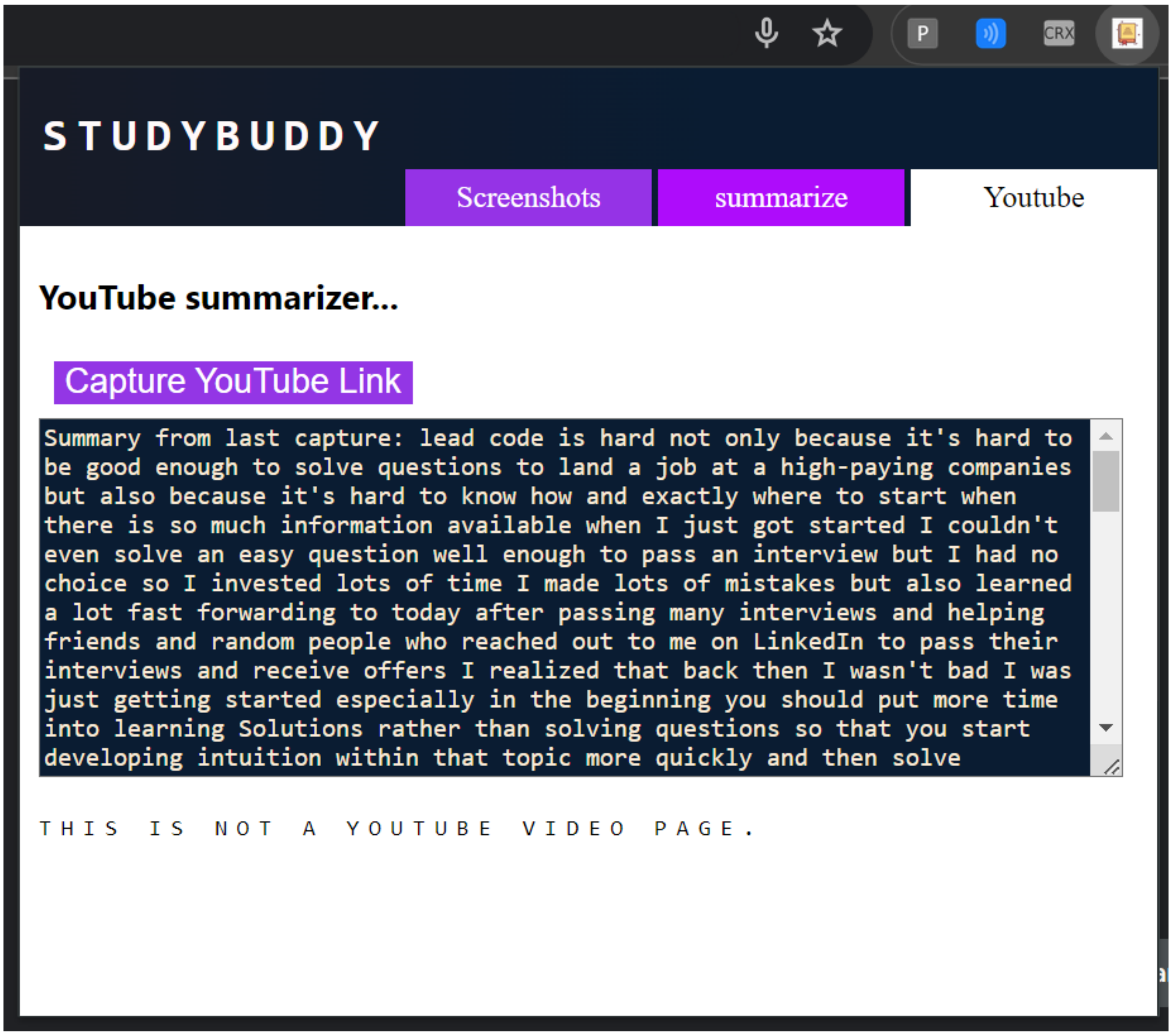
Summarize this Page

Make sure this is a article/blog for relevant summary

```
**Summary in 50 words:**

This article provides a comprehensive curriculum for learning artificial
intelligence (AI), covering mathematics, programming tools, machine
learning, deep learning, and large language models (LLMs). It includes
resources for beginners and advanced learners, along with practical tips
and platforms for practicing and staying updated.

**10 Major Points:**

1. Start with Python and PyTorch and Master Essential Features
2. Learn Deep Learning from Scratch and Implement Papers
3. Understand NLP and Build LLM Applications
4. Study Mathematics for Machine Learning and Understand Statistics
5. Explore DALL-E, Prompt Engineering, and LLMs
```

summarize page

Youtube page

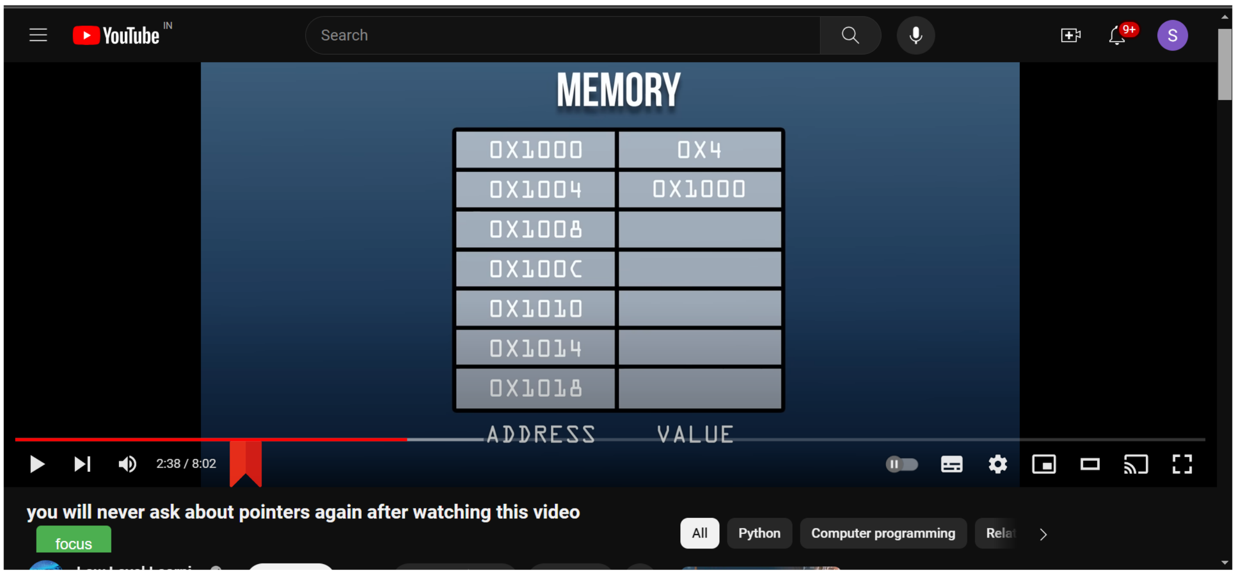


YouTube summarizer...

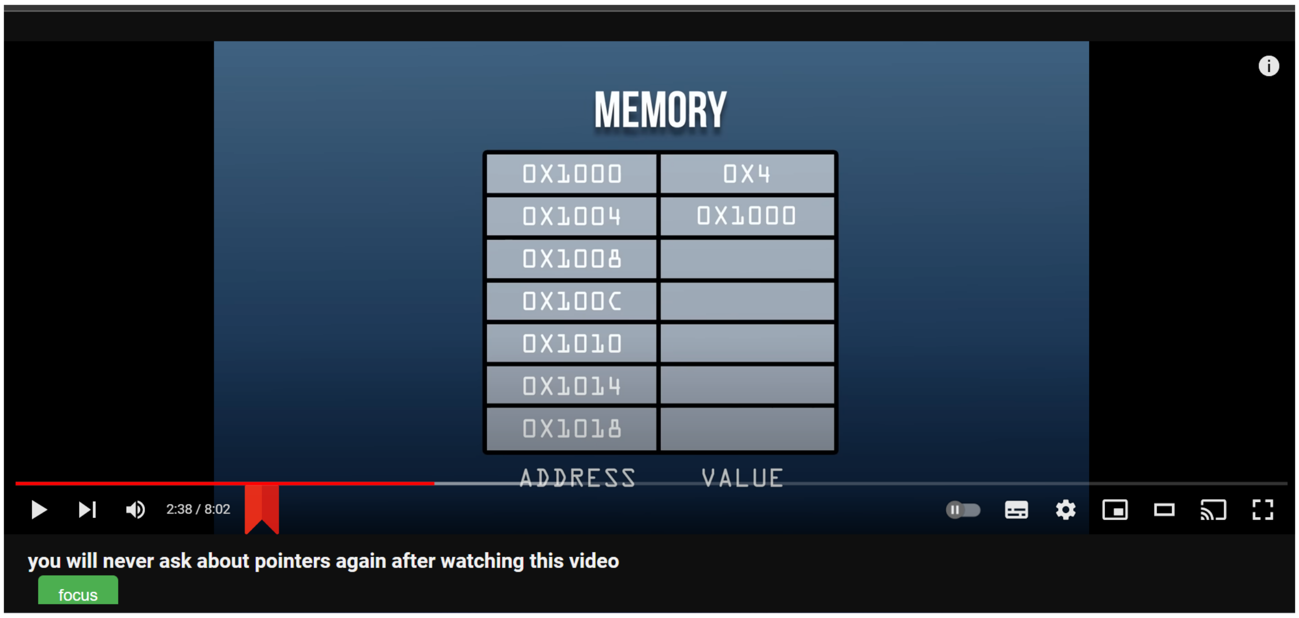
```
Summary from last capture: lead code is hard not only because it's hard to
be good enough to solve questions to land a job at a high-paying companies
but also because it's hard to know how and exactly where to start when
there is so much information available when I just got started I couldn't
even solve an easy question well enough to pass an interview but I had no
choice so I invested lots of time I made lots of mistakes but also learned
a lot fast forwarding to today after passing many interviews and helping
friends and random people who reached out to me on LinkedIn to pass their
interviews and receive offers I realized that back then I wasn't bad I was
just getting started especially in the beginning you should put more time
into learning Solutions rather than solving questions so that you start
developing intuition within that topic more quickly and then solve
```

THIS IS NOT A YOUTUBE VIDEO PAGE.

When opens any youtube video

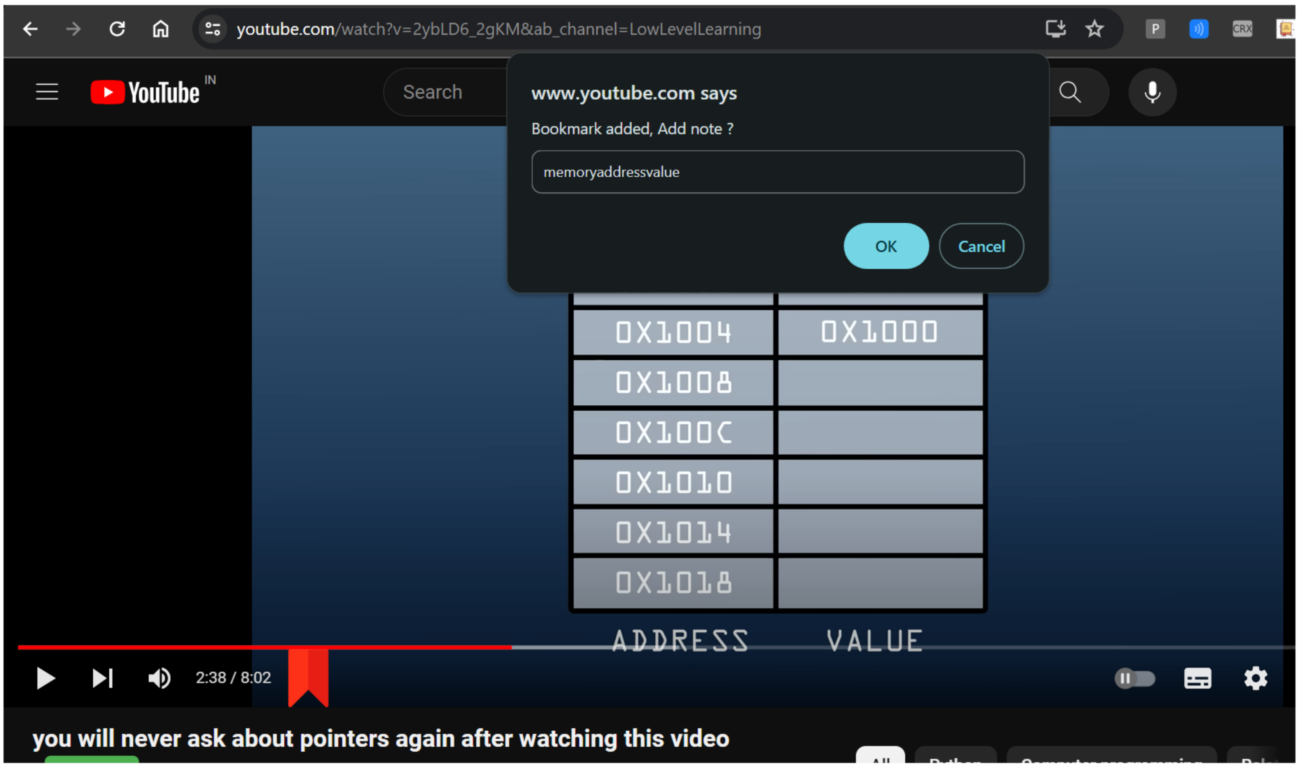


When clicks on focus mode other elements are hidden

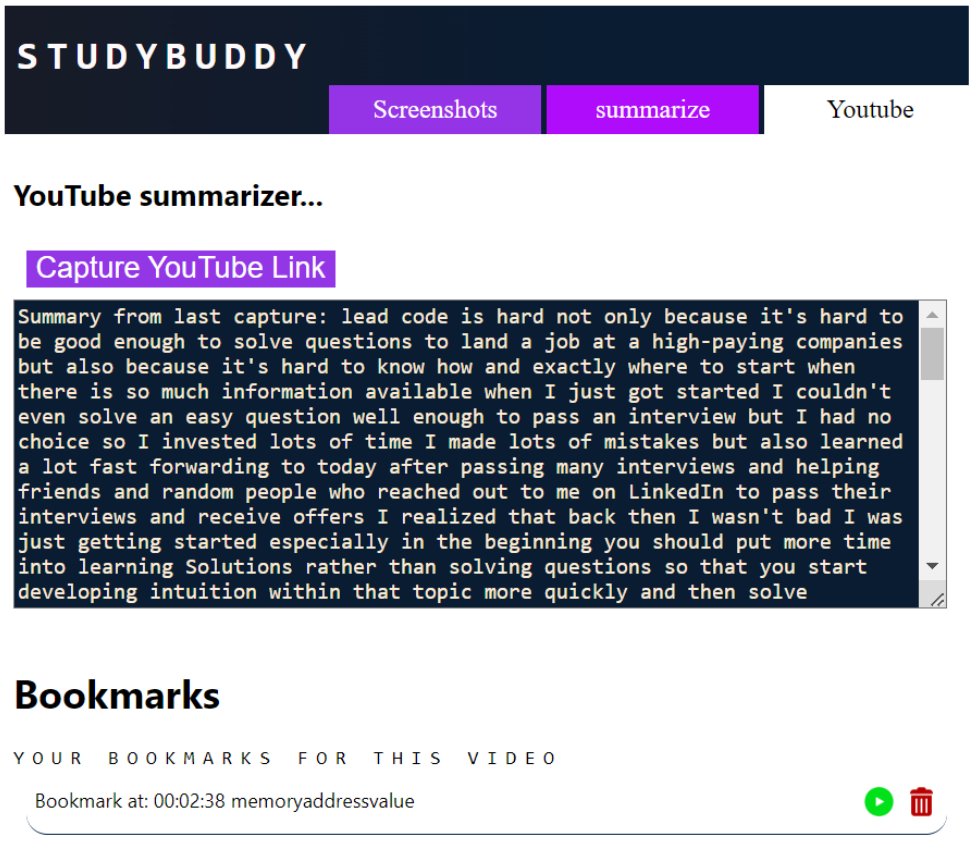


-no searchbar
-no comments
-no related videos list
Again clicking on the button unhides the distractions

When user clicks on bookmark icon


youtube.com/watch?v=2ybLD6_2gKM&ab_channel=LowLevelLearning
YouTube
Search
www.youtube.com says
Bookmark added, Add note ?
memoryaddressvalue
OK
Cancel
0X1004 0X1000
0X1008
0X100C
0X1010
0X1014
0X1018
ADDRESS VALUE
2:38 / 8:02
you will never ask about pointers again after watching this video

Bookmark gets added



YouTube summarizer...

Capture YouTube Link

```
Summary from last capture: lead code is hard not only because it's hard to
be good enough to solve questions to land a job at a high-paying companies
but also because it's hard to know how and exactly where to start when
there is so much information available when I just got started I couldn't
even solve an easy question well enough to pass an interview but I had no
choice so I invested lots of time I made lots of mistakes but also learned
a lot fast forwarding to today after passing many interviews and helping
friends and random people who reached out to me on LinkedIn to pass their
interviews and receive offers I realized that back then I wasn't bad I was
just getting started especially in the beginning you should put more time
into learning Solutions rather than solving questions so that you start
developing intuition within that topic more quickly and then solve
```

Bookmarks

YOUR BOOKMARKS FOR THIS VIDEO

Bookmark at: 00:02:38 memoryaddressvalue

Coding

Screenshot tab

Dashboard.html

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <script  type="module" src="jspdf.min.js"></script>
    <link rel="stylesheet" href="style.css">
    <script type="module" src="dashboard.js"></script>

    <style>
      button {
          color: white;
          background-color: #9336e5;
          border: none;
          border-radius: 3px;
          margin: 5px;
          border: 3px solid white;
          font-size: 18px;
      }
      button:hover {
          cursor: pointer;
      }
      #button-container {
          float: right;
          margin-top: -40px;
          margin-right: -25px;
      }
      #local-urls-scanned-count {
          border-radius: 50%;
          width: 310px;
          height: 310px;
          border: 30px solid #9336e5;
          text-align: center;
          font-family: "THICCBOI-Bold";
          line-height: 310px;
          margin: auto;
      }
      #url-count {
          display: inline-block;
          vertical-align: middle;
      }
      #urls-scanned-sub-text {
          font-family: "THICCBOI";
          font-size: 36px;
          color: #717171;
          text-align: center;
          margin-top: 20px;
      }
      .sharing-button {
          background-color: transparent;
          border: none;
          padding: 0;
      }
      .button-img {
          width: 1.75rem;
          height: 1.75rem;
      }
      .statistic {
          text-align: center;
          margin-top: 30px;
          margin-bottom: 30px;
          width: 242px;
          height: 131px;
      }
      .statistic-header {
          font-size: 16px;
          font-family: "THICCBOI-bold";
          color: #717171;
      }
      .statistic-value {
          font-family: "THICCBOI";
          color: #222222;
          margin-top: 0;
      }

  </style>
</head>
<body>
    <header>
        <div id="header">
          <!--<h1 data-value="studyBUddy! " class="title">studyBuddy</h1>-->
          <div id="header-left">
            <h1 data-value="ChromeXtnd" class="title">studyBuddy</h1>
          </div>
          <div id="header-right">
            <ul>
              <li id="dashboard-tab" class="tab tab-selected">
                <a href="dashboard.html">Screenshots</a>
              </li>
              <li id="earn-tab" class="tab">
                <a href="websum.html">summarize</a>
              </li>
              <li id="about-tab" class="tab">
                <a href="youtube.html">Youtube</a>
              </li>
            </ul>
          </div>
        </div>
      </header>
     <div style="padding: 10px;">
      <div style="background-color:beige;">
        <h1>Enter name of pdf</h1>
        <input type="text" id="pdfN" size="50" >
        <button id="capture">Capture Screenshot</button>
        <textarea id="inputText" rows="15" cols="75" resize="none" style="background-color:rgba(10,28,50,1);
color:antiquewhite" placeholder="TAKE A NOTE"></textarea><br>

        <button id="downloadAll">Download PDF</button>
        <button id="deleteButton">Delete Recent Screenshot</button>
        <button id="clearAll">clear all Screenshot</button>
        <br>
        </div>
    </div>
</body>
</html>
```

Dashboard.js

```
//importing the summaryof youtube from chrome storage
//var youtubesum = chrome.storage.sync.get(['summary']);

document.addEventListener('DOMContentLoaded', function () {
  const captureButton = document.getElementById('capture');
  const downloadAllButton = document.getElementById('downloadAll');
  const pdfNameInput = document.getElementById('pdfN');
  const clearAllButton = document.getElementById('clearAll'); // New button
  const deleteButton = document.getElementById('deleteButton'); // Define deleteButton
  const textArea = document.getElementById('inputText'); // Reference to the textarea
```

```
  //let txt;
  let tabTitle;
  let savedText = '';
  let screenshotCounter = 0;
  let screenshots = [];
  let pdfFileName = 'screenshots.pdf';
  let youtubesummary=''
  chrome.storage.sync.get(['summary'], function(result) {
    if (result.summary) {
        //document.getElementById('ytsum').innerText = "Summary from last capture: " + result.summary;
        youtubesummary+=result.summary;
        return youtubesummary;
        //console.log("indise result sum",youtubesummary);
        //txt=senddata(youtubesummary);
    } else {
        document.getElementById('ytsum').innerText = "No summary available.";
    }

});
// function senddata(data){
//   return data;
// }

  const screenshotCountElement = document.createElement('span');
  screenshotCountElement.textContent = 'Screenshots taken: 0';
  document.body.appendChild(screenshotCountElement);
  chrome.storage.sync.get(['savedText'], function (result) {
    if (result.savedText) {
      textArea.value = result.savedText;
      savedText = result.savedText;
    }
  });
  textArea.addEventListener('input', function () {
    const text = textArea.value.trim();
    savedText = text;
    // Save the text to Chrome's storage
    chrome.storage.sync.set({ savedText: text });
  });
  function updateScreenshotCount() {
    screenshotCountElement.textContent = `Screenshots taken: ${screenshotCounter}`;
  }
  chrome.storage.local.get(['screenshots'], (storedData) => {
    if (storedData.screenshots) {
      screenshotCounter = storedData.screenshots.length;
      screenshots = storedData.screenshots;
      updateScreenshotCount();
    }
  });
  captureButton.addEventListener('click', () => {
    chrome.tabs.captureVisibleTab(null, { format: 'png' }, (dataUrl) => {
      screenshotCounter++;
      updateScreenshotCount();
      screenshots.push(dataUrl);
      chrome.storage.local.set({ screenshots }, () => {
        console.log('Screenshots saved to storage');
      });
      if (screenshotCounter === 50) {
        captureButton.disabled = true;
      }
    });
  });

  function captureTitleAndAddToPDF(pdfDocument, callback) {
    chrome.tabs.query({ active: true, currentWindow: true }, function (tabs) {
      const currentTab = tabs[0];
       tabTitle = currentTab.title;
      pdfDocument.setFont('helvetica', 'bold');
      pdfDocument.setFontSize(40);
      // pdfDocument.text(20, 20, tabTitle);
      pdfDocument.setFont('helvetica', 'normal');
      pdfDocument.setFontSize(12);
      callback(tabTitle); // Call the callback function with the tab title
    });
  }
  downloadAllButton.addEventListener('click', () => {
    const text = savedText.trim()+youtubesummary;
    // if (screenshotCounter === 0) {
    //   console.log('No screenshots captured yet.');
    //   return;
    // }
    const pdfDocument = new jsPDF();
    if (text !== '' && screenshots.length == 0) {
      const doc = new jsPDF();
      const textLines = doc.splitTextToSize(text, doc.internal.pageSize.width - 20);
      let cursor = 10;
      textLines.forEach(line => {
        if (cursor > 280) {
          doc.addPage();
          cursor = 10;
        }
        doc.text(10, cursor, line);
        cursor += 10; // Increase cursor position for next line
      });
      doc.save('generated_pdf.pdf');
    }    else if (screenshots.length > 0 && text !== '') {
      captureTitleAndAddToPDF(pdfDocument, (tabTitle) => {
        function addScreenshotsToPdf() {
          let currentIndex = 0;
          while (currentIndex < screenshots.length) {
            const pageWidth = pdfDocument.internal.pageSize.width;
            const pageHeight = pdfDocument.internal.pageSize.height;
            const imageHeight = (pageHeight - 20) / 2;
            const screenshotDataUrl = screenshots[currentIndex];
            pdfDocument.rect(5, 5, pageWidth - 10, imageHeight, 'S');
            pdfDocument.addImage(screenshotDataUrl, 'PNG', 10, 10, pageWidth - 25, imageHeight - 15);
            if (currentIndex + 1 < screenshots.length) {
              const secondScreenshotDataUrl = screenshots[currentIndex + 1];
              pdfDocument.rect(5, imageHeight + 15, pageWidth - 10, imageHeight, 'S');
              pdfDocument.addImage(secondScreenshotDataUrl, 'PNG', 10, imageHeight + 20, pageWidth - 25, imageHeight - 15);
              currentIndex += 2;
            } else {
              currentIndex += 1;
            }
            if (currentIndex < screenshots.length) {
              pdfDocument.addPage();
            }
          }
        }
        addScreenshotsToPdf();
        // Calculate total number of pages used by screenshots
        const pagesUsedByScreenshots = pdfDocument.internal.getNumberOfPages();
        // Add text from the next page onwards
        for (let i = pagesUsedByScreenshots + 1; i <= pagesUsedByScreenshots + 1; i++) {
          pdfDocument.addPage();
          const textLines = pdfDocument.splitTextToSize(text, pdfDocument.internal.pageSize.width - 20);
          let cursor = 10;
          textLines.forEach(line => {
            if (cursor > 280) {
              pdfDocument.addPage();
              cursor = 10;
            }
            pdfDocument.text(10, cursor, line);
            cursor += 10; // Increase cursor position for next line
          });
        }
        let userInputName = pdfNameInput.value.trim();
        if (userInputName === '') {
          // If user doesn't enter a PDF name, use the current tab title
          userInputName = tabTitle+'.pdf';
          //userInputName = 'usb.pdf';
        }
        pdfFileName = userInputName || pdfFileName;
        const pdfDataUri = pdfDocument.output('datauristring');
        screenshots.push(pdfDataUri);
        chrome.storage.local.set({ screenshots }, () => {
          console.log('Screenshots updated in storage');
        });
        const downloadLink = document.createElement('a');
        downloadLink.href = pdfDataUri;
        downloadLink.download = pdfFileName;
        //downloadLink.setAttribute('download', 'pdfs/' + pdfFileName);
        downloadLink.click();
        chrome.storage.local.remove(['screenshots'], () => {
          console.log('Screenshots cleared from storage');
          screenshotCounter = 0; // Reset the counter to zero
          updateScreenshotCount(); // Update the counter display
        });
      });
    }
    else if (screenshots.length > 0 && text == '') {
```

```
      captureTitleAndAddToPDF(pdfDocument, (tabTitle) => {
        function addScreenshotsToPdf() {
          let currentIndex = 0;
          while (currentIndex < screenshots.length) {
            const pageWidth = pdfDocument.internal.pageSize.width;
            const pageHeight = pdfDocument.internal.pageSize.height;
            const imageHeight = (pageHeight - 20) / 2;
            const screenshotDataUrl = screenshots[currentIndex];
            pdfDocument.rect(5, 5, pageWidth - 10, imageHeight, 'S');
            pdfDocument.addImage(screenshotDataUrl, 'PNG', 10, 10, pageWidth - 25, imageHeight - 15);
            if (currentIndex + 1 < screenshots.length) {
              const secondScreenshotDataUrl = screenshots[currentIndex + 1];
              pdfDocument.rect(5, imageHeight + 15, pageWidth - 10, imageHeight, 'S');
              pdfDocument.addImage(secondScreenshotDataUrl, 'PNG', 10, imageHeight + 20, pageWidth - 25, imageHeight - 15);
              currentIndex += 2;
            } else {
              currentIndex += 1;
            }
            if (currentIndex < screenshots.length) {
              pdfDocument.addPage();
            }
          }
        }
        addScreenshotsToPdf();
        let userInputName = pdfNameInput.value.trim();
        if (userInputName === '') {
          // If user doesn't enter a PDF name, use the current tab title
          userInputName = tabTitle+'.pdf';
          //userInputName = 'usb.pdf';
        }
        pdfFileName = userInputName || pdfFileName;
        const pdfDataUri = pdfDocument.output('datauristring');
        screenshots.push(pdfDataUri);
        chrome.storage.local.set({ screenshots }, () => {
          console.log('Screenshots updated in storage');
        });
        const downloadLink = document.createElement('a');
        downloadLink.href = pdfDataUri;
        downloadLink.download = pdfFileName;
        //downloadLink.setAttribute('download', 'pdfs/' + pdfFileName);
        downloadLink.click();
        chrome.storage.local.remove(['screenshots'], () => {
          console.log('Screenshots cleared from storage');
          screenshotCounter = 0; // Reset the counter to zero
          updateScreenshotCount(); // Update the counter display
        });
      });
    }
    else {
      console.log('No screenshots or text available.');
    }
  });
  clearAllButton.addEventListener('click', () => {
    screenshots = [];
    screenshotCounter = 0;
    updateScreenshotCount();
    captureButton.disabled = false;
    chrome.storage.local.set({ screenshots }, () => {
      console.log('All screenshots cleared from storage');
    });
  });
  deleteButton.addEventListener('click', () => {
    if (screenshotCounter > 0) {
      screenshots.pop();
      screenshotCounter--;
      updateScreenshotCount();
      if (screenshotCounter < 50) {
        captureButton.disabled = false;
      }
    } else {
      console.log('No screenshots to delete.');
    }
    chrome.storage.local.set({ screenshots }, () => {
      console.log('Screenshots updated in storage');
    });
  });
});
// ---- ---- ---- ---- ---- //
const letters = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ';
let interval = null;
document.querySelector('.title').onmouseover = (event) => {
  let iteration = 0;
  clearInterval(interval);
  interval = setInterval(() => {
    event.target.innerText = event.target.innerText
      .split('')
      .map((letter, index) => {
        if (index < iteration) {
          return event.target.dataset.value[index];
        }
        return letters[Math.floor(Math.random() * 26)];
      })
      .join('');
    if (iteration >= event.target.dataset.value.length) {
      clearInterval(interval);
    }
    iteration += 1 / 3;
  }, 50);
};
```

Summarize tab

Websum.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <link rel="stylesheet" href="style.css">
    <script type="module" src="websum.js"></script>
    <style>
      button {
          color: white;
          background-color: #9336e5;
          border: none;
          border-radius: 3px;
          margin: 5px;
          border: 3px solid white;
          font-size: 18px;
      }
      button:hover {
          cursor: pointer;
      }
      #button-container {
          float: right;
          margin-top: -40px;
          margin-right: -25px;
      }
      #local-urls-scanned-count {
          border-radius: 50%;
          width: 310px;
          height: 310px;
          border: 30px solid #9336e5;
          text-align: center;
          font-family: "THICCBOI-Bold";
          line-height: 310px;
          margin: auto;
      }
      #url-count {
          display: inline-block;
          vertical-align: middle;
      }
      #urls-scanned-sub-text {
          font-family: "THICCBOI";
          font-size: 36px;
          color: #717171;
          text-align: center;
          margin-top: 20px;
      }
      .sharing-button {
          background-color: transparent;
          border: none;
          padding: 0;
      }
      .button-img {
          width: 1.75rem;
          height: 1.75rem;
      }
      .statistic {
          text-align: center;
          margin-top: 30px;
          margin-bottom: 30px;
          width: 242px;
          height: 131px;
      }
      .statistic-header {
          font-size: 16px;
          font-family: "THICCBOI-bold";
          color: #717171;
      }
      .statistic-value {
          font-family: "THICCBOI";
          color: #222222;
          margin-top: 0;
      }
  </style>
</head>
<body>
    <div>
      <header>
        <div id="header">
          <!--<h1 data-value="studyBUddy! " class="title">studyBuddy</h1>-->
          <div id="header-left">
            <h1 data-value="SUMMARIZE.. " class="title">studyBuddy</h1>
          </div>
          <div id="header-right">
            <ul>
              <li id="dashboard-tab" class="tab">
                <a href="dashboard.html">Screenshots</a>
              </li>
              <li id="earn-tab" class="tab tab-selected">
                <a href="websum.html">summarize</a>
              </li>
              <li id="about-tab" class="tab">
                <a href="youtube.html">Youtube</a>
              </li>
            </ul>
          </div>
        </div>
      </header>
<div>
  <div style="padding: 10px;">
    <div id="pagesum">
        <h1 class="custom-heading">Summarize this Page</h1>
        <p style="font-size: medium; font-family:cursive">Make sure this is a article/blog for relavent summary</p>
        <textarea id="gensum" rows="5" cols="75" style="background-color:rgba(10,28,50,1);
color:antiquewhite">summary..</textarea>
        <button id="websum">summarize page</button>
      </div>
    </div>
  </div>

</body>
</html>
```

Websum.js

```
let capturedText = "";

document.getElementById("websum").addEventListener("click", () => {
    chrome.tabs.query({ active: true, currentWindow: true }, (tabs) => {
        chrome.scripting.executeScript({
            target: { tabId: tabs[0].id },
            function: captureText
        });
    });
});
function captureText() {
    const text = document.body.innerText;
    chrome.runtime.sendMessage({ method: "getText", data: text });
}
chrome.runtime.onMessage.addListener((request, sender, sendResponse) => {
    if (request.method === "getText") {
        capturedText = request.data;
        //document.getElementById("summary").textContent = capturedText;

        // Send the captured text to the Flask app
        sendTextToFlask(capturedText);
    }
});
function sendTextToFlask(text) {
    fetch('http://127.0.0.1:5000/capturetext', {
        method: 'POST',
        headers: {
            'Content-Type': 'application/json',
        },
        body: JSON.stringify({ text: text }),
    })
    .then(response => {
        if (!response.ok) {
            throw new Error('Network response was not ok');
        }
        return response.json();
    })
    .then(data => {
        console.log('Response from Flask app:', data);
        // Update the summary in the HTML after 5 seconds
        setTimeout(() => {
            document.getElementById("gensum").textContent = data.answer;
        }, 5000);
    })
    .catch(error => {
        console.error('There was a problem with your fetch operation:', error);
    });
}
const letters = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ';
let interval = null;
document.querySelector('.title').onmouseover = (event) => {
  let iteration = 0;
  clearInterval(interval);
  interval = setInterval(() => {
    event.target.innerText = event.target.innerText
      .split('')
      .map((letter, index) => {
        if (index < iteration) {
          return event.target.dataset.value[index];
        }
        return letters[Math.floor(Math.random() * 26)];
      })
      .join('');
    if (iteration >= event.target.dataset.value.length) {
      clearInterval(interval);
    }
    iteration += 1 / 3;
  }, 50);
};
```

Youtube.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <link rel="stylesheet" href="style.css">
    <script type="module" src="youtube.js"></script>
    <style>
      button {
          color: white;
          background-color: #9336e5;
          border: none;
          border-radius: 3px;
          margin: 5px;
          border: 3px solid white;
          font-size: 18px;
      }
      button:hover {
          cursor: pointer;
      }
      #button-container {
          float: right;
          margin-top: -40px;
          margin-right: -25px;
      }
      #local-urls-scanned-count {
          border-radius: 50%;
          width: 310px;
          height: 310px;
          border: 30px solid #9336e5;
          text-align: center;
          font-family: "THICCBOI-Bold";
          line-height: 310px;
          margin: auto;
      }
      #url-count {
          display: inline-block;
          vertical-align: middle;
      }
      #urls-scanned-sub-text {
          font-family: "THICCBOI";
          font-size: 36px;
          color: #717171;
          text-align: center;
          margin-top: 20px;
      }
      .sharing-button {
          background-color: transparent;
          border: none;
          padding: 0;
      }
      .button-img {
          width: 1.75rem;
          height: 1.75rem;
      }
      .statistic {
          text-align: center;
          margin-top: 30px;
          margin-bottom: 30px;
          width: 242px;
          height: 131px;
      }
      .statistic-header {
          font-size: 16px;
          font-family: "THICCBOI-bold";
          color: #717171;
      }
      .statistic-value {
          font-family: "THICCBOI";
          color: #222222;
          margin-top: 0;
      }
  </style>
</head>
<body>
  <header>
    <div id="header">
      <!--<h1 data-value="studyBUddy! " class="title">studyBuddy</h1>-->
      <div id="header-left">
```

```html
        <h1 data-value="studyBUddy! " class="title">studyBuddy</h1>
      </div>
      <div id="header-right">
        <ul>
          <li id="dashboard-tab" class="tab" id="earn-tab" class="tab">
            <a href="dashboard.html">Screenshots</a>
          </li>
          <li id="earn-tab" class="tab">
            <a href="websum.html">summarize</a>
          </li>
          <li id="about-tab" class="tab tab-selected">
            <a href="youtube.html">Youtube</a>
          </li>
        </ul>
      </div>
    </div>
  </header>
  <div style="padding: 10px;">
      <div id="cont">
        <h2>YouTube summarizer...</h2>
        <button id="captureButton">Capture YouTube Link</button>
        <textarea id="result" rows="12" cols="75" style="background-color:rgba(10,28,50,1);
color:antiquewhite"></textarea><br>
       <!-- <a href="http://127.0.0.1:5000/" target="_blank">http://127.0.0.1:5000/</a>-->
      </div>
<br>
      <div class="container">
        <div>
          <h1>Bookmarks</h1>
          <div class="title" style="color: black;">Your bookmarks for this video</div>

          <div class="bookmarks" id="bookmarks"></div>
        </div>
      </div>
    </div>
</body>
</html>
```

Youtube.js

```javascript

document.getElementById('captureButton').addEventListener('click', function () {
  chrome.tabs.query({ active: true, currentWindow: true }, function (tabs) {
      const currentTab = tabs[0];
      if (isYouTubeTab(currentTab)) {
          const youtubeLink = currentTab.url;
          // Send the YouTube link to the Flask app
          fetch('http://127.0.0.1:5000/receive-link', {
              method: 'POST',
              headers: {
                  'Content-Type': 'application/json',
              },
              body: JSON.stringify({ youtube_link: youtubeLink }),
          })
          .then(response => response.text())
          .then(data => {
              console.log(data);
              // Fetch the summary and display it below the button
              fetch('http://127.0.0.1:5000/')
                  .then(response => response.json())
                  .then(summaryData => {
                      const summary = summaryData["summary of this is "]; // Access the summary text using the appropriate key
                      if (summary) {
                          // Display summary
                          document.getElementById('result').innerText = "YouTube Link captured! Summary: " + summary;
                          // Save summary in Chrome storage
                          chrome.storage.sync.set({ 'summary': summary }, function() {
                              console.log('Summary saved in Chrome storage.');
                          });
                      } else {
                          document.getElementById('result').innerText = "Failed to retrieve summary.";
                      }
                  })
                  .catch(error => {
                      console.error('Error fetching summary:', error);
                      document.getElementById('result').innerText = 'Error fetching summary.';
                  });
          })
          .catch(error => {
              console.error('Error capturing YouTube link:', error);
              document.getElementById('result').innerText = 'Error capturing YouTube link.';
          });
      } else {
          document.getElementById('result').innerText = "Works only for YouTube tabs.";
      }
  });
});
// Function to check if the tab is a YouTube tab
function isYouTubeTab(tab) {
  return tab.url.startsWith("https://www.youtube.com/") || tab.url.startsWith("http://www.youtube.com/");
}
// Retrieve and display summary from Chrome storage when popup is opened
document.addEventListener('DOMContentLoaded', function () {
  chrome.storage.sync.get(['summary'], function(result) {
      if (result.summary) {
          document.getElementById('result').innerText = "Summary from last capture: " + result.summary;
      } else {
          document.getElementById('result').innerText = "No summary available.";
      }
  });
});
async function getActiveTabURL() {
    const tabs = await chrome.tabs.query({
        currentWindow: true,
        active: true
    });

    return tabs[0];
  }

  const addNewBookmark = (bookmarks, bookmark) => {
    const bookmarkTitleElement = document.createElement('div');
    const controlsElement = document.createElement('div');
    const newBookmarkElement = document.createElement('div');

    bookmarkTitleElement.textContent = bookmark.desc;
    bookmarkTitleElement.className = 'bookmark-title';
    controlsElement.className = 'bookmark-controls';

    setBookmarkAttributes('play', onPlay, controlsElement);
    setBookmarkAttributes('delete', onDelete, controlsElement);

    newBookmarkElement.id = 'bookmark-' + bookmark.time;
    newBookmarkElement.className = 'bookmark';
    newBookmarkElement.setAttribute('timestamp', bookmark.time);

    newBookmarkElement.appendChild(bookmarkTitleElement);
    newBookmarkElement.appendChild(controlsElement);
    bookmarks.appendChild(newBookmarkElement);
  };

  const viewBookmarks = (currentBookmarks = []) => {
    const bookmarksElement = document.getElementById('bookmarks');
    bookmarksElement.innerHTML = '';

    if (currentBookmarks.length > 0) {
      for (let i = 0; i < currentBookmarks.length; i++) {
        const bookmark = currentBookmarks[i];
        addNewBookmark(bookmarksElement, bookmark);
      }
    } else {
      bookmarksElement.innerHTML = '<i class="row">No bookmarks to show</i>';
    }

    return;
  };

  const onPlay = async (e) => {
    const bookmarkTime = e.target.parentNode.parentNode.getAttribute('timestamp');
    const activeTab = await getActiveTabURL();

    chrome.tabs.sendMessage(activeTab.id, {
      type: 'PLAY',
      value: bookmarkTime,
    });
  };

  const onDelete = async (e) => {
    const activeTab = await getActiveTabURL();
    const bookmarkTime = e.target.parentNode.parentNode.getAttribute('timestamp');
    const bookmarkElementToDelete = document.getElementById(
      'bookmark-' + bookmarkTime
    );

    bookmarkElementToDelete.parentNode.removeChild(bookmarkElementToDelete);

    chrome.tabs.sendMessage(
      activeTab.id,
      {
        type: 'DELETE',
        value: bookmarkTime,
      },
      viewBookmarks
    );
  };

  const setBookmarkAttributes = (src, eventListener, controlParentElement) => {
    const controlElement = document.createElement('img');

    controlElement.src = 'images/' + src + '.png';
    controlElement.title = src;
    controlElement.addEventListener('click', eventListener);
    controlParentElement.appendChild(controlElement);
  };

  //DOMContentLoaded is fired when an HTML document has innitially been loaded
  document.addEventListener('DOMContentLoaded', async () => {
    const activeTab = await getActiveTabURL();
    const queryParameters = activeTab.url.split('?')[1];
    const urlParameters = new URLSearchParams(queryParameters);

    const currentVideo = urlParameters.get('v');

    if (activeTab.url.includes('youtube.com/watch') && currentVideo) {
      chrome.storage.sync.get([currentVideo], (data) => {
        const currentVideoBookmarks = data[currentVideo]
          ? JSON.parse(data[currentVideo])
          : [];

        viewBookmarks(currentVideoBookmarks);
      });
    } else {
      const container = document.getElementsByClassName('container')[0];

      container.innerHTML =
        '<div class="title" style="color:black">This is not a youtube video page.</div>';
    }
  });
```

Manifest.json

```json
{
  "manifest_version": 3,
  "name": "1integrate",
  "version": "1.0",
  "description": "Capture YouTube tab link",
  "permissions": ["tabs","storage", "activeTab", "downloads","scripting"],

  "icons":{
    "128":"images/logo.png"
  },
  "action": {
    "default_popup": "dashboard.html"

  },

  "content_scripts": [
    {
      "matches": ["https://www.youtube.com/*"],
      "js": ["content.js"]
    }
  ],
  "web_accessible_resources": [
    {
      "resources": [
        "images/bookmark.png"
      ],
      "matches": ["https://*.youtube.com/*"]
    }
  ],
  "background": {
    "service_worker": "background.js"
  },
  "options_page": "options.html"
}
```

Content.js

```javascript
function addButton() {
    const metadataElement = document.querySelector('yt-formatted-string.style-scope.ytd-watch-metadata');
    if (metadataElement && !document.getElementById('customButton')) {
        const buttonHTML = `<button id="customButton">focus</button>`;
        const tempContainer = document.createElement('div');
        tempContainer.innerHTML = buttonHTML;
        const button = tempContainer.firstElementChild;

        button.addEventListener('click', () => {
            // Toggle visibility of the specific <div> with class 'style-scope ytd-watch-flexy'
            //const secondaryDiv = document.getElementById('columns');
            //const menuRenderer = document.getElementById('comments');
            const menuRenderer = document.querySelector('ytd-item-section-renderer#sections[initial-count="2"].style-
scope.ytd-comments[section-identifier="comment-item-section"]');

            if (menuRenderer) {
                if (menuRenderer.style.display === 'none') {
                    menuRenderer.style.display = 'block';
                } else {
                    menuRenderer.style.display = 'none';
                }
            }

            const secondaryDiv = document.getElementById('secondary');

            if (secondaryDiv) {
                if (secondaryDiv.style.display === 'none') {
                    secondaryDiv.style.display = 'block';
                } else {
                    secondaryDiv.style.display = 'none';
                }
            }
```

```
          const ythead = document.getElementById('masthead-container');

          if (ythead) {
              if (ythead.style.display === 'none') {
                  ythead.style.display = 'block';
              } else {
                  ythead.style.display = 'none';
              }
          }

          const desc = document.getElementById('bottom-row');

          if (desc) {
              if (desc.style.display === 'none') {
                  desc.style.display = 'block';
              } else {
                  desc.style.display = 'none';
              }
          }

      });

      // Add CSS style to the button
      button.style.cssText = `
      background-color: #4CAF50;
      border: none;
      color: white;
      padding: 10px 20px;
      text-align: center;
      text-decoration: none;
      display: inline-block;
      font-size: 16px;
      margin-left: 10px;
      cursor: pointer;
      border-radius: 5px;
      transition-duration: 0.4s;
    `;

      // Add button to the DOM
      metadataElement.parentNode.insertBefore(button, metadataElement.nextSibling);
  }
}

// Function to handle mutations
function mutationHandler(mutationsList, observer) {
  for (let mutation of mutationsList) {
      if (mutation.type === 'childList') {
          addButton();
      }
  }
}

// Observer configuration
const observerConfig = { childList: true, subtree: true };

// Create a new mutation observer
const observer = new MutationObserver(mutationHandler);

// Start observing the document
observer.observe(document.body, observerConfig);

// Initial action when the extension is loaded
addButton();

(() => {
  let youtubeLeftControls, youtubePlayer;
  let currentVideo = '';
  let currentVideoBookmarks = [];

  const fetchBookmarks = () => {
    return new Promise((resolve) => {
      chrome.storage.sync.get([currentVideo], (obj) => {
        resolve(obj[currentVideo] ? JSON.parse(obj[currentVideo]) : []);
      });
    });
  };

  const addNewBookmarkEventHandler = async () => {
    const currentTime = youtubePlayer.currentTime;
    const message = prompt(`Bookmark added, Add note ?`);
    const newBookmark = {
      time: currentTime,
      desc: `Bookmark at: ${getTime(currentTime)} ${message}`,
    };

    currentVideoBookmarks = await fetchBookmarks();

    chrome.storage.sync.set({
      [currentVideo]: JSON.stringify(
        [...currentVideoBookmarks, newBookmark].sort((a, b) => a.time - b.time)
      ),
    });
  };

  const newVideoLoaded = async () => {
    const bookmarkBtnExists =
      document.getElementsByClassName('bookmark-btn')[0];

    currentVideoBookmarks = await fetchBookmarks();

    if (!bookmarkBtnExists) {
      const bookmarkBtn = document.createElement('img');

      bookmarkBtn.src = chrome.runtime.getURL('images/bookmark.png');
      bookmarkBtn.className = 'ytp-button ' + 'bookmark-btn';
      bookmarkBtn.title = 'Click to bookmark current timestamp';

      youtubeLeftControls =
        document.getElementsByClassName('ytp-left-controls')[0];
      youtubePlayer = document.getElementsByClassName('video-stream')[0];

      youtubeLeftControls.appendChild(bookmarkBtn);
      bookmarkBtn.addEventListener('click', addNewBookmarkEventHandler);
    }
  };

  chrome.runtime.onMessage.addListener((obj, sender, response) => {
    const { type, value, videoId } = obj;

    if (type === 'NEW') {
      currentVideo = videoId;
      newVideoLoaded();
    } else if (type === 'PLAY') {
      youtubePlayer.currentTime = value;
    } else if (type === 'DELETE') {
      currentVideoBookmarks = currentVideoBookmarks.filter(
        (b) => b.time != value
      );
      chrome.storage.sync.set({
        [currentVideo]: JSON.stringify(currentVideoBookmarks),
      });

      response(currentVideoBookmarks);
    }
  });

  newVideoLoaded();
})();

const getTime = (t) => {
  var date = new Date(0);
  date.setSeconds(t);

  return date.toISOString().substr(11, 8);
};
```

Styles.css

```css
body {
    min-width: 600px;
    min-height: 500px;
    max-width: 600px;
    max-height: 500px;
    margin: 0;
    resize :none;
}
#header {
    background: rgb(24,27,39);
    background: linear-gradient(90deg, rgba(24,27,39,1) 0%, rgba(10,28,50,1) 60%);
    height: 83px;
    width: 100%;
}
#header a {
    text-decoration: none;
    color: white;
}
#header-left {
    float: left;
    padding-top: 5px;
    padding-left: 12px;
}
#header-right {
    float: right;
    text-align: right;
    margin-top: 53px;
}
#header-right ul {
    margin: 0;
    padding: 0;
}
.tab {
    display: inline-block;
    line-height: 30px;
    font-family: 'THICCBOI';
    font-size: 16px;
    min-width: 130px;
    height: 30px;
    color: white;
    text-align: center;
}
.tab-selected {
    color: black !important;
    background-color: white !important;
}
.tab-selected a {
    color: black !important;
}
#dashboard-tab {
    background-color: #9533e6;
}
#earn-tab {
    background-color: #ae0cfb;
}
#about-tab {
    background-color: #7c489e;
}
.title {
    font-family: 'Space Mono', monospace;

    text-transform: uppercase;
    letter-spacing: 5px;
    color: #fffafa;

    border-radius: 10px;
}

.bookmarks {
  margin: 5px 5px;
  padding: 3px;
}

.bookmark {
  display: flex;
  border-bottom-color: #00254d;
  border-bottom-width: 1px;
  border-bottom-style: solid;
  border-radius: 0.8rem;
  padding: 3px;
  padding-bottom: 7px;
  margin-bottom: 7px;
}

.bookmark-title {
  padding-left: 2px;
}

.bookmark-controls img {
  margin: 0 4px;
  width: 18px;
  height: 18px;
  cursor: pointer;
}
.bookmark-controls {
  flex: auto;
  text-align: right;
}
```

Secure Coding Practices

The project hosts no database, and no user instances of login or signup. Hence, there is no need for distinguished, secure coding practices.

Testing Approach

Unit Testing:

Definition:

Unit testing is the process of testing individual units or components of a software independently to ensure they function as expected.

Purpose in Secure Testing:

Authentication Module:

Test the authentication module independently to ensure that user credentials are verified securely.

Verify that password hashing and encryption techniques are properly implemented.

Authorization Checks:

Test authorization mechanisms to ensure that only authorized users have access to specific resources.

Confirm that roles and permissions are correctly enforced.

Input Validation:

Validate user inputs for various forms and fields to prevent common security vulnerabilities such as SQL injection or Cross-Site Scripting (XSS).

Session Management:

Test the session management module to prevent session fixation, session hijacking, or other related vulnerabilities.

Integration Testing:

Definition:

Integration testing is the process of testing the combined components of a system to verify they work together as intended.

Purpose in Secure Testing:

Secure Data Transmission:

Test the integration of components responsible for data transmission to ensure the use of secure protocols (HTTPS).

Verify the proper implementation of encryption during data transit.

Third-Party Integrations:

Integrates with third-party services (e.g., payment gateways, databases) ensure that these integrations are secure.

Check that data shared between your website and external services is handled securely.

E. FUTURE ENHANCEMENTS

Cross-Browser Compatibility:

The extension will be made available for other popular browsers such as Firefox, Safari, and Microsoft Edge, to cater to a wider audience.
Compatibility testing will be conducted to ensure seamless functionality across different browser environments.

Enhanced Screenshot Capture:

Introduce the ability to capture screenshots of specific areas of the webpage, providing users with more flexibility and precision.
Implement features like customizable selection tools and zoom functionality to enhance the screenshot capturing experience.
Improved Summarization Model:

Invest in refining and optimizing the summarization model used by the extension to provide more accurate and concise summaries of webpage content and YouTube videos.
Explore advanced natural language processing techniques and machine learning algorithms to improve the model's response quality and efficiency.

Text Highlighting in PDF Documents:

Integrate a feature to highlight important text segments within the final PDF documents generated by the extension.
Provide users with options to customize the highlight colors and intensity to suit their preferences and emphasize key information.

Download Document Format Options:

Offer users the choice to download the generated documents in PDF or DOC format, based on their requirements and preferences.
Implement conversion functionality to enable seamless conversion between different document formats without compromising quality.

Text and Image Manipulation Tools:

Introduce tools within the extension's interface to allow users to resize, reposition, and format text and images before finalizing the document download.
Provide intuitive drag-and-drop functionality and resizing handles to facilitate easy manipulation of content elements within the document.

These future enhancements aim to elevate the functionality and user experience of the Chrome extension, making it a versatile tool for capturing, summarizing, and organizing content from webpages and YouTube videos. By expanding compatibility, enhancing features, and introducing new capabilities, the extension will continue to meet the evolving needs of its users and remain competitive in the market.

F. BIBLIOGRAPHY

Chrome Extension Documentation:

https://developer.chrome.com/docs/extensions/get-started/tutorial/hello-world

Jspdf documentation
https://artskydj.github.io/jsPDF/docs/jsPDF.html

Flask Documentation:

https://flask.palletsprojects.com/en/3.0.x/