INTERNSHIP REPORT
ON
CHROME EXTENSIONS DEVELOPMENT

A Report submitted to the Rajiv Gandhi University of Knowledge Technologies in fulfillment of the degree of Bachelor of Technology in Computer Science.

By

Babanbai Babafakruddin
R170846

Under the supervision of

SHABANA SHAIK
Assistant Professor
Computer Science Engineering

Idupulapaya, Vempalli,
Kadapa - 516330, Andhra Pradesh, India

CERTIFICATE

This is to certify that the report entitled **"Long Term Internship on Chrome Extensions Development"** submitted by Babanbai Babafakruddin (R170846) in fulfillment of the requirements for the award of Bachelor of Technology in Computer Science is a bonafide work carried out by him under my supervision and guidance. The report has not been submitted previously in part or in full to this or any other University or Institution for the award of any degree or diploma.

Shabana Shaik,
Project Internal Guide,
CSE,
RGUKT, R.K Valley.

N.Satyanandaram
Head of the Department,
CSE,
RGUKT, R.K Valley.

DECLARATION

I Babanbai Babafakruddin hereby declare that this report entitled “Long Term Internship on Chrome extensions development” submitted by me under the guidance and supervision of Shabana Shaik is a bonafide work. I also declare that it has not been submitted previously in part or in full to this University or other University or Institution for the award of any degree or diploma.

Date: 05-05-2023 Babanbai Babafakruddin
Place: RK Valley (R170846)

ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of any task would be incomplete without the mention of the people who made it possible and who's constant guidance and encouragement crown all the efforts success.

We are extremely grateful for the confidence bestowed in us and entrusting our project entitled "Development of Chrome Extensions". At this juncture we feel deeply honoured in expressing our sincere thanks to the GSTECH Technology Pvt Ltd team for the constant support to complete the projects tasks at right time with the great knowledge.

We would like to express my sincere gratitude to Ms.S.Shabana, my project guide for valuable suggestions and keen interest throughout the progress of our project.

We are grateful to Mr. N. Satyanandram HOD CSE, for providing excellent computing facilities and congenial atmosphere for progressing our project.

At the outset, we would like to thank Rajiv Gandhi University of Knowledge Technologies (RGUKT), for providing all the necessary resources and support for the successful completion of my course work

TABLE OF CONTENTS

ABSTRACT

1. Chrome Extensions Development:-

Chrome extensions are software programs that run within the Google Chrome browser and can modify its behavior, add functionality, and improve user experience. To create a Chrome extension, developers need to know web technologies like HTML, CSS, and JavaScript, and be familiar with the Chrome extension API. The process involves creating a manifest file, defining the user interface, and implementing the extension's logic using JavaScript. Chrome extensions can be published to the Chrome Web Store for users to download and install, allowing developers to create innovative tools that enhance browsing experience.

INTRODUCTION

Chrome Extensions Development: -

Chrome extensions are a powerful way to customize the user experience in the Google Chrome browser. They allow developers to modify the browser's behavior, add new features, and enhance the user interface. With millions of users worldwide, Chrome extensions offer an excellent opportunity for developers to create innovative tools that can improve browsing experience.

In this process, developers create a manifest file that describes the extension's properties, define the user interface, and implement the extension's logic using JavaScript. The final product can be published to the Chrome Web Store, where users can download and install the extension. In this way, Chrome extensions provide a powerful tool for developers to create new and exciting functionality within the Chrome browser.

Next, developers define the user interface of the extension, which can include browser actions, page actions, popups, options pages, and content scripts. Browser actions are buttons that appear on the browser toolbar and allow users to interact with the extension. Page actions are similar to browser actions, but they appear only when specific web pages are loaded. Popups are small windows that appear when users click on a browser or page action. Options pages allow users to configure the extension's settings. Content scripts are used to interact with web pages and manipulate their content.

Finally, developers implement the extension's logic using JavaScript. They can use the Chrome extension API to access the browser's functionality, including tabs, windows, cookies, storage, and notifications. They can also interact with web pages and manipulate their content using the Document Object Model (DOM).

Technologies Used

Chrome extensions are developed using web technologies such as HTML, CSS, and JavaScript. Developers also use the Chrome extension API, which provides access to the browser's functionality and allows them to interact with web pages and manipulate their content.

HTML is used to create the structure of the extension, including its user interface elements. CSS is used to style these elements, defining their visual appearance and layout. JavaScript is used to add interactivity and dynamic behavior to the extension, implementing its logic and responding to user input.

The Chrome extension API provides developers with a range of functionalities, including:

- Tabs and windows: Developers can use the API to create, modify, and interact with browser tabs and windows.

- Cookies and storage: Developers can use the API to read and write data to cookies and local or synced storage.

- Notifications: Developers can use the API to create notifications that appear in the browser.

- Messaging: Developers can use the API to communicate between the background scripts and the extension's user interface or content scripts.

- Web navigation: Developers can use the API to monitor and control web page navigation.

- Web requests: Developers can use the API to intercept and modify HTTP requests and responses.

Overall, Chrome extension development requires knowledge of web technologies and the Chrome extension API. Developers need to be proficient in HTML, CSS, and JavaScript, and understand how to interact with the browser's functionality using the API. With these skills, developers can create powerful and innovative extensions that enhance the browsing experience for millions of users.

- **SDLC for Chrome Extensions Development**:-

The process of Chrome extension development can be broken down into the following steps:

1. Planning and Conceptualization: In this phase, developers brainstorm ideas for their extension and create a basic concept of what the extension should do. They also analyze the feasibility of their ideas and consider the potential user base for their extension.

2. Designing the User Interface: Once developers have a clear idea of the extension's functionality, they start designing the user interface. This involves creating mockups or wireframes of the extension's UI, including its layout, color scheme, and typography.

3. Creating the Manifest File: The manifest file is a configuration file that describes the extension's properties, including its name, version, permissions, and resources. Developers need to create a valid manifest file to package and distribute the extension.

4. Defining the Extension's User Interface: Developers need to define the extension's user interface, which can include browser actions, page actions, popups, options pages, and content scripts. They also need to create the necessary icons and images for the extension.

5. Implementing the Extension's Logic: After defining the UI, developers need to implement the extension's logic using JavaScript. They can use the Chrome extension API to access the browser's functionality, including tabs, windows, cookies, storage, and notifications. They can also interact with web pages and manipulate their content using the Document Object Model (DOM).

6. Testing and Debugging: Once the extension is developed, developers need to test and debug it to ensure that it works correctly. They can use the Chrome Developer Tools to debug and test the extension.

7. Publishing the Extension: Finally, developers can publish the extension to the Chrome Web Store, where users can download and install it. The Chrome Web Store provides a platform for developers to distribute their extensions to millions of users worldwide.

- **WorkFlow:-**

The workflow for Chrome extension development can vary depending on the project and the developer's preferences. Some developers may prefer to start with UI design, while others may begin with the implementation of the extension's logic. However, the overall process typically follows the steps outlined above.

Chrome Extensions

manifest.json

Describes the contents of the extension.

Defines rules for content scripts injection (url matchers).

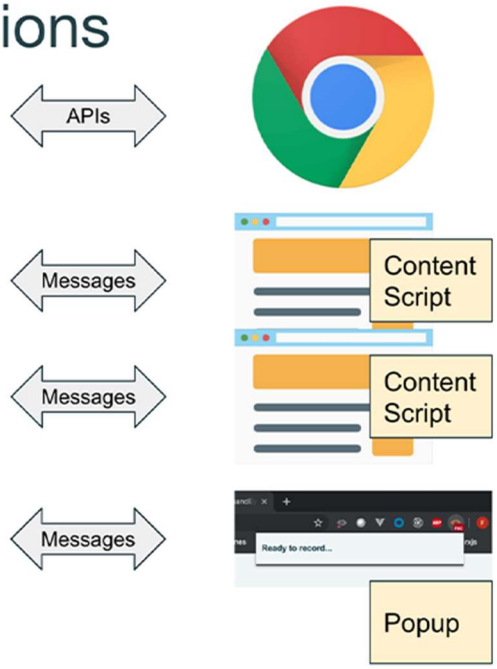
background.js (singleton)

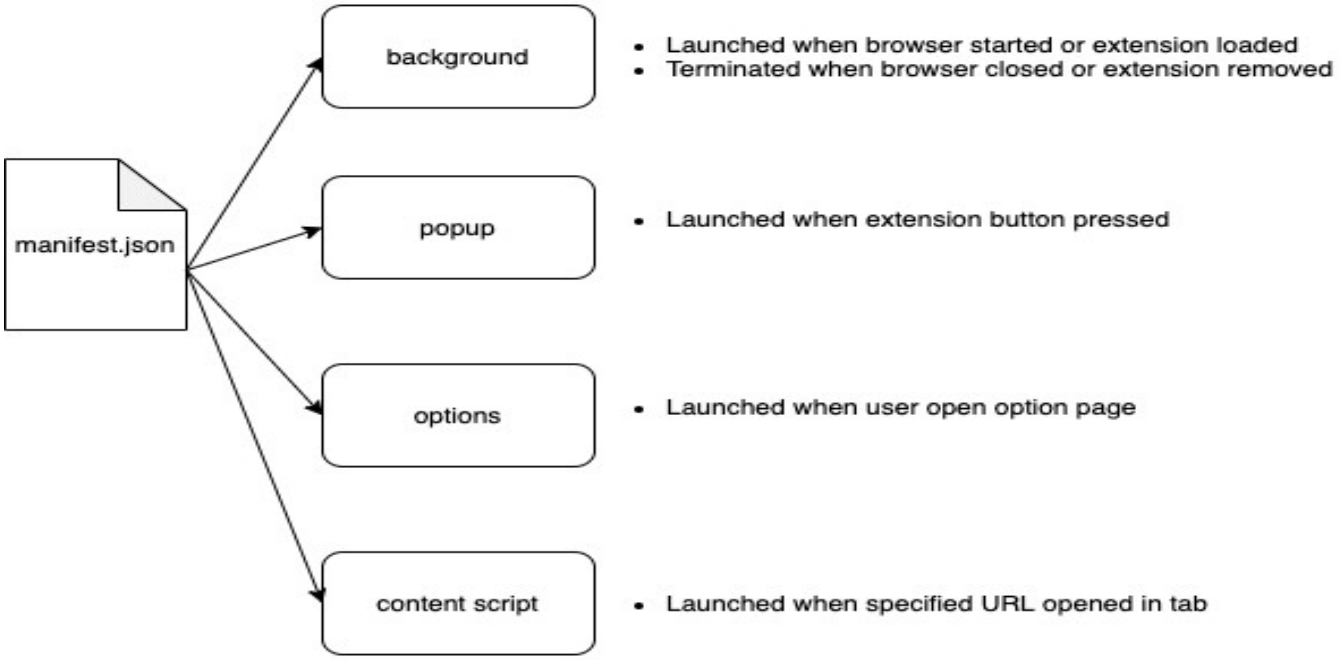
Can register to Chrome APIs such as:

- Native Messaging
- Web Request
- Web Navigation

Does not have direct access to DOM and pages, can have access through Content Scripts.

Interacts with the Popup.





In Chrome extension development, the manifest.json file is a crucial component of the extension that provides important information about the extension to the browser. The manifest.json file is a JSON file that contains a set of key-value pairs that define various properties of the extension, including its name, version, description, icons, permissions, and more.

Here are some of the key properties that can be defined in the manifest.json file:

- "name": The name of the extension as it will appear in the Chrome Web Store and in the browser's user interface.

- "version": The version number of the extension, which is used for updating the extension.

- "description": A short description of the extension that appears in the Chrome Web Store and in the browser's user interface.

- "icons": A set of icons that represent the extension at different sizes, including 16x16, 48x48, and 128x128 pixels.

- "permissions": A list of permissions that the extension requires to function properly, such as the ability to access the user's browsing history, tabs, or cookies.

- "content_scripts": A list of scripts that are injected into web pages and can modify the content of the page, listen for events, and communicate with other scripts in the extension.

- "background": A script that runs in the background of the browser and can interact with the browser's APIs.

- "browser_action" or "page_action": These properties define the extension's user interface, including the placement of the extension's icon in the browser's toolbar and the behavior of the extension's popup window.
In addition to these properties, the manifest.json file can also define a range of other properties that are used to configure the extension's behavior, including

the default settings, the supported languages, and more.

Overall, the manifest.json file is a critical part of any Chrome extension and provides important information that the browser needs to properly load and execute the extension. By understanding the various properties that can be defined in the manifest.json file, developers can create powerful and effective extensions that enhance the browsing experience for millions of users.

In Chrome extension development, background.js, popup.js, and content.js are three commonly used JavaScript files that work together to add functionality to the browser.

1. background.js: The background.js file is a special script that runs in the background of the browser and can interact with the browser's APIs. This file is loaded when the extension is first installed and remains active for as long as the browser is open. Developers typically use background.js to implement long-running tasks, such as monitoring for events or handling user input. For example, a developer might use background.js to listen for incoming messages from other scripts in the extension, or to update the extension's icon badge based on the number of unread messages.

2. popup.js: The popup.js file is used to create and manage the extension's popup window, which appears when the user clicks on the extension's icon in the browser's toolbar. Developers use popup.js to create the UI for the popup window and to handle user interactions. For example, a developer might use popup.js to display a form for the user to input data, or to send messages to background.js to trigger other actions in the extension.

3. content.js: The content.js file is used to interact with the content of web pages. This file is loaded into the context of each web page that the user visits and can manipulate the page's DOM, listen for events, and communicate with other scripts in the extension. Developers typically use content.js to add new functionality to web pages or to automate tasks. For example, a developer might use content.js to add a button to a web page that triggers a specific action in the extension, or to monitor the contents of a page and display a notification when certain conditions are met.

Overall, these three JavaScript files work together to add functionality to a Chrome extension. background.js handles long-running tasks and communicates with the browser's APIs, popup.js creates and manages the extension's popup window, and content.js interacts with the content of web pages. By using these files in combination, developers can create powerful extensions that can enhance the browsing experience for millions of users.

- **Front-end Technologies:-**

It requires an intuitive user interface that enables users to enhancing user interface experience.Common front-end technologies used in the development are HTML, CSS, and JavaScript,Reactjs.

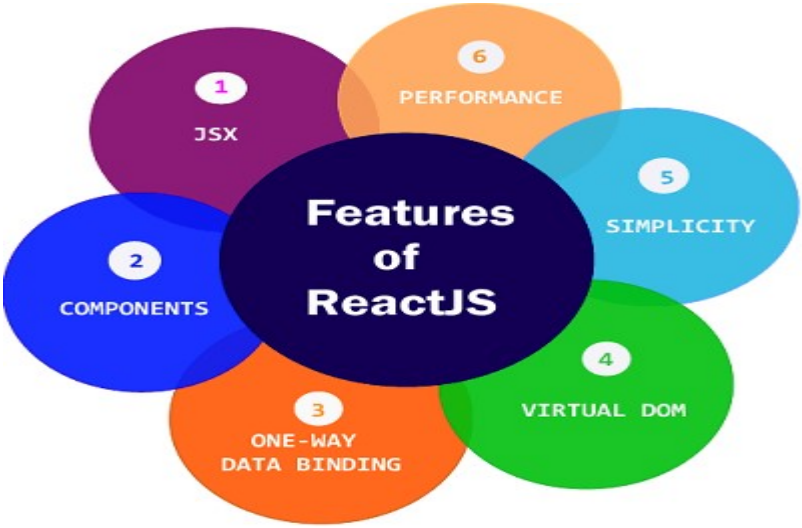
- **React JS:-**

React is a popular open-source JavaScript library used for building user interfaces. It was developed by Facebook and released in 2013. React has gained a lot of popularity due to its simplicity, flexibility, and efficiency in building complex user interfaces. It allows developers to create reusable UI components that can be used across different projects, making it easy to maintain and scale applications.

One of the key features of React is its use of a virtual DOM (Document Object Model), which allows it to update and render changes to the UI efficiently. Instead of updating the entire DOM whenever there is a change, React only updates the necessary components, resulting in faster performance and a smoother user experience.

React is often used in combination with other popular web technologies such as Redux, React Router, and Webpack. Redux is a popular state management library used with React to manage the application's data and state. React Router is a routing library used for handling navigation in a React application. Webpack is a module bundler used for optimizing and packaging the application's code for deployment.

React can be used to build a wide range of applications, including web applications, mobile apps, and desktop applications. It has a large and active community, with many resources available for learning and development.



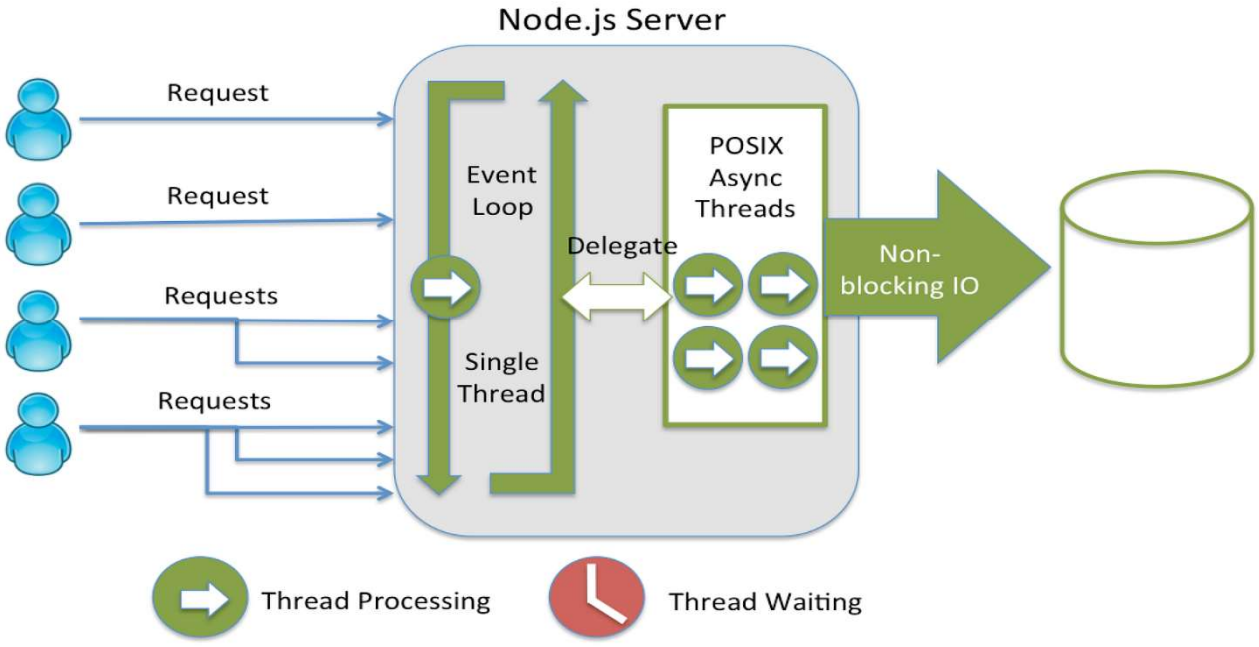
- **Back-end Technologies**:-

The back-end is the server-side of the chrome extensions development are that processes requests, stores data, and interacts with other systems. The back-end typically includes database management, order processing, inventory management, and integration with third-party systems such as payment gateways and shipping providers. Here we have used **Nodejs**.

- **Nodejs:-**

Node.js is often used as a backend technology in Chrome extension development to handle server-side tasks, such as storing and retrieving data, interacting with APIs, and performing other operations that require server-side processing.

In Chrome extension development, Node.js can be used to create a server that communicates with the extension's frontend through an API. The server can handle tasks such as retrieving data from a database, performing calculations, or processing large amounts of data. Node.js can also be used to set up websockets, which allows for real-time communication between the extension and the server.



Node.js can be integrated with other popular web technologies such as Express.js, a web application framework used for building APIs, and MongoDB, a popular NoSQL database used for storing and retrieving data.

Overall, Node.js is a powerful and flexible backend technology that can be used in Chrome extension development to handle server-side tasks, improve the extension's performance, and enhance its functionality.

Mongodb :-

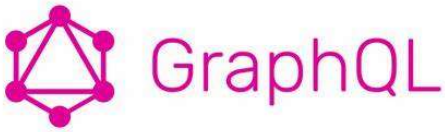
MongoDB is a popular NoSQL database that can be used in Chrome extension development to store and retrieve data. In a Chrome extension, MongoDB can be used to store data that is not required to be stored locally on the user's machine, such as user preferences, history, or settings. The extension can communicate with the MongoDB database through a Node.js server that is set up to handle requests to the database.

To use MongoDB in a Chrome extension, developers can use the MongoDB Node.js driver, which allows for easy integration between Node.js and MongoDB. The driver provides a simple API for connecting to the database, inserting and retrieving data, and performing other database operations.

MongoDB's flexible schema also allows for easy modification of the data structure, which can be beneficial in cases where the data requirements for the extension may change over time. Additionally, MongoDB's scalability allows for easy scaling of the database to handle large amounts of data.

Overall, using MongoDB in Chrome extension development can provide a reliable and flexible solution for storing and retrieving data.

GraphQL :-



GraphQL is a query language for APIs that can be used in Chrome extension development to simplify communication between the frontend and backend.

In a Chrome extension, GraphQL can be used to define a schema that specifies the available data and operations that can be performed on the data. The extension can then use this schema to make requests to the backend, which can then return the requested data.

One of the benefits of using GraphQL in Chrome extension development is its ability to reduce over-fetching and under-fetching of data. With traditional REST APIs, developers often have to make multiple requests to retrieve all the required data, resulting in over-fetching of data. On the other hand, under-fetching occurs when the API does not provide enough data in a single request, resulting in additional requests to retrieve the required data. GraphQL solves these issues by allowing the frontend to specify exactly what data is required in a single request.

Additionally, GraphQL allows for easy versioning of the API, as changes to the schema can be made without affecting the existing functionality of the extension. This can be beneficial in cases where the data requirements for the extension may change over time.

To use GraphQL in a Chrome extension, developers can use a GraphQL client library, such as Apollo Client, which provides a simple API for making GraphQL queries and mutations. Overall, using GraphQL in Chrome extension development can provide a more efficient and flexible solution for communicating between the frontend and backend.

Chrome API ‘s :-

Chrome APIs provide developers with a wide range of functionalities to build powerful and feature-rich Chrome extensions. Here are some of the key functionalities that Chrome APIs offer in Chrome extension development:

1. Tab and Window Management: Chrome APIs allow developers to manage tabs and windows in the Chrome browser. Developers can create, move, and close tabs, and manipulate the content of a tab, such as loading a new URL or injecting JavaScript.

   chrome.tabs.create({ url: 'https://www.example.com' });

2. User Interface: Chrome APIs provide developers with tools to create custom user interfaces for their extensions, such as popup windows, notification bars, and context menus.

```
chrome.browserAction.onClicked.addListener(function(tab) {
  chrome.windows.create({ url: 'popup.html', type: 'popup' });
});
```

3. Storage: Chrome APIs offer several storage options for Chrome extensions, including local storage, session storage, and sync storage. These storage options allow developers to store data, settings, and preferences for their extensions.

```
chrome.storage.sync.set({ key: 'value' }, function() {
  console.log('Value is set to ' + value);
});
```

4. Networking: Chrome APIs provide tools for communicating with external servers and APIs. Developers can use the Chrome APIs to make HTTP requests, send and receive data, and handle responses.

```
chrome.runtime.sendMessage({ message: 'Hello, world!' }, function(response) {
```

```
  console.log('Received response: ' + response);
});
```

5. WebRequest: Chrome APIs allow developers to monitor and intercept web requests made by the browser. This can be useful for adding custom functionality, such as blocking ads or redirecting URLs.

```
chrome.webRequest.onBeforeRequest.addListener(function(details) {
  return { redirectUrl: 'https://www.example.com' };
}, { urls: ['*://www.google.com/*'] }, ['blocking']);
```

6. Notifications: Chrome APIs provide developers with tools to create and manage notifications for their extensions. Developers can use the APIs to create pop-up notifications, desktop notifications, and notifications that appear in the Chrome browser.

```
chrome.notifications.create('notification-id', {
  type: 'basic',
  iconUrl: 'icon.png',
  title: 'Notification Title',
  message: 'Notification Message'
}, function() {
  console.log('Notification created');
});
```

7. Identity: Chrome APIs offer authentication and authorization options for extensions, including user authentication and OAuth2 authentication.

```
chrome.identity.getAuthToken({ interactive: true }, function(token) {
  console.log('Access token:', token);
});
```

8. Content Scripts: Chrome APIs allow developers to inject JavaScript and CSS into web pages, enabling them to modify the content and behavior of web pages.

```
chrome.tabs.executeScript(null, { file: 'content.js' }, function() {
```

```
  console.log('Content script injected');
});
```

Overall, the Chrome APIs offer a wide range of functionalities that can be used to create powerful and feature-rich Chrome extensions. By utilizing these APIs, developers can build extensions that offer unique and innovative functionality to enhance the browsing experience for millions of users.

GitLab :-

GitLab is a web-based Git repository manager that provides version control, continuous integration, and continuous deployment services. It is designed to help teams collaborate on software development projects and manage code, issue tracking, and CI/CD pipelines in a single platform.

GitLab supports both public and private repositories and allows developers to host their own GitLab instance on-premise or use the cloud-based GitLab.com service. It provides features such as code reviews, merge requests, automated testing, code coverage reports, and many more.
GitLab also provides a wide range of integrations with other development tools and services, including IDEs, project management tools, monitoring systems, and deployment platforms. This makes it easy for developers to connect all of their tools and workflows into a single, streamlined development pipeline.

Overall, GitLab is a comprehensive and powerful platform for managing software development projects, providing a wide range of features and integrations that help developers collaborate more effectively and deliver high-quality code faster.

Push: Pushing changes to a GitLab repository means uploading the local changes made to the code to the remote repository hosted on GitLab. This can be done through the command line or through GitLab's web interface. The push command transfers the changes to the remote repository, allowing others to access and collaborate on the code. Before pushing changes, developers

should ensure that they have the latest version of the code by pulling the changes from the remote repository.

```
# add changes to staging area
git add .

# commit changes with a message
git commit -m "Added new feature"

# push changes to remote repository
git push origin master
```

Pull: Pulling changes from a GitLab repository means downloading the latest version of the code from the remote repository to the local machine. This ensures that developers have the most up-to-date version of the code before making any changes. Pulling changes can be done through the command line or through GitLab's web interface.

```
# fetch latest changes from remote repository
git fetch

# merge changes into local branch
git merge origin/master
```

Commit: Committing changes to a GitLab repository means saving the changes made to the code in the local repository. This can be done through the command line or through GitLab's web interface. Commit messages should be descriptive and explain the changes made to the code. Committing changes also allows developers to revert to previous versions of the code if necessary.

```
# add changes to staging area
git add .

# commit changes with a message
git commit -m "Updated documentation"
```

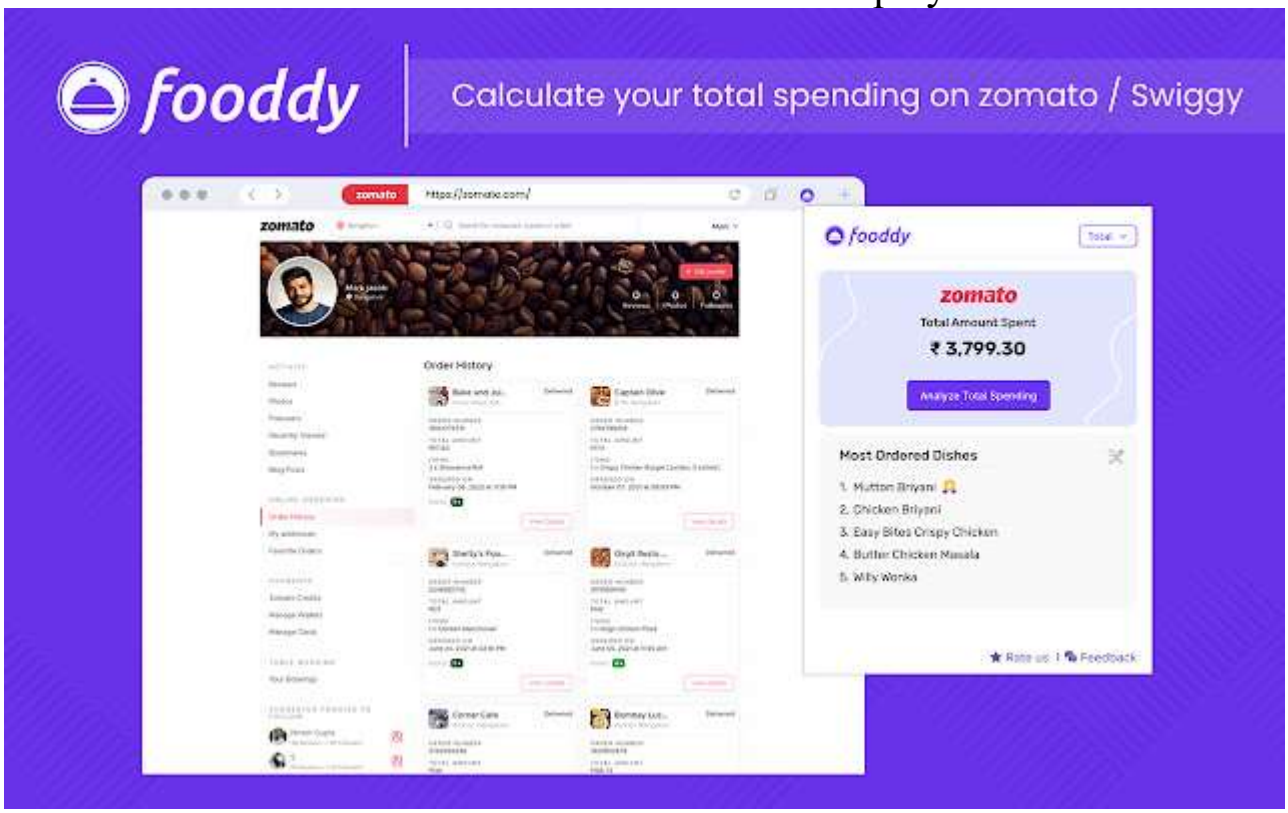
Overall, push, pull, and commit are important features of GitLab that allow developers to collaborate on code and manage changes to a repository. These features are essential for effective version control and maintaining code

quality.

Applications Developed:

1. Spending Calculator for Swiggy™ and Zomato™

This extension to calculate total spending on Swiggy™ and Zomato™ Find the total amount spent on Swiggy and Zomato online food orders. This extension calculates the total order amount and displays till to date.

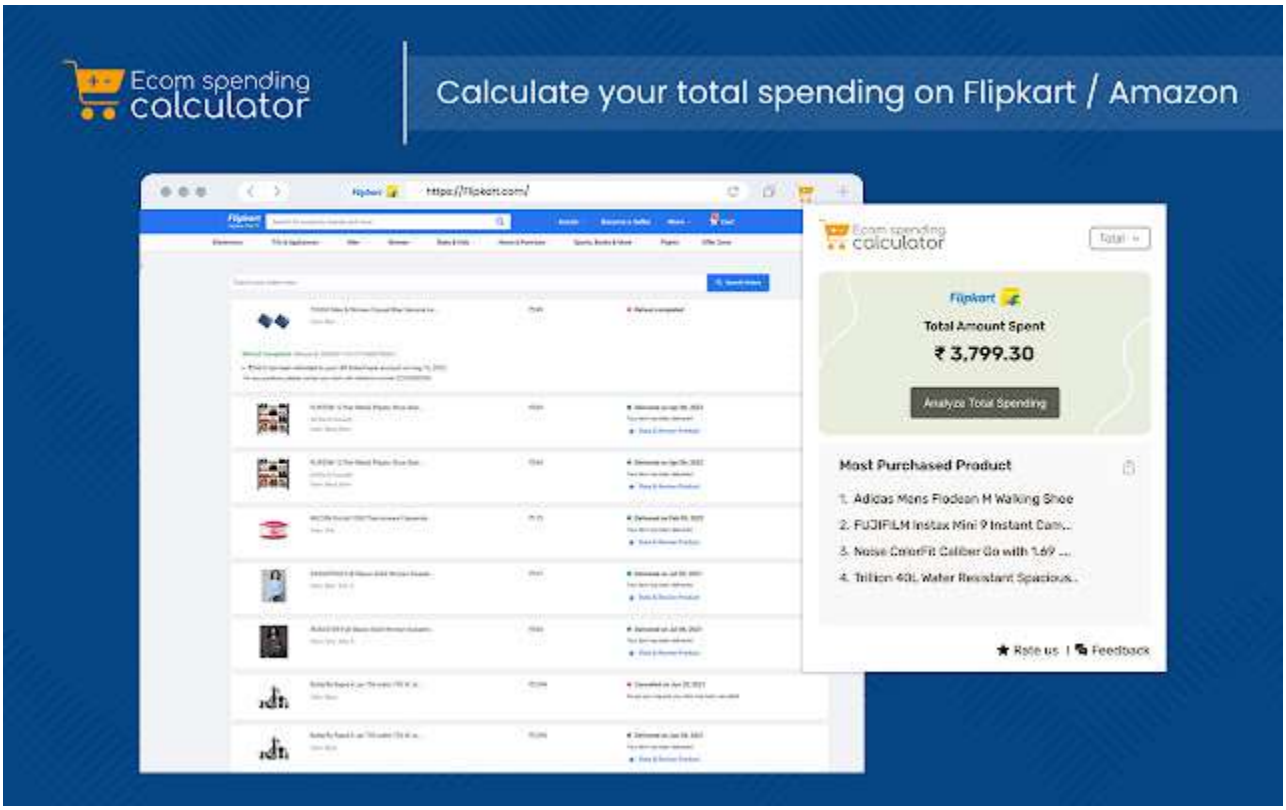


Spending tracker for Amazon & Flipkart:

This extension to calculate total spending on Amazon and Flipkart.

The Spending tracker for Amazon and Flipkart chrome extension helps users keep track of their spending on two of the most popular e-commerce platforms. The extension calculates the rear-wise total amount spent on orders made through Amazon™ and Flipkart™, allowing users to track their

spending over time and make more informed purchasing decisions.



Screen Recorder for Chrome™:

Free screen recorder is an easy-to-use screen recording tool that allows users to capture their screen, webcam, and microphone to create high-quality videos.

Key Features of Free Screen Recorder

1. Easy to use: It is user-friendly simple to use and completely free to use. With just one click, you can start recording your screen, webcam, or both.
2. High resolution recording: This free screen recording extension enables you to record your screen in up to 1080p resolution, ensuring that your videos are of high quality.
3. Multiple recording options: It offers a range of recording options, including recording your Screen, screen or webcam or both simultaneously. This allows you to choose the best recording option and

ensures that you capture the content according to your requirements.

4. Audio recording: Allows you to capture both system audio and microphone audio, which is essential for creating high-quality videos with clear sound.

Autoskip for Youtube™ Ads

Autoskip for Youtube™ is also known as Adblock for Youtube™ being used by thousands of fans across the globe.

Here is how it works:

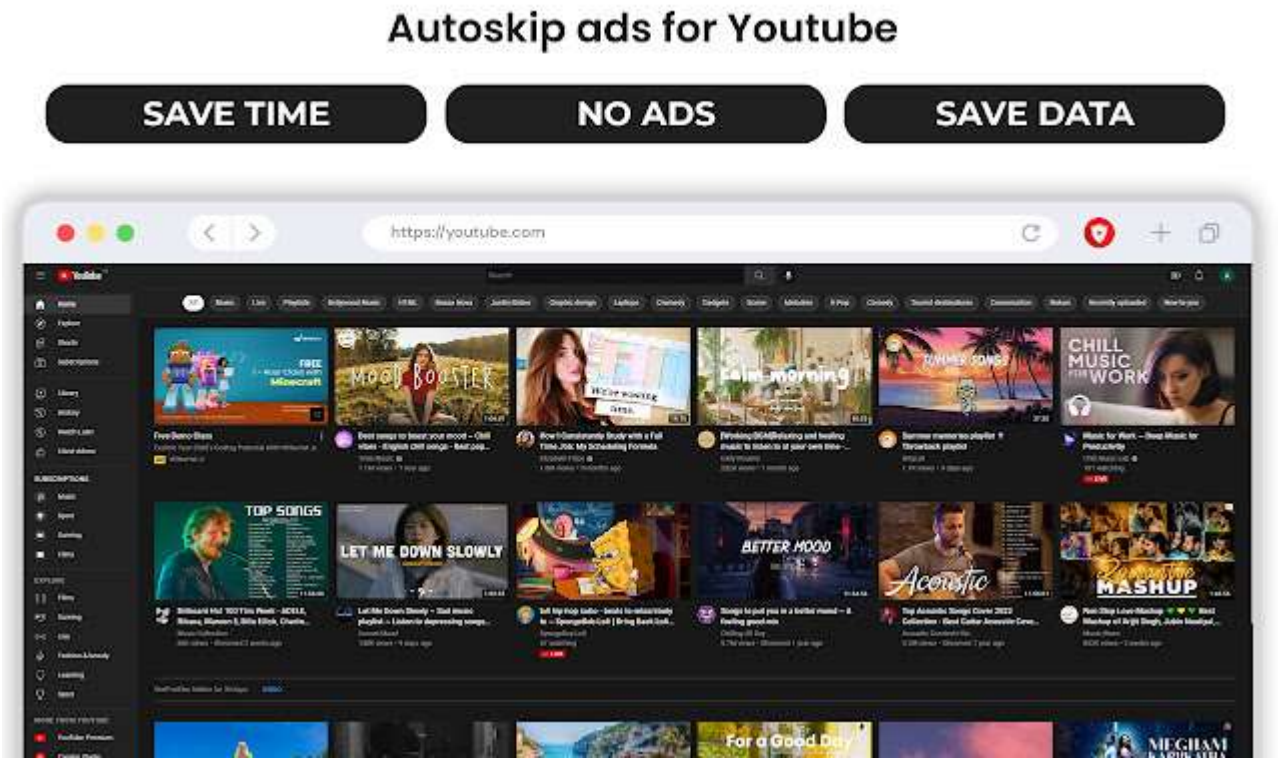
- ➔ Install this extension to your browser
- ➔ Visit any video you want to watch on youtube or other sites
- ➔ This extension automatically clicks on Skip button to block all the ads
- ➔ This will also block the sidebar ads and any ads in general

Features:

- ➔ blocks all ads, banner and pop ups on youtube™, vimeo and other video platform.

➔ blocking of ads on external sites where youtube™ video is embedded.
➔ prevent preroll ads from loading on Youtube™ and thus save your time.
➔ loads videos and YouTube website faster.
➔ Blocks ads in live streaming website also

Adblock for youtube™ can now block ads on YouTube™ videos that are embedded in websites.



Color Finder:

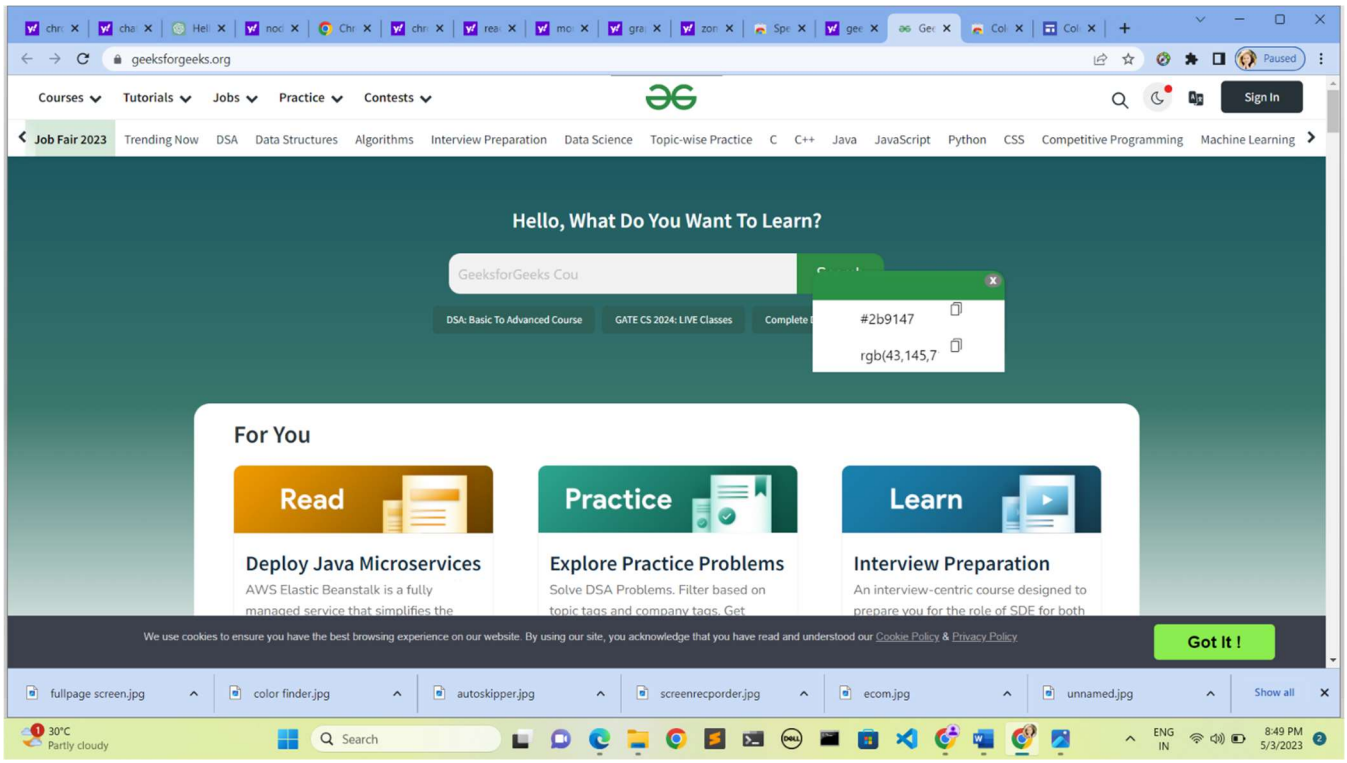
HEX, RGB Color Finder, Live Color picker eyedropper to find color codes Color finder makes it easier than ever to get accurate RGB and HEX color code values. Use this extension to find color codes quickly. A color finder enables you to find the perfect color for your design. It accurately identifies the HEX and RGB values. This is a great tool for web developers and graphic designers, as well as anyone who simply loves to find different color codes.

Key Features of Color Finder -

1. Find color codes with 100% accuracy
2. User friendly interface

3. Easily copy detected color codes with one click
4. Picks color codes from any online image
5. Smoothly works on all the websites

The Color Finder is an amazing tool for those who love designing images, logos, and webpages. This tool is a great help for developers and designers to find the right color for them. It automatically detects the color codes on your browser and generates color codes that can be copied and used easily. It works smoothly with Windows and Mac PCs. This is the best color picker Chrome extension. You can save time and effort by choosing the best one.



Full Page Screen Capture:

One click Full page screenshots of any webpage. Capture Full pages with ease Capture full page screenshots of any webpage

Looking for a quick and easy way to take full-page screenshots of websites? Look no further than the Full-Page Screenshot extension.
This free extension allows you to capture high-quality screenshots of entire web pages, including areas that are not visible on the screen.

Key Features of Full-Page screenshot:

- ➔ Takes high-quality screenshots of entire web pages, including areas that are not visible on the screen
- ➔ Saves screenshots as PNG and JPG files
- ➔ Completely free to use
- ➔ Easy user interface

How to Use:

- ➔ Install the Full-Page Screenshot extension
- ➔ Once installed, click on the extension icon in your browser toolbar to open the extension.
- ➔ Click the "Capture" button to take the screenshot.
- ➔ Save the screenshot as a PNG or JPG file

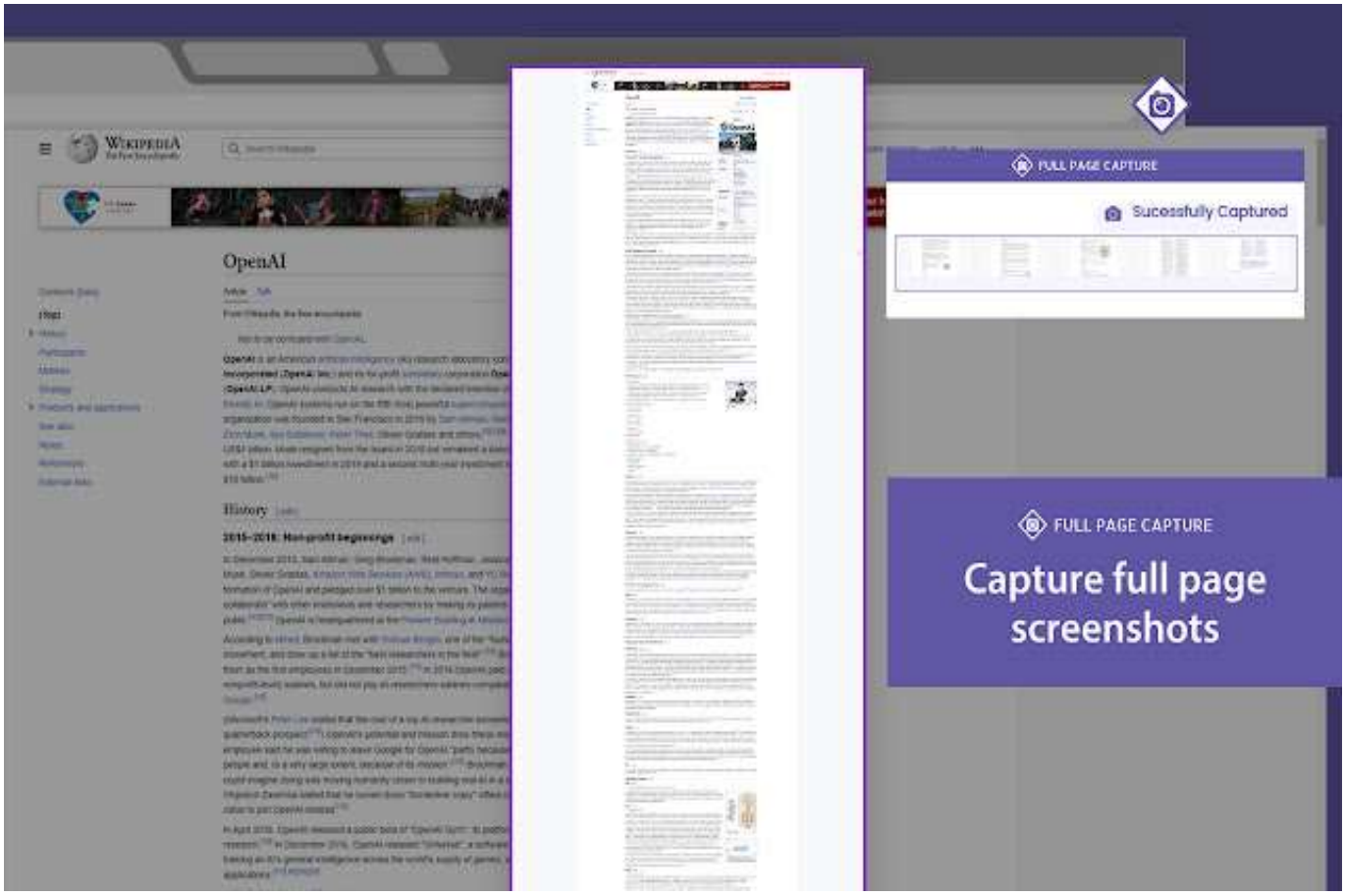
Why Is It Useful?

Here are some reasons why this extension is useful:

- ➔ Capture Entire Web Pages: The Full-Page Screenshot extension allows users to capture high-quality screenshots of entire web pages, including areas that are not visible on the screen.

- ➔ Easy to Use: It is easy to use, with a simple interface and intuitive controls.

➔ Ideal for Professionals: The extension is particularly useful for professionals such as designers, developers, and content creators who need to capture and share screenshots of web pages for work purposes.



Conclusion:

In conclusion, my experience in developing Chrome extensions was both challenging and rewarding. Throughout the project, I gained valuable hands-on experience in various aspects of software development, including design, coding, testing, and deployment. Developing Chrome extensions allowed me to explore the intricacies of web technologies such as HTML, CSS, and JavaScript, and how they can be leveraged to enhance the user experience.

Furthermore, the project taught me the importance of following best practices in software development, such as implementing agile methodologies and emphasizing collaboration and communication with other developers and stakeholders. By working in a team environment, I was able to learn how to manage my time efficiently and meet project deadlines.

Overall, this experience was invaluable in developing my skills and knowledge in Chrome extension development, and I look forward to applying them to future projects. I am grateful for the opportunity to work on this project and am excited about the endless possibilities of developing innovative tools and features for Chrome.