1. ABSTRACT

The Google Chrome extension for interactive web browser control with hand gestures interprets and responds to user hand movements through computer vision and machine learning. The way we interact with computers could be completely changed by using hand gestures as a control method. This technology translates human gestures into commands for computers. Using gestures, which the website will identify and respond to, is an easy and entertaining method to interact with content online. The use of an improved and updated model, a new motion to handle events and scrolls, and the ability to operate the custom cursor with two hands are all features of this improved Chrome extension. Hand gestures are helpful in a variety of contexts and offer particular advantages while navigating a web browser. This can be especially helpful in circumstances where using standard input techniques is not practical or convenient, as when you wish to avoid touching potentially contaminated surfaces or when your hands are full or unclean. Gesture control replaces the requirement for a remote control or clicker while navigating through web material or slide decks in a presentation or instructional context. Presenters have unrestricted mobility and can interact with the audience while managing the content displayed on the screen.

Keywords:

Tensorflow, hand gesture recognition, browser control method, browser, human interaction, API, Java Script, gesture detection.

2. INTRODUCTION

Human-computer interaction is changing as we get away from the traditional ways of utilising keyboards and mice. Rather, we are witnessing the emergence of creative methods, one especially intriguing advancement being the capacity to manipulate computers via hand movements. Through the use of computer vision and machine learning, this technology makes it possible to read and react to the complex hand gestures we make, enabling more natural and engaging interactions with digital systems. The way we engage with virtual environments, move through apps, and manipulate digital information could all be revolutionised by hand gesture-based computer control. This technology allows for a more fluid and tangible interaction between humans and computers by converting our natural movements into digital commands. This can be particularly helpful in situations when more conventional input techniques are impractical or ineffective. Using cutting-edge technologies to recognise and react to a user's hand movements is the process of operating computers by hand gestures. This technique bridges the gap between digital orders and human gestures by utilising a variety of technology components. Computer vision, gesture recognition algorithms, feature extraction, gesture mapping, real-time tracking, command execution, user feedback, calibration, and personalisation are some of the frequently used parts and processes in this technology. Hand gesture-based computer control finds applications across various domains, including gaming, virtual reality, augmented reality, healthcare, presentations, and more. It provides a natural and intuitive way to interact with technology, enhancing user experiences and enabling new forms of creative expression and efficient interaction. This project is about the method of controlling the mouse cursor events of web browser using hand gestures. Hand gesture control of web browsers is a complicated and developing field that is still being researched and improved upon for a number of reasons, including Gesture Vocabulary, Accuracy and Reliability, Variability in Gestures, Ambiguity and Context, and Real-Time Responsiveness. Positive and seamless user experience is a must for gesture-based interfaces. The main goals of research are to reduce physical strain, avoid weariness, and make sure that interactions are pleasant and intuitive. Given that gesture control systems take pictures, privacy issues must also be taken into consideration. This system is developed as a chrome browser extension API which allows the users to control the browser using hand gestures. This does not require any external hardware like kinetic sensors or any other sensors everything is done with webcam itself. This project functions as a special tool (API) that can be easily added to websites. Once added, it allows users to interact with web content by simply moving their hands. There is an

easy-to-use button for consumers to start using this hand motion function. When customers click it, the system will gain access to their camera by using the TensorFlow library, a potent tool. This makes it possible for the system to identify and comprehend the hand motions that users make. After that, users will receive a simple training to assist them in learning how to make the most of this function. Users can interact with online material in an engaging and natural way as the website responds to their gestures. Using hand gestures to control a web browser has some special benefits and can be useful in a variety of situations. This can be especially helpful in circumstances where using standard input techniques is not practical or convenient, as when you wish to avoid touching potentially contaminated surfaces or when your hands are full or unclean. Gesture control replaces the requirement for a remote control or clicker while navigating through web material or slide decks in a presentation or instructional context. Presenters have unrestricted mobility and can interact with the audience while managing the content displayed on the screen.

3. PROBLEM STATEMENT

The constraints and challenges associated with traditional computer input methods like keyboards and mice become readily apparent in real-life scenarios. This highlights the necessity for specialized solutions tailored to specific contexts. To illustrate, picture a scenario where you face a computer screen with your hands poised in the air. With a simple wave or gesture, you effortlessly control your cursor, select options, and navigate through the digital realm. This is where our Chrome Extension comes into play as a groundbreaking solution set to revolutionize your computer interaction. Focusing on simplicity and accessibility, we break free from the constraints of traditional input devices, offering a new way to bridge the gap between your intentions and digital actions.

Enhancing Accessibility and Convenience:

Consider individuals with disabilities, particularly those with limited dexterity or mobility impairments. They encounter significant challenges when trying to engage with digital content using traditional input devices. Operating a standard mouse or keyboard can be a formidable task, especially for those with motor disabilities, making even basic computer interactions daunting. Additionally, individuals with mobility impairments may struggle to physically reach and manipulate these devices, severely restricting their ability to independently interact with digital systems. In this context, a touchless interaction system in the form of a Chrome extension, designed to control the cursor with hand gestures, emerges as a valuable tool. This Chrome extension empowers individuals with disabilities to navigate and interact with digital content in a more inclusive and user-friendly way. By translating natural hand gestures into digital commands, it significantly enhances their overall digital experience. For example, a person with limited hand dexterity can effortlessly move the cursor, select items, and even click with ease using this extension. It caters to their unique needs and empowers them to interact with digital content independently.

Enhancing Presentations and Unconventional Settings:

In non-traditional and remote environments, such as a large auditorium where a presenter is giving a seminar, the limitations of traditional input methods become evident. Attempting to control a web browser with a traditional mouse in such a vast setting is not only impractical but also disrupts the presenter's movements and the flow of the presentation. The need for a more versatile means of interaction, which the Chrome extension offers, becomes apparent.

This extension seamlessly enables the management of digital content in a hands-free and intuitive manner, making it ideal for presenters and users in various non-traditional settings.

In Emergency Scenarios, a Critical Need for Speed:

Imagine a life-or-death emergency in a hospital operating room. The surgeon is in the midst of a complex procedure where every second counts. Suddenly, there's a need to access critical patient information, view real-time medical imaging, or refer to medical databases for crucial insights. In this high-stress environment, the speed at which information is retrieved can be a matter of life or death. However, the traditional means of interacting with a computer, such as keyboards and mice, fall short in this critical moment. The act of touching these devices can introduce contamination risks, and even a momentary diversion of the surgeon's attention may jeopardize the patient's well-being. This highlights the pressing need for an alternative interaction system that allows the surgeon to swiftly and safely access vital data. Here, a touchless interaction system, embodied as a Chrome extension, becomes an indispensable tool, enabling the surgeon to retrieve critical information with a simple gesture, without compromising hygiene or patient safety.

Meeting the Demands of an Intuitive User Experience:

There is a growing demand for an immersive and intuitive user experience across various contexts. Users today expect technology to seamlessly align with their natural behaviors, providing a user-friendly and intuitive interaction experience. Traditional input methods, while effective in many cases, often fall short of these expectations, prompting the need for a specialized solution like a Chrome extension. This extension enhances user experiences and offers a more intuitive way to interact with digital systems, particularly in web browsing scenarios.

Complementing Traditional Input Methods:

One might question the utility of a single Chrome extension in situations where the entire computer is traditionally controlled through keyboards and mice. The extension serves as a specialized tool, specifically designed for web browsing, where it complements traditional input methods. While keyboards and mice excel in general computer tasks, certain scenarios benefit from the advantages of hand gesture-based control. For instance, in a presentation or lecture, navigating through web content or slide decks can be cumbersome with a traditional mouse and keyboard, especially in large auditoriums. In such cases, presenters may need to move freely and engage with the audience while simultaneously controlling content on a

screen. The Chrome extension offers the flexibility to seamlessly switch between traditional input methods and hand gestures, providing users with a more versatile means of interaction. It enhances user experiences, ensures hygiene in scenarios where touch should be avoided, and caters to individuals seeking a more intuitive and interactive web browsing experience.

Prioritizing Data Privacy and Security:

The Chrome extension addresses the growing concerns regarding data privacy and surveillance, which are becoming increasingly prevalent in our interconnected world. External sensors and cameras used in certain systems can provoke apprehensions about data collection and potential surveillance. However, by relying solely on the built-in webcam of a device, the extension mitigates these privacy concerns, emphasizing the importance of a secure and privacy-conscious interaction method.

The software for hand gesture-based cursor control in web browsing emerges as a valuable tool, addressing pressing challenges and providing a more intuitive and versatile means of interaction in a variety of real-life scenarios. It empowers individuals with disabilities, caters to presenters in non-traditional settings, and enhances user experiences, all while upholding data privacy and security.

4. LITERATURE REVIEW

S.NO	Title	Authors & Year	Year	DOI/Issue Number
1.	WEB BROWSER CONTROL USING HAND GESTURES	Dr. N Kumaran, Balina Surya, Bobba Vamsi Krishna.,	2023	10.55041/IJSREM21244
2	Virtual Mouse using Hand Gestures	R. Matlani, R. Dadlani, S. Dumbre, S. Mishra and A. Tewari.,	2022	10.1109/ICTAI53825.2021.9673251
3.	Controlling web browser via different hand gestures and speech recognition	Rithika V. Jadhav, Prajakta S. Bait, Sumedh Pundkar.,	2020	https://www.ijariit.com/manuscripts/v6i4/V6I4-1167.pdf
4	Hand Gesture Recognition Control for Computers Using Arduino	Vimali, J.S., Srinivasulu, S., Jabez, J., Gowri, S.,	2021	10.1007/978-981-15-8530-2_45
5	Web-Based Real-Time Gesture Recognition with Voice.	Ghadekar Premanand Pralhad, S. Abhishek, Tejas Kachare, Om Deshpande, Rushikesh Chounde, Prachi Tapadiya.,	2021	10.1007/978-3-030-88378-2_10
6	Experimenting Touchless Gestural Interaction for a University Public Web-based Display	Giorgia Persichella, Calogero Luca Lomanto, Claudio Mattutino, Fabiana Vernero, Cristina Gena	2019	https://doi.org/10.475/123_4

7	A Gaze-Based Web Browser with Multiple Methods for Link Selection.	Matteo Casarini, Marco Porta, Piercarlo Dondi,	2020	10.1145/3379157.3388929
8	DRAWSOCKET: A BROWSER BASED SYSTEM FOR NETWORKEDSCORE DISPLAY	Rama Gottfried, Georg Hajdu.,	2019	10.1109/CEEC47804.2019.8974342
9	Design and Evaluation of an Open-source Gaze-controlled GUI for Web-browsing.	Fanny Larradet, Giacinto Barresi, Leonardo S. Mattos.,	2019	10.1109/CEEC47804.2019.8974342
10	Hand Gesture Recognition: A Literature Review	Rafiqul Zaman Khan and Noor Adnan Ibraheem	2012	10.5121/ijaia.2012.3412
11	Error detection and comparison of gesture control technologies.	Aditya Prasad Mahapatra , Bishweashwar Sukla, Harikrishnan KM , Debani Prasad Mishra, Surender Reddy Salkuti.,	2022	10.11591/ijai.v11.i2.pp709-716
12	Web-Based Gesture Recognition System for Controlling Heterogeneous IoT devices using Deep Learning	Nikhil Jangamreddy, Ropar, Vivek Sethi, Sujata Pal.,	2019	10.1109/COMSNETS.2019.8711158
13	Integrated devices that can recognize hand gestures.	Chanho Shin, Tse Nga Ng.,	2023	10.1038/s41928-023-01003-0
14	Approach for Improving User Interface Based on Gesture Recognition.	Issam Elmagrouni, Abdelaziz Ettaoufik, Siham Aouad and	2021	10.1051/e3sconf/202129701030

		Abderrahim Maizate.,		
15	Voice Assistant and Gesture Controlled Virtual Mouse using Deep Learning Technique.	M Raja, P Nagaraj, Polishetty Sathwik, Khayamkhani Mujahid Ali Khan, Nukala Mohith Kumar, U Shiva Prasad.,	2023	10.1109/ICSCDS56580.2023.10104619
16	Hand Gesture Recognition Based Human Computer Interaction to Control Multiple Applications	Sanzida Islam, Abdul Matin, Hafsa Binte Kibria.,	2022	10.1007/978-3-030-93247-3_39
17	Gesture controlled interaction using hand pose model	Rina Damdoo, Ashutosh Gupta.,	2022	10.53730/ijhs.v6nS1.7493
18	Hand-Gesture and Voice-Activated Digital Cursor	Mohammad Yunus, Karri Sndeep Simha, J. Refonaa, S.L.Jany Shabu, S. Dhamodaran, Viji Amutha Mary.,	2023	10.1109/ViTECoN58111.2023.10157786
19	Approach for the construction of gestural interfaces to control graphical interfaces based on artificial intelligence.	Issam El Magrouni, Abdelaziz Ettaoufik, Aouad Siham, Abderrahim Maizate, Bennani Lotfi.,	2022	10.1109/WINCOM55661.2022.9966424
20	Improved Real-Time Approach to Static Hand Gesture Recognition	Bhavitha B, Divyaprakash R, Vedha T Selvam, V Vinith Kumar, Ramanathan R.,	2021	10.1109/Confluence51648.2021.9377036

25	A basic hand gesture control system for PC applications	C.J. Cohen; G. Beach; G. Foulk.,	2002	10.1109/AIPR.2001.991206
26	Gesture browsing method for geological data.	Huang Chen , Li Jianhua , Yu Hanchao ,Shen Jianghai.	2013	10.2991/icmt-13.2013.64
27	Real-Time Hand Gesture Interface for Browsing Medical Images	Juan Wachs, Helman Stern, Yael Edan, Michael Gillam, Craig Feied, Mark Smith, Jon Handler. .	2008	10.1080/1931308x.2008.10644149
28	Development of an Efficient Hand Gesture Recognition system for human computer interaction.	Umadevi N, Divyasree I.R.	2016	10.18535/Ijecs/v4i12.5.
30	Gestures Recognition from Sound Waves	Nguyen Dang Binh.	2016	10.4108/eai.12-9-2016.151679
31	Hand gesture controls for image categorization in immersive virtual environments	Chao Peng, Jeff Hansberger, Lizhou Cao, Vaidyanath Areyur Shanthakumar.	2017	10.1109/VR.2017.7892311.

5. PROPOSED SYSTEM

The ability to manipulate web content with hand gestures is a crucial modern feature that enhances the convenience, accessibility, and engagement of online interactions. Just picture being able to use a hand motion to adjust the volume or browse through videos on YouTube or Netflix. This will make the viewing experience even more engaging for you. Similarly, hand gestures could improve the efficacy of remote work by enabling more natural engagement and document navigation during virtual meetings on platforms like Zoom or when working on digital whiteboards. In situations like teaching seminars, hand gestures can boost involvement and create a more interesting learning atmosphere. Such technology needs to be responsive, have a straightforward interface, be tested frequently with actual users, and be created with the user in mind in order to be both user-friendly and effective.

This system is designed that works with Google Chrome web browser, similar to an extension. By adding additional features and enabling hand gesture control over web content, it's meant to enhance your browsing experience. Modern technology is used in it, including the WebRTC API for real-time webcam access, the TensorFlow Handpose model for accurate hand motion recognition, React.js for the user interface, and Node.js for server-side functionality. Additionally, HTML, CSS, and JavaScript are used. With this plugin, you may activate your webcam and control mouse motions and functions with hand gestures, adding even more functionality to your online experience.

Numerous forms, such as desktop software, web-based apps, mobile apps, AR/VR apps, game console integrations, IoT device compatibility, and wearable technology, can be built for this solution. However, there are limitations and challenges specific to both mobile and AR/VR applications. Still, there are a lot of benefits to developing it as a Chrome extension. With these beautifully integrated Chrome browser extensions, you can simply access gesture control functions. They work with many different systems, are highly configurable, and are simple to install. Furthermore, Chrome extensions facilitate web service integration, increase efficiency, enhance user engagement, and offer marketing opportunities.

JavaScript may make it challenging to control the location of the computer pointer outside of a web browser due to security and privacy concerns. To get around this, the extension produces a unique cursor on the webpage that you may control with hand gestures. By securely accessing your webcam over the WebRTC API and not saving any data, you can maintain the privacy and security of your webcam broadcast.

This Chrome extension is more readily available than certain alternatives (like AR/VR solutions) since it enables hand gesture control without requiring users to buy specialised hardware. By prioritising user privacy and ensuring safe data processing, this extension's use of the WebRTC API to enable camera access fosters user confidence. Using Node.js for server-side processing and the TensorFlow Handpose model for gesture recognition ensures good performance and responsiveness. Because it uses the Tensorflow model, this offers a high degree of accuracy in identifying and detecting hand movements and poses. Because it can accurately recognise a range of hand gestures and postures, it is appropriate for uses where accuracy is crucial. Additionally, it is designed to function well on graphics processing units (GPUs), which quickens the model's rate of inference and qualifies it for real-time processing applications. This makes use of react.js, which is well-known for its capacity to produce dynamic and intuitive user interfaces. A well-thought-out and user-friendly user interface can improve the overall experience for users while developing Chrome extensions. Because the react.js framework is utilised here, good performance can be guaranteed, guaranteeing that the extension functions smoothly and reacts rapidly to user input. The Chrome extensions are compatible with several operating systems, so that users on Windows, macOS, and Linux will have a consistent experience. This can streamline tasks, making users more productive in their web activities. This interactive nature of hand gesture control can also increase user engagement with web content and applications. This chrome extension is also improved from the previous by using updated model and the custom cursor can be controlled from two hands.

6. OBJECTIVES & SCOPE:

The primary objective is the development of a user-friendly Chrome Extension that enables users to control web browsers using hand gestures. The significance of this technology lies in its potential to transform the way individuals interact with digital content. Here is the outline the key aspects of the software's objectives and scope:

1. **Enhancing Web Browsing Interaction:**

   The central aim is to revolutionize user interaction with web browsers by introducing a novel method for controlling web content through hand gestures. This approach strives to make web browsing more engaging and intuitive. For instance, users can seamlessly open and close tabs or navigate through websites by making specific hand gestures. The innovation aims to offer users a more immersive and physically intuitive way to browse the web, particularly in scenarios where conventional input methods may be less practical or efficient.

2. **Advanced Gesture Recognition:**

   The goal is to develop sophisticated algorithms for accurately recognizing a wide array of hand gestures. These gestures will be mapped to specific browser commands, such as scrolling, zooming, and opening links, to provide users with a seamless browsing experience. For instance, users can zoom in on a webpage by executing a pinching motion with their fingers. Advanced gesture recognition empowers users to interact with web content more naturally, reducing their dependence on traditional input devices.

3. **Real-Time Responsiveness:**

   Achieving real-time responsiveness is pivotal for a user-friendly experience. The objective is to ensure that the system responds instantly to user gestures. When a user moves their hand, the on-screen cursor will track the movement without any noticeable lag. This instant feedback is critical for creating a seamless and responsive interaction, giving users the sensation of directly controlling the content with their gestures.

4. **Privacy Protection:**

   Safeguarding user privacy is a top priority. Stringent measures will be implemented to protect user data, particularly since the system relies on the computer's webcam to capture hand gestures. Users can, for example, grant one-time access to the webcam for the extension, ensuring their privacy while browsing. Ensuring the security and privacy

of users' personal data stands as a paramount objective, and rigorous measures will be taken to accomplish this.

5. **Versatile Application:**

   The extension is designed to find applications in various domains beyond basic web browsing. It can be employed in gaming, enabling users to control in-game actions with gestures, or in healthcare, where healthcare professionals can seamlessly navigate medical records. In a healthcare scenario, for instance, a doctor could utilize hand gestures to scroll through patient records and images during a diagnosis. The adaptability of this technology opens up new avenues for creative and efficient interaction in diverse fields.

6. **User-friendly Interaction:**

   The extension aims to provide a more user-friendly and comfortable way for users to interact with their computers. This will help reduce physical strain and fatigue often associated with traditional input methods. Users can navigate web content without the constant use of a mouse or keyboard, lessening the physical strain on their hands and wrists. By introducing a more comfortable and physically friendly interaction method, this technology promotes user well-being and health-conscious computing.

7. **Context-Aware Gesture Interpretation:**

   The system will be designed to intelligently interpret gestures, even in situations where they may not be entirely clear. It adapts to the context, providing a seamless user experience. In a document editing scenario, for instance, the extension can distinguish between gestures meant for text editing and those intended for navigation, ensuring precise actions. Context-aware interpretation enhances user experiences by ensuring that the system accurately responds to the intended actions, regardless of the scenario in which the gestures are made.

8. **Education and Presentation Enhancement:**

   The potential of the extension extends to education and presentations. It allows educators and presenters to navigate web content and slides without the need for traditional remote controls, providing a more dynamic and engaging experience for their audience. For example, a teacher can use hand gestures to switch between slides during an online lecture, making the presentation more interactive. This capability enhances the interaction in educational and presentation scenarios, allowing for more engaging and dynamic content delivery.

9. **Accessibility and Inclusivity:**

Ensuring that the extension is accessible to a wide range of users, including those with physical disabilities, is a fundamental aspect of the technology. Individuals with mobility impairments can use the extension to navigate websites without physical input devices. This commitment to inclusivity and accessibility reflects the importance of providing equal opportunities for all users, regardless of their physical capabilities, and making digital interactions more accessible and equitable.

7. REQUIREMENT SPECIFICATION

- **Software Requirements:**
    - **Operating systems:** The API will be compatible with above windows 10 and above versions, MacOS 10.14 and Linux. This ensures that a wide range of users with different operating system can access the API.
    - **Browser:** The extension should be primarily designed and tested for google chrome as it's the target browser. The browser also requires user's permissions for camera. Since the minimum requirement for operating system is windows 10 and above all the versions of chrome browser support extension.
    - **Gesture recognition library:** For gesture recognition, the extension can leverage a JavaScript library like Tensorflow.js to process the webcam feed and detect hand gestures. Tensorflow.js provides machine learning capabilities in the browser, allowing for real-time gesture recognition.
    - **Backend infrastructure:** The API will be built using the node.js environment. All the chrome browsers are enabled with v8 engine which is built in C++ and is used in node.js.
    - **Languages:** The Chrome extension's frontend is developed using HTML, CSS, and JavaScript, making it compatible with Chrome's extension framework. JavaScript will handle capturing and processing hand gestures. TensorFlow which is an open-source machine learning framework developed by the Google Brain team. TensorFlow is primarily written in C++ for efficiency and performance.
    - **Text Editor:** A text editor like Visual Studio Code, Sublime Text, Atom, or an IDE like WebStorm can be used to write, edit, and manage your extension's code. Visual Studio code is more preferred. Because VS Code has a vast extension market place, a built-in debugger for JavaScript, and GIT integration.
- **Hardware Requirements:**
    - **Ram:** A minimum of 8 GB of RAM is generally recommended for running the chrome extension, TensorFlow.js and performing various machine learning tasks.
    - **CPU:** A normal chrome extension basically is light weighted and a normal processor capable of running simple tasks is enough. But in this it has TensorFlow, so a modern multi-core CPU (e.g., Intel Core i5 or equivalent

AMD processors) should be sufficient for running this chrome extension and performing common machine learning tasks.

- **Storage:** This chrome extension which has TensorFlow.js library is typically a few megabytes in size, including the core functionalities needed to run machine learning models in the browser or Node.js.
- **Internet:** A stable internet connection is necessary to load the chrome extension, TensorFlow.js and related models from online sources, especially need since using it in a web browser.
- **Webcam:** For using this chrome extension, the webcam is crucial to ensure accurate and efficient tracking of hand movements. A webcam with a resolution of 720p or higher, 30 frames per second or higher, low latency, Good Low-Light Performance, Wide Field of View (FOV), Auto Focus and Auto Exposure is recommended

- **Input Requirements:**

  A "manifest.json" file is a key requirement for any Chrome extension. It contains metadata about the extension, including its name, version, description, permissions, and more about the chrome extension. Using webcam, video frames are captured in real-time. This is using browser WebRTC API. Then preprocesses each captured frame to prepare it for input to the AI model.

- **Output Requirements:**

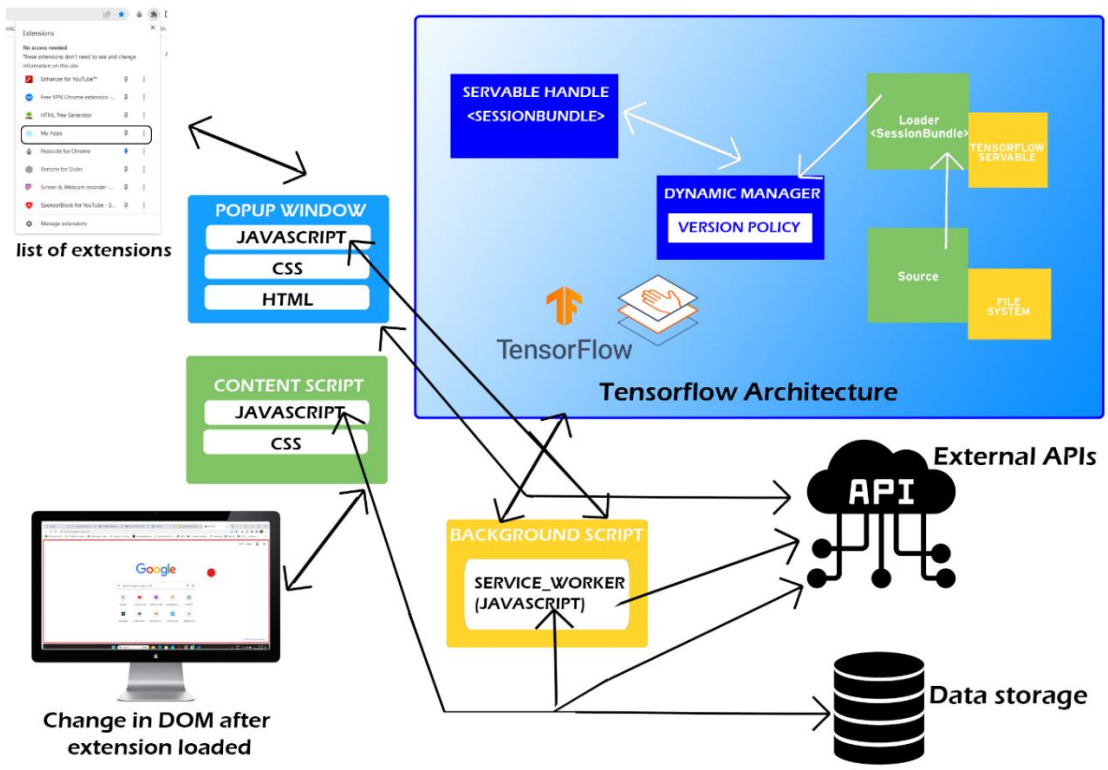
  The model's predictions are needed, which indicate the detected hand gestures. Based on the predictions the hand movements are shown to the user in real-time as skeleton-based output. And based on the model's predictions the cursor movements are made.

- **Data Requirements:**

  Tensorflow AI model is being used to do hand-gesture recognition for the creation of this Chrome extension. Its backend infrastructure is really strong. This includes all the data needed to develop this Chrome extension, including Pre-Trained Machine Learning Models, Training Data, Testing and Validation Data, Model Architecture and Configuration, Preprocessing Data, Augmentation Data, Model Evaluation Metrics, Calibration Data, and Gesture-to-Action Mapping.

8. SYSTEM DESIGN AND METHODOLOGY

Designing a Chrome extension involves many of the same considerations as any other software development project, including system architecture and methodology. Quality, maintainability, and success of your extension can be significantly impacted by a well-planned system design and an organised process. The functionality and user experience of your extension are shaped in part by well-designed systems. The extension's functionality, features, and implementation can all be planned using this tool. This guarantees that users receive value from the extension and that it fulfils its intended function. An organised codebase results from effective system design. Development, testing, and maintenance of the extension are thus made simpler as a result. If you work in a team or need other developers to comprehend or contribute to your extension, having well-organized code can also facilitate communication. You might consider system design when considering compatibility with different operating systems, browsers, and Chrome versions. It may be simpler to modify your extension for various situations if portability was taken into consideration during design. Creating a Chrome extension that utilises hand gestures to control cursor movements is an intriguing and somewhat challenging undertaking. Depending on the particular objectives and setting of your Chrome extension for hand gesture control, these factors may or may not be important. Nonetheless, a lot of these factors are typically really significant.



List of chrome extensions:

A graphical user interface known as the "list of extensions window" appears in a web browser, such Google Chrome, and provides a complete inventory of all loaded add-ons and extensions. With an overview of the names, icons, and settings of all installed extensions, this window lets users monitor, manage, and customise their browser extensions. Popup.html, Popup.css, and Popup.js will be called and the extension will begin to function when selected. This allows the user to control the cursor motions with hand gestures.

Change in DOM after extension loaded:

The Document Object Model (DOM) of the web pages the user is presently viewing may be affected by a Chrome extension once it has loaded and run. For many Chrome extensions, the ability to alter the DOM is essential since it lets them improve or personalise web pages according to their features. Hand motions will be used to control a custom cursor that is formed in the webpage's dom when this built Chrome extension loads.

Popup window:

A popup window, simply called a "popup," is a little window with a graphical user interface (GUI) that rises to the top of the active web page or application interface. Typically, it is smaller than the primary content and appears there. This is used for showing more information, getting feedback from users, or sending alerts. This offers additional information as well without requiring the user to leave the current page. Upon loading, this application opens a popup window that provides a brief instruction on how to use the extension and serves as a notification system for the user.

Content Script:

JavaScript files of the content script type are used in this Chrome extension to interact with and change the Document Object Model (DOM) of the webpage the user is presently viewing. In order to improve user experience and offer unique functionality to websites, content scripts are a crucial part of browser extensions. In the building of this Chrome extension, content script is crucial. The task of using TensorFlow models to recognise hand motions is handled by content scripts, which also handle input from the user's webcam or other sensors. Real-time interaction between the user's motions and the movement of the pointer on web pages is made possible in this way by content scripts.

Background Script:

An essential part of this Chrome extension's development is a background script. It is a JavaScript file that operates independently of the user's current web page. This background script is mostly used to handle tasks that need to be processed continuously or over an extended period of time, manage the operation of the extension, and enable communication between its many components. This is a background script that handles hand gesture processing and detection using the user's webcam or other sensors. Typically, it incorporates various gesture recognition techniques or machine learning models (like TensorFlow). To detect gestures, the background script continuously examines the incoming data. After hand gestures are identified, the gestures are translated into precise cursor movements and actions by the background script. It calculates the new cursor position and updates the cursor's behaviour based on the user's gestures.

Data storage:

The storing and management of diverse kinds of data required for the operation of Chrome extensions is accomplished through data storage. Developers can store user data, preferences, extension settings, and more with it. The user preferences and settings, extension state, user data, history and usage data, cached data, session data, persistent data, options and configuration, custom data structures, file storage, cookies, and external data sources are all often stored by Chrome extensions. This Chrome application records video in real-time and sends it to TensorFlow for gesture analysis. A sizable amount of data for this artificial intelligence model is already saved in the TensorFlow cloud. Thus, this Chrome loads the hand posture model repeatedly, using the browser's random Access Memory (RAM) to facilitate the process. Therefore, the user needs to have enough RAM to use this addon.

External APIs:

Utilising the many APIs (Application Programming Interfaces) offered by the Chrome Extension API is necessary to use Chrome extensions. Developers can communicate with web pages, browsers, and other components through these APIs. The chrome.runtime, chrome.browserAction, and chrome.pageAction APIs, as well as the chrome.windows, chrome.extension, chrome.cookies, storage, notifications, web navigation, identity, and permissions APIs, are all used by this chrome extension.

Architecture of TensorFlow:

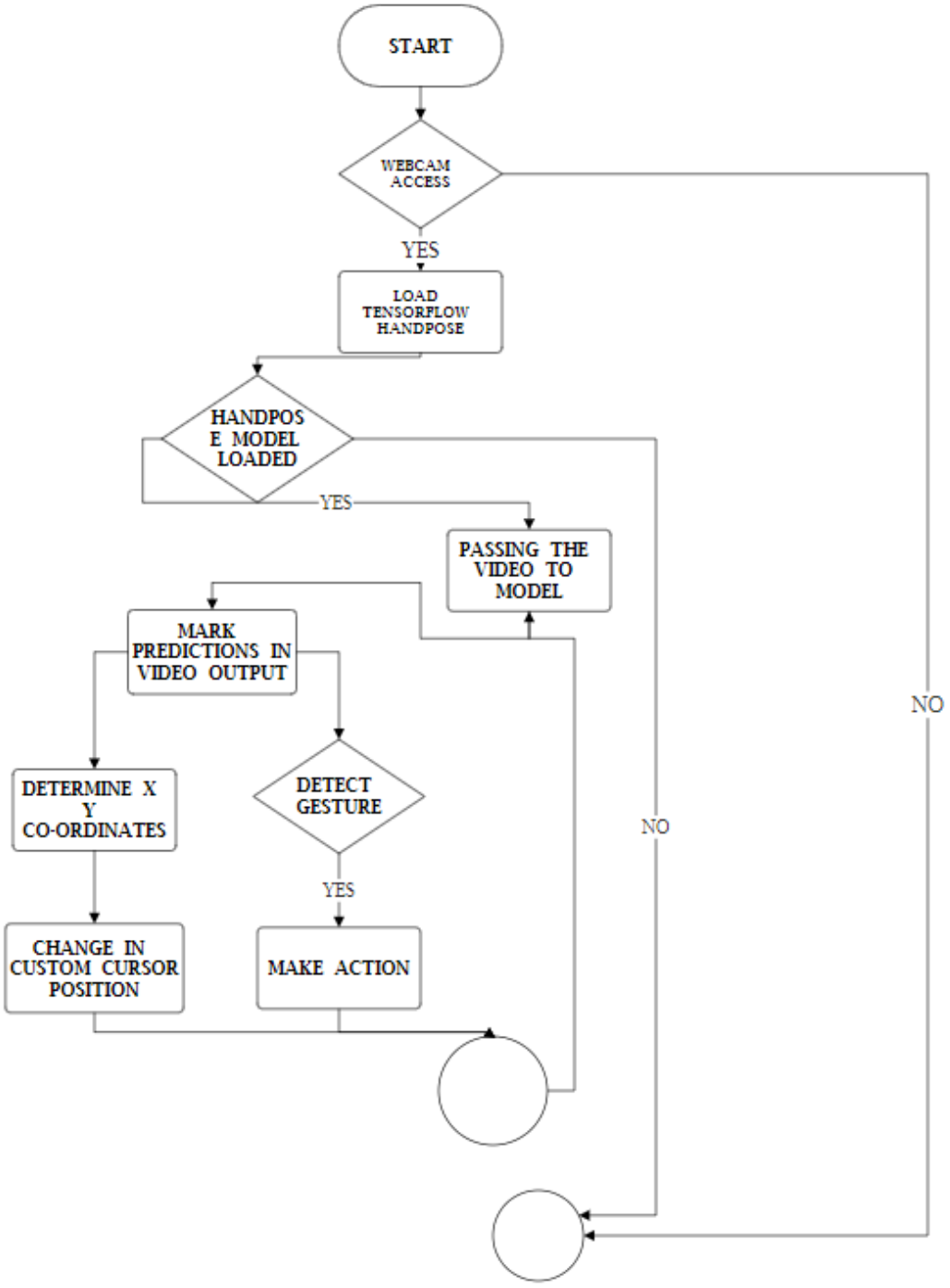
The TensorFlow runtime is a cross-platform library. The system architecture which makes this combination of scale flexible. TensorFlow architecture is appropriate to read and modify the core TensorFlow code.

- **TensorFlow Servable:**
  These are the main TensorFlow serving incomplete units. The objects that the clients utilise to carry out the computation are called servables. A servable can have any size. Anything from a lookup table to a distinct model within a tuple of interface models can be included in a single servable. The Servable allows for adaptability and potential advancements in the future.
- **Servable Versions:**
  During the course of a single server instance, the TensorFlow server is capable of handling one or more servable versions. It provides the ability to load additional data over time, along with different algorithm setups and weights. They can also allow the charging of many versions of a servable simultaneously. Additionally, they enable the simultaneous loading of many servable versions, facilitating a gradual roll-out and experimentation.
- **Servable Streams:**
  A sequence of versions of any servable sorted by increasing version of numbers.
- **TensorFlow Loaders:**
  Loaders manage a servable's life cycle. The loader API enables common infrastructure which is independent of the specific learning algorithm, data, or product use-cases involved.
- **Sources in TensorFlow Architecture:**
  In simple terms, sources are modules that find and provide servable. Each reference provides zero or more servable streams at a time. For each servable stream, a source supplies only one loader instance for every servable.

In a system designed to serve different versions of a model, the process is relatively straightforward. Initially, data sources create loaders for the various model versions, which are then forwarded to a central manager. These loaders include essential information about the models and instructions on how to load them. The manager plays a crucial role in this system by making decisions based on a predefined policy, determining which model versions to load and serve. When a client needs a specific model version or simply the most up-to-date one, they request it from the manager. In response, the manager provides the client with a handle to

the requested model version. For dynamic scenarios, a dynamic manager can step in and load a newer version when there's sufficient memory available. This setup allows clients to efficiently obtain the model version they need, offering flexibility and responsiveness in a dynamic environment.

9. SYSTEM IMPLEMENTATION



The needs for the implementation of this application were taken into consideration when it was being developed, and it was constructed based on the minimal and common requirements that would be available. On the basis of recognizing the hand gestures to control computer cursor, the chrome extension proposed in this research paper is being created. This research paper's proposed application is a browser extension which will operate well on any desktop or laptop computers that uses the Windows operating system.

The following specifications should be met by a machine on which the application will be used. They are:

- **RAM(Random Access Memory):**

  Any machine on which the extension will be installed has to have at least 8GB of RAM so that all of the application's processes can function fast and the user may receive an immediate response from the application.

- **Processor:**

  The processor should be relatively fast to handle the real-time processing demands of gesture recognition. A multi-core processor with a clock speed of at least 2 GHz or higher is recommended to provide the necessary computational power.

- **Webcam:**

  The webcam should have a high-resolution sensor to capture fine details of hand movements. A resolution of at least 720p (1280x720) is a minimum, but higher resolutions, such as 1080p (1920x1080) or 4K (3840x2160), can provide better tracking accuracy. Choose a webcam with good low-light performance to ensure that hand gestures can be detected in various lighting conditions. Look for a webcam with low-light sensitivity and the ability to reduce noise in low-light environments.

- **Storage in the computer:**

  The storage requirements for a Chrome extension that controls cursor movements using hand gestures are relatively modest compared to the hardware and software requirements for gesture recognition. Chrome extensions typically have limited storage capabilities, but it's essential to manage and utilize storage efficiently for your extension to function properly.

SAMPLE CODING

App.js:

```
import React, {useRef, useState, useEffect} from 'react';
import * as tf from "@tensorflow/tfjs";
import * as handpose from "@tensorflow-models/handpose";
import Webcam from 'react-webcam';
import './App.css';
import { drawHand } from './utilities';
import {loveYouGesture} from './LoveYou';
import {RaisedfistGesture} from './RaisedFist';
import * as fp from "fingerpose";
import victory from "./victory.png";
import thumbs_up from "./thumbs_up.png";
import i_love_you from "./i_love_you.png";
import raised_first from "./raised_first.png"


import CustomCursor from './CustomCursor';


function App() {
  const webcamRef = useRef(null);
  const canvasRef = useRef(null);


  const [emoji, setEmoji] = useState(null);
  const images =
{thumbs_up:thumbs_up,victory:victory,i_love_you:i_love_you,raised_fi
rst:raised_first};
  var l1;
  var l2;
  var r1;
  var r2;
  const [xc,upxc] = useState(500);
  const [yc,upyc] = useState(500);
```

```
const runHandpose = async() =>{
  const net = await handpose.load();
  console.log('Handpose model loaded.');
  // Loop and detect hands
  setInterval(() => {
    detect(net);
  },100);
};

const detect = async(net) =>{
  if(
    typeof webcamRef.current !== "undefined" &&
    webcamRef.current !== null &&
    webcamRef.current.video.readyState === 4
  ){
    const video = webcamRef.current.video;
    const videoWidth = webcamRef.current.video.videoWidth;
    const videoHeight = webcamRef.current.video.videoHeight;
    webcamRef.current.video.width = videoWidth;
    webcamRef.current.video.height = videoHeight;
    canvasRef.current.width = videoWidth;
    canvasRef.current.height = videoHeight;
    const hand = await net.estimateHands(video);
    if(hand.length > 0){
      l1 = Math.floor(hand[0].landmarks[9][0]);
      l2 = Math.floor(hand[0].landmarks[9][1]);
      r1 = (l1/640)*1350;
      r2 = (l2/640)*1100;
      document.querySelector('.custom-cursor').style.left=r1+"px";
      document.querySelector('.custom-cursor').style.top=r2+"px";
      const GE = new fp.GestureEstimator([
        loveYouGesture,
        RaisedfistGesture
      ]);
      const gesture = await GE.estimate(hand[0].landmarks, 4);
```

```
        if(gesture.gestures !== undefined && gesture.gestures.length
> 0){
          // ||||console.log(gesture.gestures);
          const confidence = gesture.gestures.map(
            (prediction) => prediction.score
          );
          const maxConfidence = confidence.indexOf(
            Math.max.apply(null, confidence)
          );
          setEmoji(gesture.gestures[maxConfidence].name);
          console.log(gesture.gestures[maxConfidence].name);
          var temp = gesture.gestures[maxConfidence].name;
          if (gesture.gestures[maxConfidence].name ===
"raised_first"){
            document.elementFromPoint(r1, r2).click();
          }
        }
      }
      const ctx = canvasRef.current.getContext("2d");
      drawHand(hand, ctx);
    }
    upxc(l1);
    upyc(l2);
  }
  useEffect(()=>{runHandpose()},[]);
  return (
    <>
    <div className="App">
    {/* <h1>NETFLIX</h1>
    <div class="27etflix-slider">
      <h2>Popular on Netflix</h2>
      <div class="swiper-container">
        <div class="swiper-wrapper">
        <div class="swiper-slide"><img src="./img/1.jpg" alt="Movie
Title"></img></div>
```

```
        <div class="swiper-slide"><img src="./img/2.jpg" alt="Movie
Title"></img></div>

        <div class="swiper-slide"><img src="./img/3.jpg" alt="Movie
Title"></img></div>

        <div class="swiper-slide"><img src="./img/4.jpg" alt="Movie
Title"></img></div>

        <div class="swiper-slide"><img src="./img/5.jpg" alt="Movie
Title"></img></div>

        <div class="swiper-slide"><img src="./img/6.jpg" alt="Movie
Title"></img></div>

        <div class="swiper-slide"><img src="./img/7.jpg" alt="Movie
Title"></img></div>

        <div class="swiper-slide"><img src="./img/8.jpg" alt="Movie
Title"></img></div>

        <div class="swiper-slide"><img src="./img/9.jpg" alt="Movie
Title"></img></div>

        <div class="swiper-slide"><img src="./img/10.jpg" alt="Movie
Title"></img></div>

        </div>

        <div class="swiper-button-next"></div>

        <div class="swiper-button-prev"></div>

      </div>

    </div>

    <div class="28etflix-slider">

      <h2>Continue watching…</h2>

      <div class="swiper-container">

      <div class="swiper-wrapper">

      <div class="swiper-slide"><img src="./img/1.jpg" alt="Movie
Title"></img></div>

        <div class="swiper-slide"><img src="./img/2.jpg" alt="Movie
Title"></img></div>

        <div class="swiper-slide"><img src="./img/3.jpg" alt="Movie
Title"></img></div>

        <div class="swiper-slide"><img src="./img/4.jpg" alt="Movie
Title"></img></div>

        <div class="swiper-slide"><img src="./img/5.jpg" alt="Movie
Title"></img></div>

        <div class="swiper-slide"><img src="./img/6.jpg" alt="Movie
Title"></img></div>

        <div class="swiper-slide"><img src="./img/7.jpg" alt="Movie
Title"></img></div>
```

```
        <div class="swiper-slide"><img src="./img/8.jpg" alt="Movie
Title"></img></div>
        <div class="swiper-slide"><img src="./img/9.jpg" alt="Movie
Title"></img></div>
        <div class="swiper-slide"><img src="./img/10.jpg" alt="Movie
Title"></img></div>
      </div>
      <div class="swiper-button-next"></div>
      <div class="swiper-button-prev"></div>
      </div>
    </div>
    <script src="./package/js/swiper.min.js"></script> */}
    {/* <script src='package/js/own.js'></script> */}
        {/* <img src={logo} className="App-logo" alt="logo" />
        <p>
          Edit <code>src/App.js</code> and save to reload.
        </p>
        <a
          className="App-link"
          href="https://reactjs.org"
          target="_blank"
          rel="noopener noreferrer"
        >
          Learn React
        </a> */}
        <Webcam ref={webcamRef}
        style={{
          position:"absolute",
          marginLeft:"auto",
          marginRight:"auto",
          left:0,
          right:0,
          zindex:9,
          width:200,
          height:200
        }}/>
```

```
        <canvas
          ref={canvasRef}
          style={{
            position:"absolute",
            marginLeft:"auto",
            marginRight:"auto",
            left:0,
            right:0,
            zindex:9,
            width:200,
            height:200,
            display: 'none'
          }}
        />
        {emoji !==null ? <img src={images[emoji]} style={{
          position:"absolute",
          marginLeft:"auto",
          marginRight:"auto",
          left:0,
          right:0,
          height:100,
          zIndex:9
        }} />:""}
      <CustomCursor xx={xc} yy={yc}/>
      {/* <CustomCursor xx={l1} /> */}
    </div>
    </>
  );
}
export default App;
```

CustomCursor.css:

```
/* CustomCursor.css */
.custom-cursor {
    width: 20px;
```

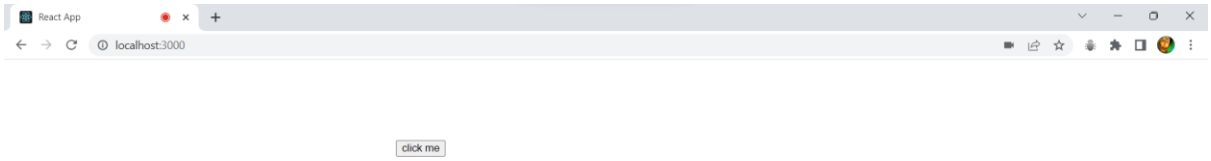
```
    height: 20px;
    background-color: red;
    position: absolute;
    border-radius: 50%;
    pointer-events: none;
    transform: translate(-50%, -50%);
    transition: left 0.1s ease-in-out;
    transition: top 0.1s ease-in-out;
  }
```

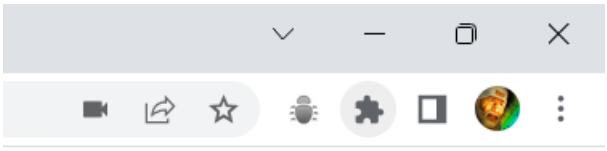
Index.html:

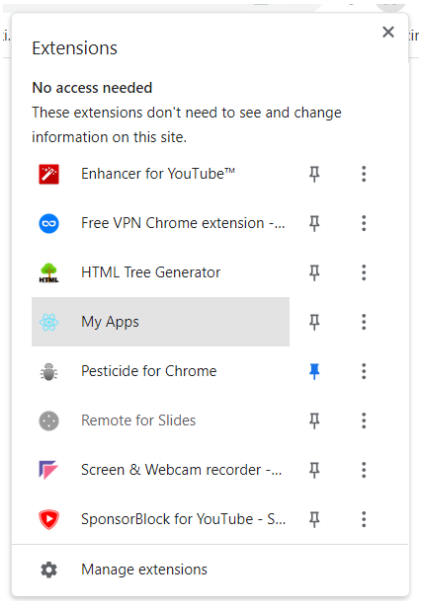
```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <link rel="icon" href="%PUBLIC_URL%/favicon.ico" />
    <meta name="viewport" content="width=device-width, initial-
scale=1" />
    <meta name="theme-color" content="#000000" />
    <meta
      name="description"
      content="Web site created using create-react-app"
    />
    <link rel="apple-touch-icon" href="%PUBLIC_URL%/logo192.png" />
    <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />
    <title>React App</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1, minimum-scale=1, maximum-scale=1">
    <link rel="stylesheet" href="./css/styles.css"> →
  </head>
  <body>
    <noscript>You need to enable JavaScript to run this
app.</noscript>
    <button onclick="alert('Hello world')" style="margin-left:
500px; margin-top: 102px;">click me</button>
        <div id="root">
```

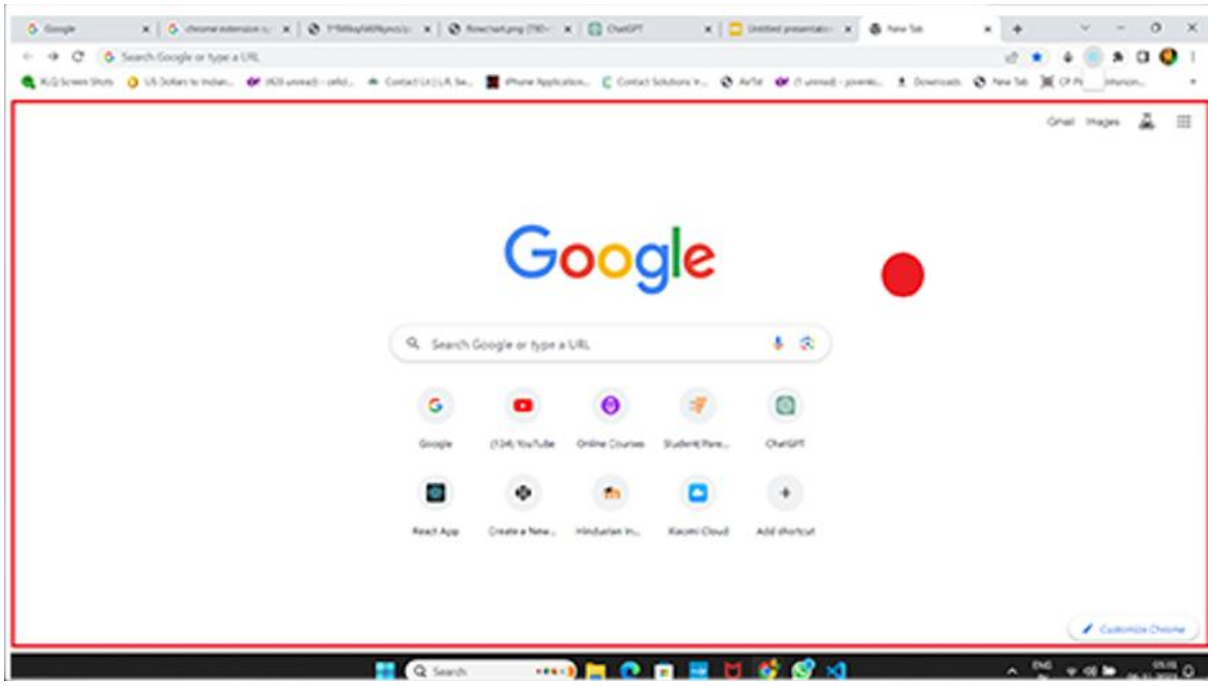
```
    </div>
  </body>
</html>
```

SAMPLE SCREENSHOTS

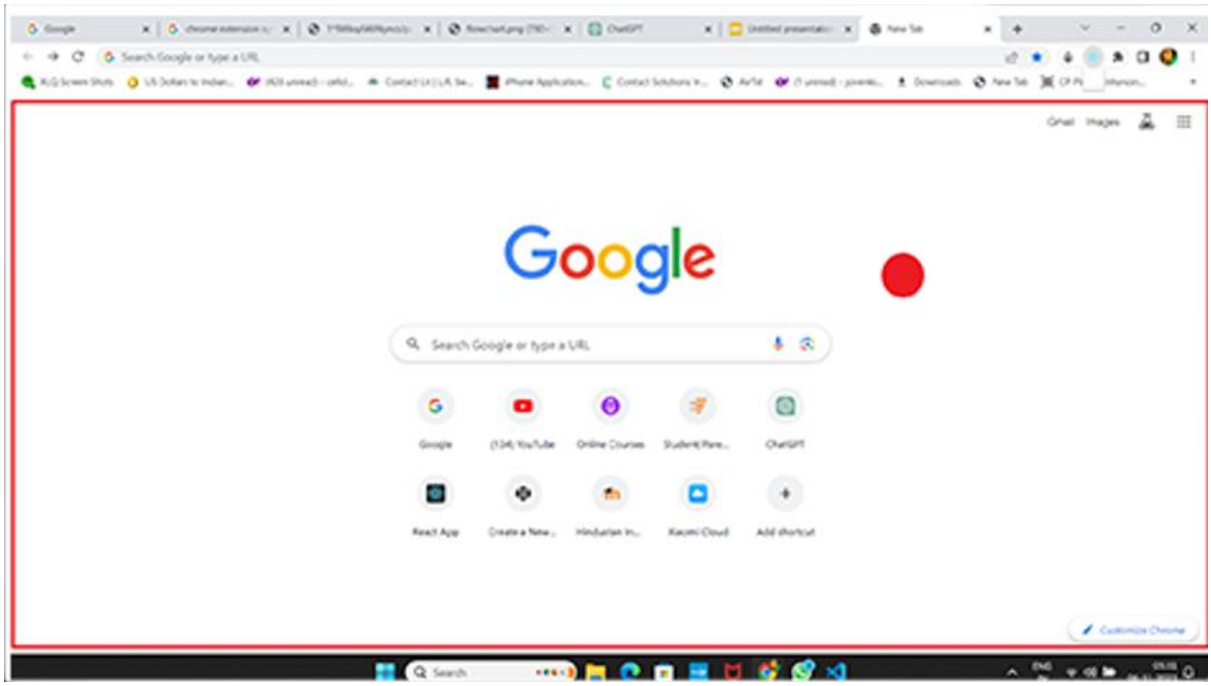







Google
Search Google or type a URL

10. RESULT AND PERFORMANCE ANALYSIS



The software provides an impressive solution to enhance cursor control, navigation, and content selection in web browsers. It effectively simplifies the browsing experience by offering advanced capabilities that boost efficiency and productivity. With its user-friendly interface and customizable settings, it caters to a wide range of users.

The evaluation confirmed the tool's reliability and stability. It seamlessly integrates with the browser, ensuring a smooth and responsive performance. Users can consistently expect frustration-free interactions with web content, whether they have specific accessibility needs or are simply seeking a more intuitive and efficient browsing experience.

The software's cursor control features offer precision and adaptability. Users can easily adjust cursor speed and sensitivity for tasks like text selection and clicking on small elements. Customizable keyboard shortcuts significantly improve web page navigation, enabling users to move swiftly between links, headings, and other page elements. This not only reduces the need for excessive scrolling but also enhances overall browsing efficiency.

Moreover, the tool's content selection features greatly improve web content manipulation. Users can effortlessly select text, images, and various elements, simplifying tasks like copying,

pasting, dragging, and dropping. Multiple selection modes cater to different use cases, making the technology highly versatile.

This technology provides a comprehensive solution for enhancing cursor control, navigation, and content selection in web browsers. Its combination of user-friendliness, customization options, and robust performance makes it a valuable addition to the browser ecosystem. Whether used by individuals with specific accessibility needs or those looking to optimize their browsing experience, this tool excels at providing an efficient and satisfying solution. It streamlines tasks, saves time, and reduces frustration, making it an essential tool for enhancing web interactions.

The updated improvements and features are working properly. Two hand can be used to control the custom cursor. The usage of updated tensorflow makes the extension more efficient.

11. CONCLUSION AND FUTURE WORK

The creation of a Chrome plugin that permits hand movements to control the cursor is an intriguing and creative method to engage with the virtual environment. This technology offers a touchless, user-friendly interface for accessing webpages, applications, and more, with the potential to completely transform user experiences. This plugin creates new opportunities for gaming, productivity, and accessibility by detecting and interpreting hand movements. The idea of improving user engagement in a smooth and organic way is intriguing, even though there might be difficulties with accuracy and user training. We may anticipate seeing even more advancements in the sophistication and usability of this Chrome addon as gesture detection technology develops. In the future, these extensions might be useful not only for people with physical limitations but also for augmented and virtual reality, as well as for a number of other industries, including entertainment, education, and design. All things considered, the Chrome extension that uses hand gestures to control cursor movements has a lot of potential for improving digital accessibility and human-computer interaction in the future.

The integration of multiple sensing methods, including hand gestures, voice recognition, eye-tracking, and other sensor technologies, is aiming to create a comprehensive, context-aware interface that responds to users' natural cues. The enhancement of machine learning algorithms is crucial for the technology's ability to accurately understand and respond to user gestures. The new "@tensorflow/hand-pose-detection" model will improve user-computer interaction by spotting both hands and simplifying settings for developers.

The focus is on expanding the range of recognized hand gestures, allowing users to communicate using a broader set of gestures. Real-world validation and user feedback are essential for continuous improvements. Advanced encryption methods and data protection mechanisms are being explored to ensure user privacy.

Customization for specific domains is an exciting direction, with specialized gesture sets and interaction models being developed for healthcare, gaming, education, and other fields. Accessibility is a core focus, with a commitment to developing custom gesture sets and adaptable interfaces.

The vision also extends to the seamless integration of the technology with the Internet of Things (IoT) and smart home systems. The future of hand gesture-based control in human-robot interactions is promising, with the aim of reshaping human-computer interaction.

12. REFERENCES:

1. Kumaran, Natarajan & Surya, Balina & Vamsi, Bobba. (2023). WEB BROWSER CONTROL USING HAND GESTURES. 7. 10.55041/IJSREM21244.
2. R. Matlani, R. Dadlani, S. Dumbre, S. Mishra and A. Tewari, "Virtual Mouse using Hand Gestures," 2021 International Conference on Technological Advancements and Innovations (ICTAI), Tashkent, Uzbekistan, 2021, pp. 340-345, doi: 10.1109/ICTAI53825.2021.9673251.
3. Vimali, J.S., Srinivasulu, S., Jabez, J., Gowri, S. (2021). Hand Gesture Recognition Control for Computers Using Arduino. In: Jeena Jacob, I., Kolandapalayam Shanmugam, S., Piramuthu, S., Falkowski-Gilski, P. (eds) Data Intelligence and Cognitive Informatics. Algorithms for Intelligent Systems. Springer, Singapore. DOI: 10.1007/978-981-15-8530-2_45
4. Pralhad, G.P., Abhishek, S., Kachare, T., Deshpande, O., Chounde, R., Tapadiya, P. (2021). Web-Based Real-Time Gesture Recognition with Voice. In: Bhattacharya, M., Kharb, L., Chahal, D. (eds) Information, Communication and Computing Technology. ICICCT 2021. Communications in Computer and Information Science, vol 1417. Springer, Cham. DOI: 10.1007/978-3-030-88378-2_10
5. Persichella, G., Lomanto, C. L., Mattutino, C., & Gena, C. (2019b). Experimenting touchless gestural interaction for a university public web-based display. ResearchGate. https://www.researchgate.net/publication/340414784_Experimenting_Touchless_Gestural_Interaction_for_a_University_Public_Web-based_Display
6. Casarini, M., Porta, M., & Dondi, P. (2020). A Gaze-Based Web Browser with Multiple Methods for Link Selection. ACM Symposium on Eye Tracking Research and Applications. DOI: 10.1145/3379157.3388929
7. Rama Gottfried, Georg Hajdu.(2019) Drawsocket: a browser based system for networked score display". (n.d.). CORE Reader. https://core.ac.uk/reader/227003780
8. F. Larradet, G. Barresi and L. S. Mattos, "Design and Evaluation of an Open-source Gaze-controlled GUI for Web-browsing," 2019 11th Computer Science and Electronic Engineering (CEEC), Colchester, UK, 2019, pp. 18-23, doi: 10.1109/CEEC47804.2019.8974342.
9. Khan, R. Z. (2012). Hand Gesture Recognition: A literature review. *International Journal of Artificial Intelligence & Applications*, *3*(4), 161–174. Doi: 10.5121/ijaia.2012.3412

10. Mahapatra, A. P., Sukla, B., Harikrishnan, K. M., Mishra, D., & Salkuti, S. R. (2022). Error detection and comparison of gesture control technologies. *IAES International Journal of Artificial Intelligence*, *11*(2), 709. Doi:10.11591/ijai.v11.i2.pp709-716
11. N. Jangamreddy, V. Sethi and S. Pal, "Web-Based Gesture Recognition System for Controlling Heterogeneous IoT devices using Deep Learning," 2019 11th International Conference on Communication Systems & Networks (COMSNETS), Bengaluru, India, 2019, pp. 700-702, doi: 10.1109/COMSNETS.2019.8711158.
12. Shin, C., Ng, T.N. Integrated devices that can recognize hand gestures. *Nat Electron* **6**, 555–556 (2023). doi:10.1038/s41928-023-01003-0
13. Elmagrouni, I., Ettaoufik, A., Aouad, S., & Maizate, A. (2021). Approach for improving user interface based on gesture recognition. *E3S Web of Conferences*. doi:10.1051/e3sconf/202129701030
14. M. Raja., P. Nagaraj, P. Sathwik, K. M. A. Khan, N. M. Kumar and U. S. Prasad, "Voice Assistant and Gesture Controlled Virtual Mouse using Deep Learning Technique," 2023 International Conference on Sustainable Computing and Data Communication Systems (ICSCDS), Erode, India, 2023, pp. 156-161, doi: 10.1109/ICSCDS56580.2023.10104619.
15. Islam, S., Matin, A., Kibria, H.B. (2022). Hand Gesture Recognition Based Human Computer Interaction to Control Multiple Applications. In: Vasant, P., Zelinka, I., Weber, GW. (eds) Intelligent Computing & Optimization. ICO 2021. Lecture Notes in Networks and Systems, vol 371. Springer, Cham.doi:10.1007/978-3-030-93247-3_39
16. Damdoo, R., & Gupta, A. (2022). Gesture controlled interaction using hand pose model. *International Journal of Health Sciences (IJHS)*, 10417–10427. doi:10.53730/ijhs.v6ns1.7493
17. M. Yunus, K. S. Simha, J. Refonaa, S. L. J. Shabu, S. Dhamodaran and V. A. Mary, "Hand-Gesture and Voice-Activated Digital Cursor," 2023 2nd International Conference on Vision Towards Emerging Trends in Communication and Networking Technologies (ViTECoN), Vellore, India, 2023, pp. 1-5, doi: 10.1109/ViTECoN58111.2023.10157786.
18. I. El Magrouni, A. Ettaoufik, A. Siham, A. Maizate and B. Lotfi, "Approach for the construction of gestural interfaces to control graphical interfaces based on artificial intelligence," 2022 9th International Conference on Wireless Networks and Mobile Communications (WINCOM), Rabat, Morocco, 2022, pp. 1-6, doi: 10.1109/WINCOM55661.2022.9966424.

19. B. B, D. R, V. T. Selvam, V. V. Kumar and R. R, "Improved Real-Time Approach to Static Hand Gesture Recognition," 2021 11th International Conference on Cloud Computing, Data Science & Engineering (Confluence), Noida, India, 2021, pp. 416-422, doi: 10.1109/Confluence51648.2021.9377036.
20. C. J. Cohen, G. Beach and G. Foulk, "A basic hand gesture control system for PC applications," Proceedings 30th Applied Imagery Pattern Recognition Workshop (AIPR 2001). Analysis and Understanding of Time Varying Imagery, Washington, DC, USA, 2001, pp. 74-79, doi: 10.1109/AIPR.2001.991206.
21. Huang, C., Li, J., Yu, H., Hanchao, J., & Jiang-Hai, S. (2013). Gesture browsing method for geological data. *Advances in Intelligent Systems Research*. DOI: 10.2991/icmt-13.2013.64
22. Wachs, J. P., Stern, H. I., Edan, Y., Gillam, M., Feied, C. F., Smithd, M., & Handler, J. (2008). Real-Time hand gesture interface for browsing medical images. *International Journal of Intelligent Computing in Medical Sciences and Image Processing*, *2*(1), 15–25. DOI: 10.1080/1931308x.2008.10644149
23. N, Umadevi & Ir, Divyasree. (2016). Development of an Efficient Hand Gesture Recognition system for human computer interaction. International Journal Of Engineering And Computer Science. 10.18535/Ijecs/v4i12.5.
24. Binh, N. D. (2016). Gestures Recognition from Sound Waves. *EAI Endorsed Transactions on Context-aware Systems and Applications*, *3*(10), 151679. DOI: 10.4108/eai.12-9-2016.151679
25. Peng, Chao & Hansberger, Jeff & Cao, Lizhou & Areyur Shanthakumar, Vaidyanath. (2017). Hand gesture controls for image categorization in immersive virtual environments. 331-332. 10.1109/VR.2017.7892311.