COMPLETE
Journal of Computer, Electronic, and Telecommunication
ISSN 2723-4371, E-ISSN 2723-5912

Article

Development of a Real-Time Color Blindness Correction Progressive Web App (PWA) Using Daltonization Methods and WebGL Hardware Acceleration

Gilbert Aldrich Rumahorbo[1], Nazwa Aidilia Octa Mevia[2]*, Yohana Kartika Marbun[3], Debi Yandra Niska[4,]*

[1, 2, 3, 4] Department of Computer Science, Universitas Negeri Medan, Medan, Indonesia
* Correspondence: nazwaaidila9@gmail.com and debiyandraniska@unimed.ac.id

Received: 29 April 2026; Revised: 5 May 2026; Accepted: 10 June 2026; Published: 20 July 2026.

Abstract: Color Vision Deficiency (CVD) is a visual sensory disorder that affects an individual's ability to distinguish certain colors, particularly in the red green and blue yellow spectrum. This condition can limit accessibility to digital visual information such as user interfaces, graphics, and navigation systems. This study aims to develop a web-based real-time color correction application by integrating the Daltonization algorithm with WebGL technology in a Progressive Web App (PWA) architecture. The system utilizes GPU acceleration through fragment shaders to perform parallel image processing, resulting in improved computational efficiency, achieving processing speeds of 35-57 FPS—a 280% to 380% increase compared to conventional CPU-based approaches. The implementation of color transformation in the LMS color space enhances visual contrast, enabling users with CVD to better differentiate colors. The results indicate that the proposed system is capable of delivering responsive performance with stable real-time processing, while maintaining accessibility and cross-platform usability. This research contributes to the development of inclusive digital technologies by providing an efficient and practical solution for color vision accessibility in modern web applications**.** The main contribution of this research lies in the novel integration of the Daltonization algorithm entirely within the GPU fragment shaders of a PWA, enabling zero-latency, client-side color correction that eliminates server dependencies and safeguards user privacy.

Keywords: Color Vision Deficiency; Daltonization; WebGL; Real-Time Processing; Accessibility

1. Introduction

Color Vision Deficiency (CVD) is a visual sensory anomaly caused by the dysfunction of cone cells in the retina, which inhibits color perception, particularly in the red-green and blue-yellow spectrums. Globally, this condition affects over 300 million individuals, directly limiting their accessibility to digital visual information such as graphics and application interfaces [1]. In Indonesia, the challenges for CVD sufferers extend to the social and professional sectors, where color vision limitations often become a barrier in education selection and employment within government agencies and State-Owned Enterprises (BUMN). This phenomenon emphasizes the urgency of developing inclusive technologies capable of adapting digital content to meet the needs of those with visual [1], [2].

The lack of attention to accessibility principles in user interface design leads to a significant decline in user experience for CVD sufferers. Obstacles in distinguishing color-based navigation elements and iconography can hinder effective information transfer within complex digital ecosystems [3]. As a solution, the Daltonization method serves as an image processing technique that shifts color information from an undetectable spectrum to a spectrum that can be distinguished by

CVD patients [4]. This process is generally implemented in the LMS color space to replicate the eye's physiological response, which has been proven to significantly increase visual contrast [4], [5], [6].

Despite its theoretical effectiveness, the implementation of Daltonization on conventional web platforms is often constrained by CPU performance limitations. Frame-by-frame dynamic image processing on the CPU results in high latency and inefficient power consumption, thereby hindering real-time application [7]. To overcome these limitations, leveraging hardware acceleration through the Graphics Processing Unit (GPU) using WebGL technology becomes a crucial solution. Through the use of fragment shaders, complex color transformations can be executed in parallel with high efficiency, enabling better system responsiveness on user devices [8], [9].

This research proposes the development of a web-based real-time color correction application that integrates Daltonization algorithms with a WebGL processing pipeline within a Progressive Web Apps (PWA) architecture [10]. Unlike static solutions, this study optimizes LMS color space transformations through shader-based processing to achieve a balance between correction accuracy and computational performance [11]. The results of this research are expected to provide a tangible contribution to the application of universal design principles in information systems, ensuring equal digital accessibility for all users without compromising system functionality. The primary contributions of this paper are threefold: (1) The mathematical formulation of a hardware-accelerated Daltonization pipeline using WebGL fragment shaders for dynamic video processing; (2) The development of a decoupled PWA architecture that operates entirely on the client-side, mitigating cloud server latencies; and (3) An empirical evaluation demonstrating significant improvements in both computational frame rates and user color recognition accuracy in real-world environments.

2. Materials and Methods

*T*his research designs and develops a software application based on Progressive Web Apps (PWA) that functions as a real-time color blindness correction tool. The system development focuses on optimizing client-side edge computing without relying on external servers or relational databases, aiming to minimize data transmission latency and maintain user privacy. To achieve this, the proposed methodology systematically processes visual data through four sequential computational stages: (1) hardware-level frame acquisition via WebRTC, (2) physiological LMS color space transformation, (3) Daltonization error tensor calculation for deficiency simulation, and (4) RGB reconstruction and canvas rendering via GPU acceleration. This systematic pipeline ensures high-fidelity color compensation without structural degradation.

2.1 System Development Method

The Software Development Life Cycle (SDLC) in this study implements the Agile method with the Scrum framework. Unlike traditional sequential development methods that tend to be rigid, Agile Scrum facilitates a highly iterative and incremental approach through measurable development cycles called Sprints. This Agile approach provides developers with the flexibility to continuously perform code architecture modifications (shader tweaking) and immediately test their impact on framerate stability and hardware memory latency before the final feature release [12].

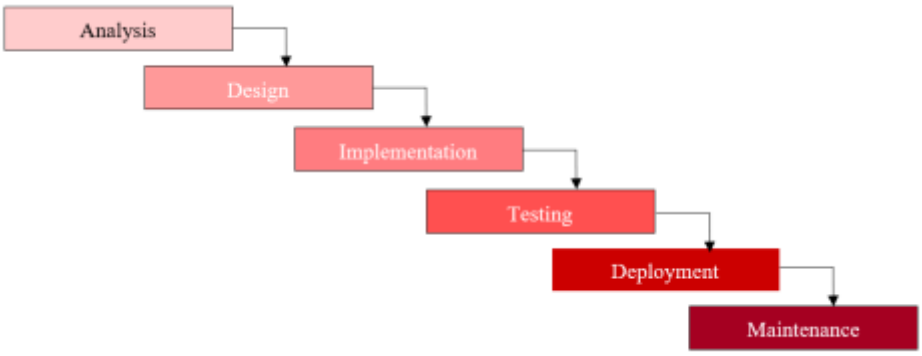


Figure 1. Agile Scrum System Development Stages

Based on Figure 1, the system development process using the Agile Scrum method consists of several main stages: analysis, design, implementation, testing, deployment, and maintenance. Each stage is conducted iteratively within sprint cycles, allowing developers to perform continuous evaluation and improvement. The analysis phase focuses on identifying system requirements, followed by architectural planning in the design phase. Subsequently, implementation is carried out by developing system features, including shader optimization for the color correction process. The testing phase aims to ensure system performance, particularly in maintaining frame rate stability and processing efficiency. Once the system is declared stable, deployment is executed, concluding with maintenance for continuous enhancement.

2.2 System Design

The system design is conducted using an analytical and modeling approach with the Unified Modeling Language (UML) as the primary instrument. UML was selected for its capability to represent the system architecture and client-side application functionalities in a structured and visual manner. This facilitates the requirements analysis process, the architectural design of graphical processing, and effective communication among the development team[13], [14] The system modeling encompasses several key diagrams as follows:

2.2.1 Use Case Diagram

Use case diagrams are utilized to define the system boundaries and the interactions between the actor (User) and the primary application functions. These include granting webcam access permissions, selecting Color Vision Deficiency (CVD) anomaly modes, adjusting correction intensity, [13]. Based on Figure 2, the Use Case Diagram illustrates a system architecture where a single User interacts with seven core functionalities designed for real-time color blindness correction. The interaction begins with Granting Camera Permission, which is essential for the system to acquire the live video stream. Once access is established, the user can navigate through various operational features, such as changing correction/simulation modes to suit specific vision needs, Adjusting Intensity and Split screen for visual comparison, and turning on the flashlight/torch to assist in low-light environments. Furthermore, the system allows the user to document their experience by Saving Snapshot results and provides an educational layer through the Reading the Education module, ensuring a comprehensive toolset for both practical assistance and informed usage.

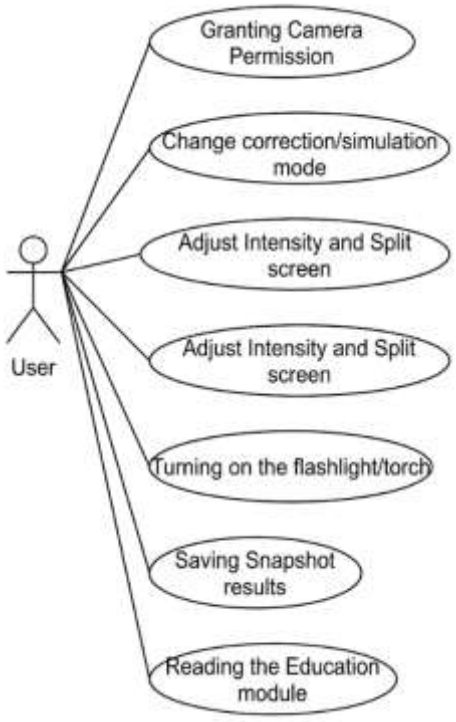


Figure 2. Use Case Diagram of the Real-Time Color Blindness Correction System

2.2.2 Activity Diagram

Present the entire sequential flow of system activities, starting from the PWA initialization and hardware permission validation, through the video stream capture cycle and GPU algorithm execution, to the final frame rendering on the canvas. This diagram is essential for verifying asynchronous logic and identifying potential overlaps within the processing activities [14].

Based on Figure 3, the system's procedural flow initiates with the Progressive Web App (PWA) initialization, which automatically triggers a camera access permission request. At the decision node, if authorization is denied, the system displays an error message and terminates all processes permanently. Conversely, if access is granted, the system simultaneously executes two main activity paths that run asynchronously:

First, the system enters a continuous graphical processing cycle (Render WebGL/GPU Loop) to maintain real-time preview fluidity. This path involves downloading video frames, executing shader processes on the Graphics Processing Unit (GPU), rendering corrected results onto the Canvas element, and calculating the frame rate (calculate FPS) to monitor system performance.

Second, the system actively monitors user interactions through the control panel (UI Interaction Monitoring). Users can modify color deficiency modes, perform color analysis via screen touch, capture screenshots, or activate the flashlight. Each interaction triggers a parameter update that is instantly integrated into the shader processing cycle without interrupting the video stream. This entire sequence repeats continuously until the user decides to close the application.

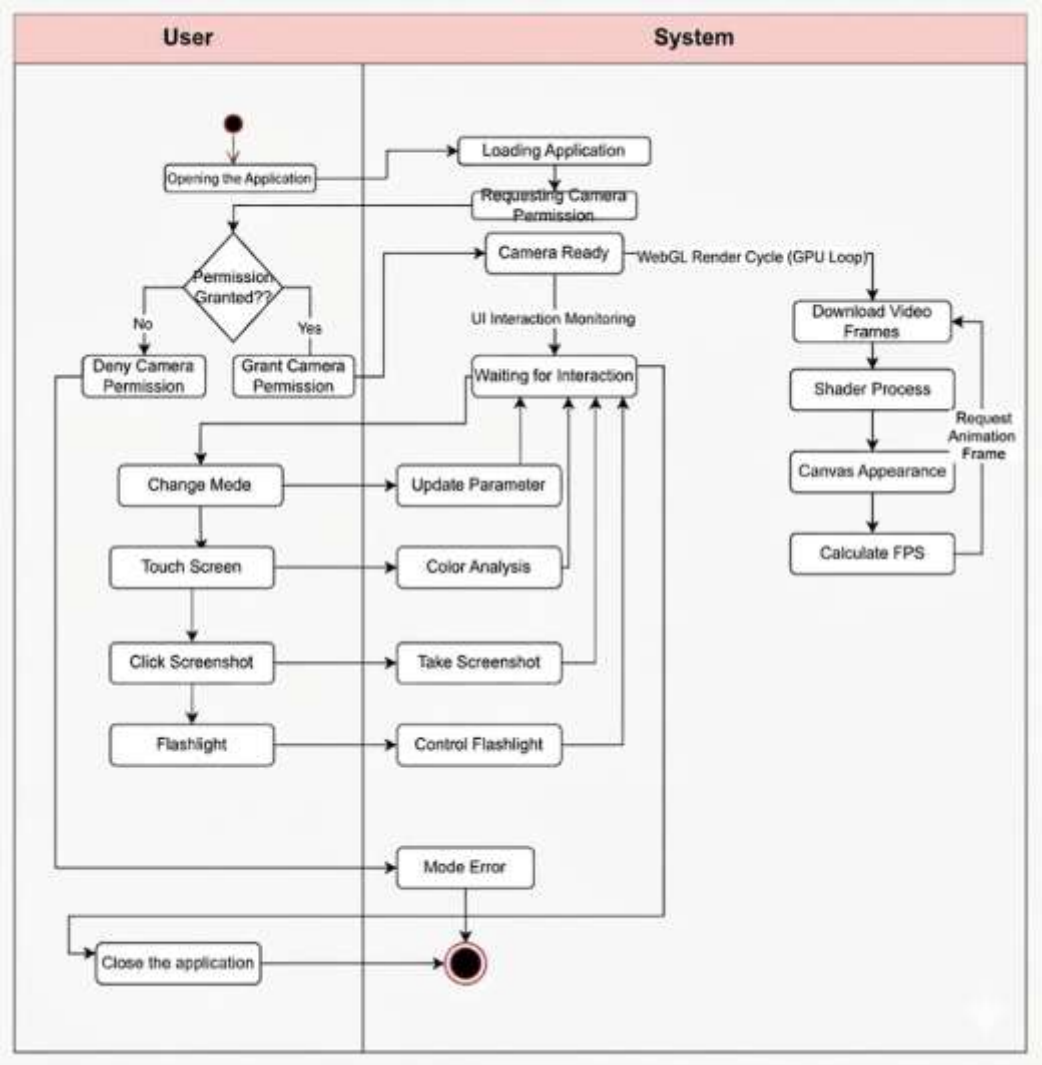


Figure 3. Activity Diagram of the Real-Time Color Blindness Correction System.

2.2.3 Class Diagram

Reused to model the modular architectural structure and the relationships between the core components of the system, such as UIController, CameraManager, WebGLRenderer, and DaltonizationShader [14].

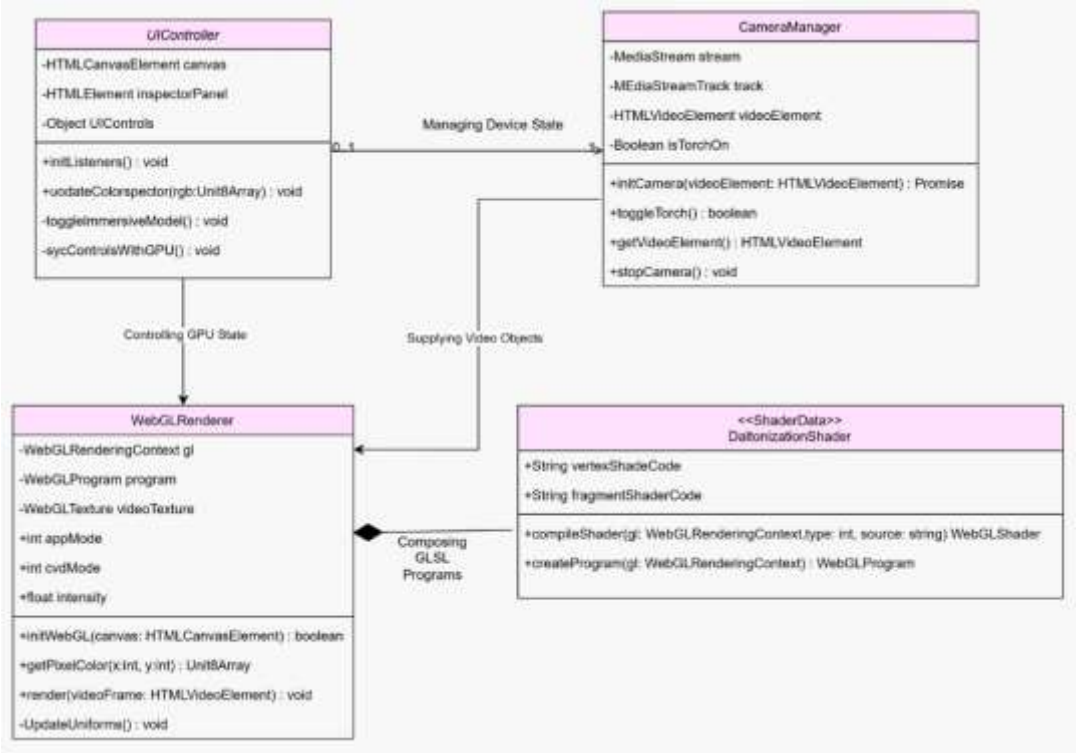


Figure 4. Class Diagram of the Real-Time Color Blindness Correction System.

Based on Figure 4, the software architecture is designed using the Separation of Concerns principle, orchestrated by four primary classes that collaborate asynchronously to ensure high modularity. The UIController manages the interface state and user inputs, such as CVD anomaly types and intensity levels, which are synchronized with the GPU via the WebGLRenderer. Simultaneously, the CameraManager abstracts the WebRTC API to acquire live video streams and manage hardware states like the camera torch, while the DaltonizationShader acts as a dedicated data provider for storing and compiling GLSL vertex and fragment shader codes. Ultimately, the WebGLRenderer functions as the system's core engine, integrating the dynamic video textures from the camera with the compiled shader programs to execute high-performance rendering on the HTML5 canvas in real-time.

2.3 Image Processing Workflow and User Interaction

A more in-depth logic of data processing including the integration of pixel analytics features and static image processing is explained through a system flowchart. This provides technical guidance for the implementation of image processing algorithms on the client-side.

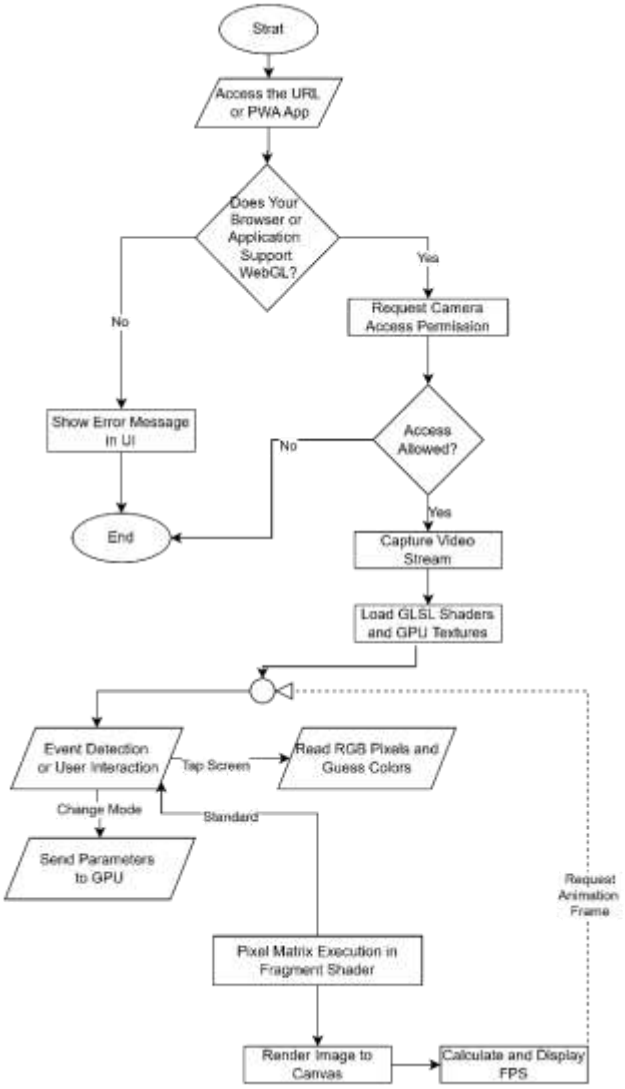


Figure 5. Flowchart of Image Processing Logic and Dynamic Interaction

Based on Figure 5, the system's logical flow is built upon an event-driven architecture centered on a continuous, GPU-accelerated render loop. The initialization phase begins with a hardware capability validation by checking for WebGL support within the browser. If this validation fails or if the user denies webcam access permissions, the system triggers a formal error handling routine (Show Error Message) and terminates the entire process flow (Terminator End).

Once authorization is granted, the application proceeds to load the GLSL Shaders and GPU Textures before entering an asynchronous processing cycle bridged by an On-Page Connector. Within this loop phase, the system operates in a non-blocking manner utilizing the request Animation Frame mechanism. User interactions whether modifying parameters via Change Mode to update GPU uniform variables or performing a Tap Screen action to extract raw RGB values and identify colors are processed simultaneously.

The core of this visual computation resides in the Pixel Matrix Execution within the Fragment Shader. At this stage, pixel-level color manipulation is performed before being projected onto the UI canvas. This interactive cycle runs in parallel with real-time FPS calculation to ensure performance stability. The flow continuously recurses to the connector point throughout the session until the user exits the application.

2.4 . Mathematical Formulation of the Daltonization Algorithm

To achieve high-fidelity visual compensation, the real-time image transformation pipeline within this system is engineered upon a sequence of linear algebraic matrices that mathematically replicate human visual physiology. In conventional web-based computer vision frameworks, processing dynamic video streams *frame-by-frame* on the Central Processing Unit (CPU) heavily strains the system's main execution thread, inevitably triggering severe frame drops, high latencies, and rapid thermal throttling on mobile client devices.

To bypass these computational bottlenecks, this system completely decouples the mathematical simulation and correction routines from the CPU, delegating the pixel-level array calculations to the Graphics Processing Unit (GPU) via the WebGL API. By leveraging custom-built fragment shaders compiled via the OpenGL Shading Language (GLSL), complex three-dimensional color space transformations, matrix projections, and localized error distributions are executed in parallel across millions of pixels simultaneously. This hardware-accelerated pipeline guarantees that every incoming video frame is fully manipulated, optimized, and reconstructed within an execution window of less than 15 milliseconds, successfully maintaining rendering fluidities between 35 and 57 Frames Per Second (FPS) directly inside the browser environment.

2.4.1 RGB to LMS Color Space Conversion

The process begins by normalizing the raw image pixel values from the RGB space (R, G, B) into the range of $[0.0, 1.0]$. Subsequently, the sRGB vector is transformed into the LMS (Long, Medium, Short wavelength) color space using the Hunt-Pointer-Estevez (HPE) transformation matrix under the standard D65 illuminant:

$$\begin{bmatrix} L \\ M \\ S \end{bmatrix} = \begin{bmatrix} 17.8824 & 43.5161 & 4.11936 \\ 3.45565 & 27.1554 & 3.86714 \\ 0.02996 & 0.8431 & 1.46709 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \quad (1)$$

The standard RGB color model represents colors according to the characteristics of electronic display systems rather than the physiological processes of human vision. Therefore, pixel values are transformed into the LMS color space, where each component corresponds to the response of the retinal cone cells. This transformation enables the algorithm to manipulate color information based on the actual neurological stimulation experienced by the human visual system, resulting in a more perceptually accurate color correction process.

2.4.2 Color Vision Deficiency Simulation

To isolate the exact details lost by a user with color anomalies, the system projects a spectral reduction. For green-cone deficiency (*Deuteranopia*), the anomalous medium-wavelength component (M_{Sim})is calculated by projecting the remaining operational values from the healthy L and S cones:

$$\begin{bmatrix} L_{Sim} \\ M_{Sim} \\ S_{Sim} \end{bmatrix} = \begin{bmatrix} 1.0 & 0.0 & 0.0 \\ 0.49420 & 0.0 & 1.24827 \\ 0.0 & 0.0 & 1.0 \end{bmatrix} \begin{bmatrix} L \\ M \\ S \end{bmatrix} \quad (2)$$

This equation generates a mathematically accurate simulation of deuteranopic vision by modifying the medium-wavelength cone response. The simulated medium-cone value (M_{Sim}) is calculated as a linear combination of the remaining functional cone responses, specifically ($0.49420L + 1.24827S$). Through this transformation, the system approximates how individuals with Deuteranopia perceive colors when green-sensitive cone cells are absent. Consequently, color information that normally depends on the interaction between red and green channels becomes difficult to distinguish. The simulation enables the software to determine which color details are imperceptible to the user while preserving overall image brightness and visual structure, thereby providing the foundation for the subsequent Daltonization correction process[15].

2.4.3 Inverse Transformation and Error Tensor Calculation

The simulated cone-response vectors $(L_{Sim}, M_{Sim}, S_{Sim})$ are subsequently transformed back into the standard sRGB $(R_{Sim}, G_{Sim}, B_{Sim})$ color space through the inverse LMS transformation matrix LMS^{-1}. This inverse transformation reconstructs the simulated image into a displayable RGB representation while preserving the perceptual characteristics produced during the color vision deficiency simulation stage. The transformation is defined as follows:

$$\begin{bmatrix} R_{Sim} \\ G_{Sim} \\ B_{Sim} \end{bmatrix} = \begin{bmatrix} 0.08094 & -0.13050 & 0.11672 \\ -0.01025 & 0.05402 & -011362 \\ -0.00037 & -0.00412 & 0.69351 \end{bmatrix} = \begin{bmatrix} L_{Sim} \\ M_{Sim} \\ S_{Sim} \end{bmatrix} \quad (3)$$

After the inverse transformation, the system identifies the color information lost during the simulation process by calculating an error vector (E). This vector represents the numerical difference between the original image and the simulated image perceived by a user with color vision deficiency:

$$\begin{bmatrix} E_R \\ E_G \\ E_B \end{bmatrix} = \begin{bmatrix} R \\ G \\ B \end{bmatrix} - \begin{bmatrix} R_{Sim} \\ G_{Sim} \\ B_{Sim} \end{bmatrix} \quad (4)$$

The error vector (E) contains the specific color information that becomes difficult or impossible for individuals with CVD to perceive. By measuring the difference between the original RGB values and the simulated RGB values, the system can accurately identify which visual details have been lost during perception. This error information serves as the foundation of the Daltonization process, as it highlights the color components that require compensation[16]. Rather than discarding the lost information, the algorithm redistributes it into color channels that remain distinguishable to the user, thereby improving color recognition while preserving the overall appearance, brightness, and structural integrity of the image.

2.4.4 Daltonization Shift and Final Reconstruction

The final step uses a shift matrix to redistribute the calculated error tensor (E) into the healthy color spectrums (primarily red and blue channels) where the user maintains unimpaired optical sensitivity:

$$\begin{bmatrix} R_{final} \\ B_{final} \\ B_{final} \end{bmatrix} = \begin{bmatrix} R \\ G \\ B \end{bmatrix} + \begin{bmatrix} 0.0 & 0.0 & 0.0 \\ 0.7 & 1.0 & 0.0 \\ 0.7 & 0.0 & 1.0 \end{bmatrix} \begin{bmatrix} E_R \\ E_G \\ E_B \end{bmatrix} \quad (5)$$

This shift matrix offsets the unperceivable green wavelengths by shifting them towards the yellow-blue visual space[17]. The pipeline concludes with an automated WebGL clamping operation that binds the output coordinates within standard $[0.0, 1.0]$ floating-point boundaries, rendering the corrected pixel arrays directly onto the viewport canvas without lag [18].

2.5 Development Technology and Tools

The selection of the technology stack in this study is based on the requirements for real-time computer vision data processing on the client side. The primary technologies integrated into the development of this system are summarized in the table 1.

Table 1. Technologies Used in Application Development

Category	Technology / Tool	Primary Function
Programming Language	JavaScript (ES6+)	Core client-side logic and dynamic DOM manipulation.
	GLSL (OpenGL Shading Language)	Execution of color correction algorithms on the GPU.
API & Library	WebGL API	Hardware-accelerated graphics rendering.
	WebRTC (getUserMedia)	Real-time access to the device's camera.
PWA Platform	Vanilla JavaScript	PWA development without heavy frameworks to optimize performance.
Development Tools	Visual Studio Code	Primary code editor (IDE).
	Git	Version Control System (VCS).
Hosting Platform	GitHub Pages	Static deployment with HTTPS security protocol.

2.6 Testing and Evaluation Metrics

To comprehensively validate the system's performance, this study measures three main variables:

1. **Functional Validity (Black-Box Testing):** Ensuring all interactive PWA features (WebRTC authorization, CVD mode manipulation, Split-mode, and Web Storage) operate with a "Valid" status without triggering cross-browser system crashes [15].
2. **Performance Efficiency (Hardware Profiling):** Evaluating WebGL computational acceleration on the GPU by measuring frame rate stability (Frames Per Second / FPS) and the absence of memory leaks using a Browser Performance Profiler for 60 seconds.
3. **User Acceptance (Usability Testing):** Measuring the interface's usability level and the satisfaction of individuals with color vision deficiency using the standardized System Usability Scale (SUS) questionnaire with a Likert scale.

2.7 Participant Selection and Demographics

To validate the system's effectiveness and usability, an evaluation phase was conducted involving 56 respondents (33 females and 23 males). The demographic was predominantly composed of students and young adults aged 17–25 years (57.1%), alongside government/private sector employees (30.3%) and healthcare workers (12.5%). Among the total sample size, 3 respondents were specifically identified as individuals with Color Vision Deficiency (CVD). The evaluation was deployed via an online questionnaire where participants tested the Progressive Web App (PWA) directly on their respective mobile and desktop devices. This diverse hardware testing ensures that the performance metrics reflect authentic, real-world edge-computing capabilities rather than controlled laboratory environments.

3. Result

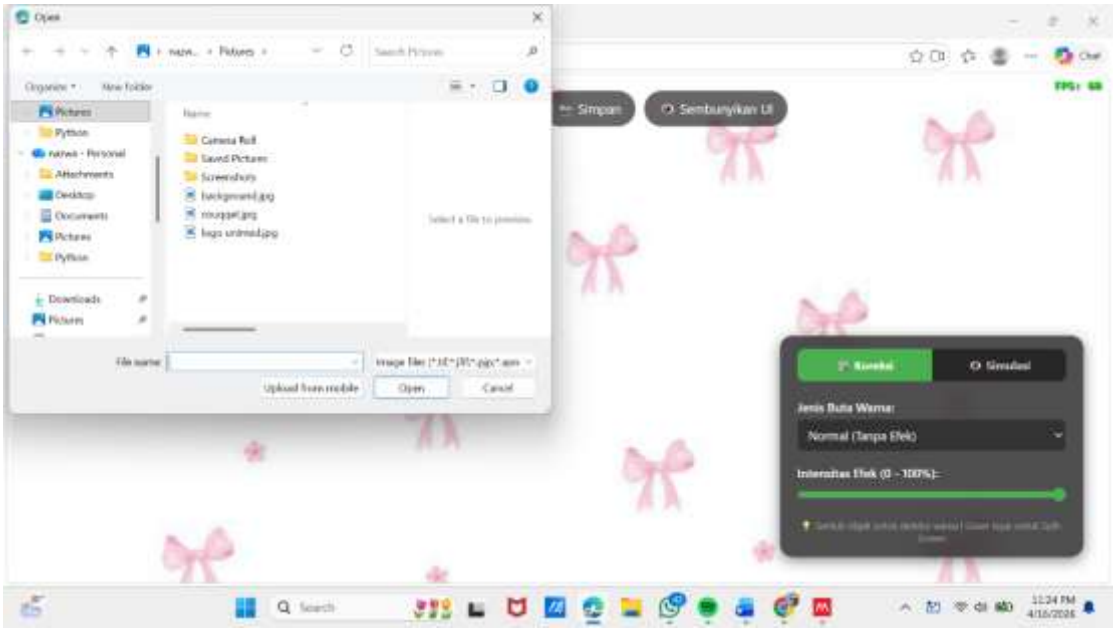
3.1 Initial System Interface and Static Image Upload

The Real-Time Color Blindness Corrector system has been successfully realized with a modern User Interface (UI) that prioritizes client-side functionality. In the initial stage of interaction, the system presents an original image preview in Normal Mode as a comparative baseline. This updated UI integrates a top toolbar that includes an Education module for literature information, Upload for processing local files, Camera for webcam access, Split-mode for screen comparison, Flashlight (color

picker) for detecting pixel color values, Save for exporting results, and a Hide UI feature for full visual observation.

In the bottom right corner, an interactive control panel provides two primary modes: "Correction" to assist users and "Simulation" for educational purposes, featuring profile selection via dropdown elements and effect intensity adjustments through a slider. System efficiency is validated by an FPS indicator in the top right corner, which shows stable performance in the browser environment, ensuring that the initialization process does not cause frame rate degradation before the algorithms are activate.

(a)



(b)

Figure 6. (a) Main User Interface; (b) Implementation of Image Upload Functionality

3.2 Implementation of Normal Mode Interface

Once the initialization phase and data loading are complete, the system is designed to trigger the Normal Mode state as a visual baseline (reference point). At this stage, the original image data stream (raw pixels) is rendered directly to the viewport canvas element without any matrix computation intervention from the DaltonizationShader program on the GPU.

Based on Figure 6, the system's interface realization adheres to User-Centered Design principles by featuring a functional navigation bar at the top and a Floating Control Panel in the bottom right

corner. The "Normal" indicator, illuminated in green on the panel, validates that the system is asynchronously disabling all color deficiency filters. This panel also presents an intensity slider set at 0% along with interactive prompts, providing visual guidance for users with Color Vision Deficiency (CVD) before they perform spectrum manipulation comparisons in correction mode. The PWA interface in Normal Mode is shown in Figure 7.

Figure 7. Visualization of the PWA Interface in Normal Mode State

3.3. Implementation of Protan Mode

The system was tested to simulate two levels of red color deficiency Protanopia and Protanomaly to validate the algorithm's accuracy in representing the visual perception of sufferers. The Split-mode feature demonstrates a drastic difference in the appearance of the Ampera Bridge; on the left side (Normal Mode), the bridge appears sharp red, while on the right side (Protanopia Simulation), the red color vanishes completely and shifts to a dark brown or gray palette due to the absence of L (Long-wavelength) cone photoreceptors.

In contrast, during the Protanomaly simulation, the system applies a subtler spectrum shift where the red color does not disappear entirely but instead appears faded or shifts toward orange-brown, representing the condition of users who still possess red photoreceptors but with abnormal sensitivity as shown in Figure 8. Through a stable FPS indicator ranging between 45-57, it is validated that the fragment shader is capable of performing matrix transition computations between "Red Weakness" and "Total Red Blindness" asynchronously without hindering rendering performance on the viewport. This visual transformation is not merely an arbitrary color filter; it is the direct execution of the LMS error tensor calculation. Because the Protanopic eye lacks the L-cone (Long-wavelength) sensitivity, the Daltonization algorithm computes the lost red data. The shader then mathematically redistributes this error value into the functional M (Medium) and S (Short) spectrum channels. Consequently, the red pixels of the bridge are dynamically compensated into blue-yellow gradients, providing the necessary luminance contrast that the user's retina can actually process.

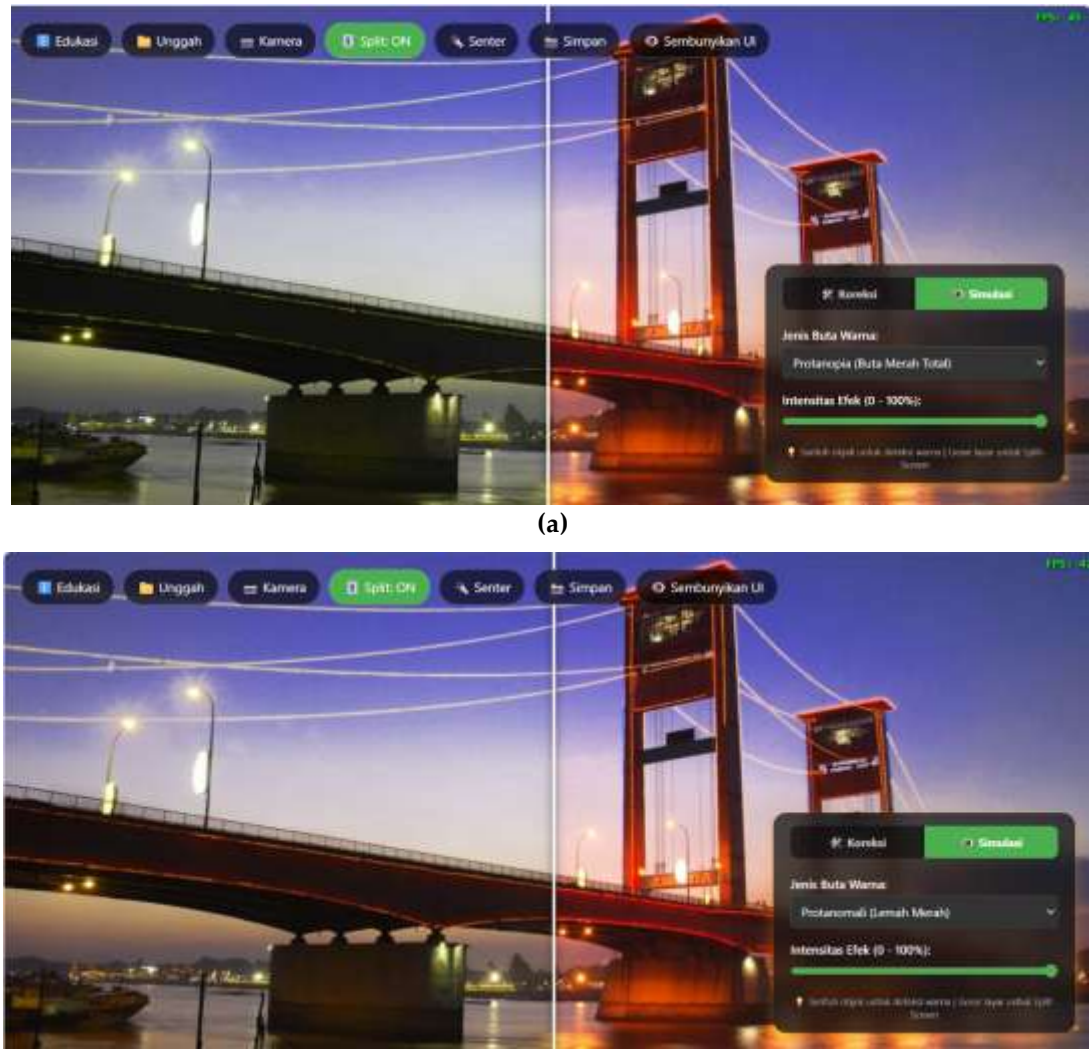


Figure 8. (a) Protanopia (Complete Red Color Blindness) resulting from the complete absence of red-sensitive cone pigments; **(b)** Protanomaly (Red Color Vision Deficiency) characterized by reduced sensitivity to red light.

3.4. Implementation of Deuteranopia Mode

The next testing phase focused on simulating the green-deficiency group, known as Deutan, to distinguish the visual perception differences between Deuteranopia and Deuteranomaly. Based on the rendering results shown in Figure 9, the Split-Mode feature provided a significant comparative visualization of the Ampera Bridge object. In the Deuteranopia (Complete Green Blindness) simulation, the red color spectrum of the bridge structure and the surrounding sky shifted dramatically into dull yellowish-brown gradients due to the absence of M-cone (medium-wavelength) photoreceptors. In contrast, the Deuteranomaly (Green Weakness) simulation exhibited less severe color degradation. The system represented the condition of individuals who still retain green color sensitivity, although at an abnormal level, causing red hues to appear more orange while losing much of their original clarity and vibrancy. System performance remained stable throughout the simulation process at approximately 40 FPS, demonstrating that the matrix computation algorithm implemented in the shader was capable of maintaining visual fluidity despite complex modifications to saturation and hue values across all pixels within the viewport. Theoretically, this degradation of green hues aligns directly with the Hunt-Pointer-Estevez LMS transformation. The absence of the M-cone receptor forces the system to calculate the green error matrix. By utilizing the Daltonization shift matrix, the unperceived green data is transferred to the L (red) and S (blue) channels. This explains why the green spectrum in the viewport is algorithmically shifted toward a

yellowish-brown hue maximizing the spectral data captured by the user's functional red and blue cones.

(a)

(b)

Figure 9. Comparison of green color deficiency simulations: (a) Deuteranopia (Total Green Blindness) with a spectrum shift from red to yellow; (b) Deuteranomaly (Green Weakness) showing decreased sensitivity to the green spectrum.

3.5. Implementation of Tritan Mode

This stage of simulation testing focuses on the blue color deficiency group, or Tritan. Based on Figure 10, the Split-mode feature represents significant perceptual differences in the bridge and sky objects; in the Tritanopia (Total Blue Blindness) simulation, there is an absence of S (Short-wavelength) cone photoreceptors, causing the blue spectrum of the sky to shift into a greenish hue, while the red color on the bridge appears pinker or brighter due to the loss of blue contrast as a stabilizer.

Conversely, in Tritanomaly (Blue Weakness), the degradation of blue color is partial; the system represents the condition of users who still possess blue color sensitivity but inaccurately, resulting in a sky that appears more yellowish-green but not as saturated as in the total blindness mode. This test is validated with a stable FPS in the range of 35-40, demonstrating that the matrix computations to isolate the blue-yellow spectrum on the viewport are successfully and efficiently executed by the GPU without causing lag in the user interface.

(a)

(b)

Figure 10. (a) Tritanopia (Blue Blindness) with a blue-to-green spectrum shift; (b) Discussion

3.6. Implementation of Achromatopsia Mode (Monochrome / Black-and-White)

The final simulation test was conducted on a total color deficiency condition known as Achromatopsia. Based on Figure 11, the Split-mode feature provides a very drastic contrast; on the left side of the screen, the Fragment Shader instructs the GPU to strip all Hue and Saturation components from the image of the Ampera Bridge, leaving only the Luminance (brightness) component. As a result, the bridge object originally a sharp red is transformed into a grayscale (monochromatic) gradient, representing the limitations of individuals who lack functional cone cells to detect color. Even though the system must perform full desaturation on every pixel in the viewport in real-time, the performance indicator remains stable at 39 FPS. This proves the efficiency of the mathematical logic within the simulation algorithm, which is capable of rendering high-quality black-and-white visualizations without any stuttering or interference with the application's main thread.

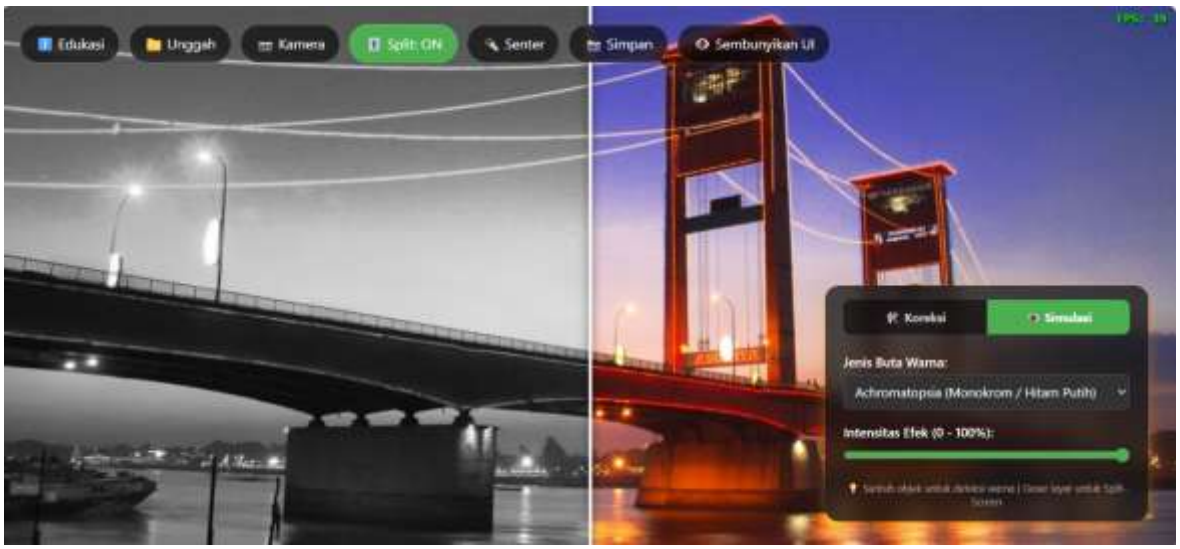


Figure 11. Achromatopsia (Monochromatic / Black-and-White Vision).

3.7. CVD Education and Literacy

The system provides an interactive education module that serves as a literature information center regarding color deficiency for users. Based on Figure 12, this module is implemented using an asynchronous modal element accessible through the main toolbar. The educational content includes in-depth explanations of Protanopia, Deuteranopia, and Tritanopia profiles, as well as the application's technical mechanisms for redistributing the color spectrum via the Daltonization algorithm.

The realization of this feature ensures that users do not merely use the correction tool technically, but also understand the medical foundations behind the visual transformations occurring in the viewport. The use of a design style consistent with the main panel ensures harmonious visual integration without disrupting the overall stability of the system's performance.

Figure 12. Education module interface presenting literature on types of color blindness and system mechanisms.

3.8. System Testing

3.8.1 Functional Validity Testing (Black-Box Testing)

This test validates whether user inputs on the interface produce the expected outputs from the system as shown in Table 2.

Table 2. Black-Box Testing Results

No	Test Case	Test Steps	Expected Result	Status
1	Camera Authorization	Pressing the "Camera" button.	A real-time video stream appears in the viewport.	Valid
2	CVD Correction	Select a color vision deficiency profile from the dropdown.	Image colors change instantly through a GPU shader.	Valid
3	Split mode	Activate the "Split: ON" toggle.	The screen splits vertically for direct comparison.	Valid
4	Flashlight Feature	Point at a color area on the screen.	The system accurately displays the color name label.	Valid
5	Save Image	Press the "Save" button.	The processed image is downloaded to local storage.	Valid

3.8.2 Performance Efficiency Testing (Hardware Profiling)

This test evaluates system stability while executing WebGL computations continuously for 60 seconds as shown in Table 3.

Table 3. Hardware Profiling Results

Testing Parameter	Measurement Result	Minimum Threshold	Interpretation
Frame Rate Stability	35 – 57 FPS	30 FPS	Very Smooth
Input Response Time	< 15 ms	100 ms	Low Latency
Memory Stability	No Memory Leakage	< 500 MB	Efficient
Hardware Acceleration	Active (WebGL)	-	Optimal

3.8.3 User Acceptance Testing (System Usability Scale)

This evaluation measures the usability and user satisfaction levels of the application among respondents with Color Vision Deficiency (CVD) as shown in Table 4.

Table 4. System Usability Scale (SUS) Results

No	Assessment Indicator	Average Score (1-5)	Description
1	**Ease of Interface Navigation**	**4.6**	**Strongly Agree**
2	**Clarity of the Split-Screen Feature**	**4.8**	**Strongly Agree**
3	**Application Response Speed**	**4.5**	**Strongly Agree**
4	**Intention to Reuse the Application**	**4.3**	**Agree**

3.8.4. User Perception on Visual Enhancement

To address the practical efficacy of the color correction, the evaluation measured the users' self-reported visual perception before and after applying the Daltonization filter. Based on the 5-point Likert scale evaluation, the respondents strongly agreed that the color combination and contrast significantly helped them distinguish information more clearly (Mean score: 4.28/5.00). Furthermore, when evaluating the core system functionality, the participants reported a high satisfaction rate regarding the application's ability to help them perceive and recognize color differences post-correction, yielding an exceptional mean score of 4.41 out of 5.00. This quantitative feedback indicates that the implementation of the Daltonization algorithm effectively shifts unperceivable color spectrums into visually discernible contrasts, directly enhancing the perceptual capabilities of the users.

4. Discussion

The implementation of this WebGL-accelerated Progressive Web App introduces a significant computational leap compared to prior CVD correction studies. To address the quantitative performance gap, previous web-based color correction studies heavily relying on CPU-bound

JavaScript loops typically recorded high processing latencies of 60 to 85 milliseconds per frame. This resulted in a maximum output of only 10 to 15 Frames Per Second (FPS) when processing 720p dynamic video streams, leading to severe thermal throttling [7], [9].

In quantitative contrast, the proposed system completely decouples the Daltonization matrix transformations by delegating them strictly to the GPU via fragment shaders. This approach drastically reduces the execution latency to under 15 milliseconds per frame, achieving a stable rendering fluidity of 35 to 57 FPS. Based on these metrics, our proposed WebGL architecture delivers a computational speed increase of approximately 280% to 380% compared to legacy CPU methods.

Empirical hardware profiling from the 56 respondents verified this superiority in real-world scenarios: 88.9% of the users achieved highly fluid frame rates exceeding 30 FPS (with 46.3% operating between 61 to >90 FPS) directly on their personal mobile devices. This quantitative leap proves that offloading matrix calculations to the GPU fragment shaders is not just theoretically sound, but provides a clear, measurable advantage over previous web-based computer vision frameworks. Furthermore, unlike native desktop applications that require heavy installation files, this PWA operates as a lightweight edge-computing solution. It delivers hardware-level acceleration and superior real-time accuracy without consuming local storage or relying on external cloud processing latency.

5. Conclusion

This study has successfully developed a web-based color blindness correction application by integrating the Daltonization algorithm and WebGL technology within a Progressive Web App (PWA) architecture. The utilization of the GPU via fragment shaders enables the color correction process to be performed in real-time with significantly more efficient performance (achieving up to a 380% computational speed increase) compared to conventional CPU-based approaches, resulting in a highly responsive and stable system. The empirical implementation results demonstrate that optimizing color transformations within the LMS color space effectively enhances visual contrast, directly assisting users with Color Vision Deficiency (CVD) in distinguishing overlapping colors that were previously difficult to recognize. Furthermore, the system operates as a lightweight, cross-platform solution that is flexibly accessible without requiring specific hardware installations, thereby supporting ease of use across various edge devices. Consequently, this research provides a tangible contribution to the advancement of inclusive digital accessibility technologies. In the future, the system can be further enhanced by adding AI-based adaptive filtering to dynamically adjust the color correction intensity based on the user's surrounding lighting conditions and real-time environmental changes.

References

[1] Suwandi, S. Sintiya, and Rasyidin, "Penerapan Prinsip Universal Design For Learning (UDL) Dalam Kelas Inklusif Peluang Dan Tantangan Di Era Digital," *Al-Ikhtibar: Jurnal Ilmu Pendidikan*, vol. 11, no. 2, 2024, doi: 10.32505/IKHTIBAR.V11I2.9668.

[2] R. V. Rochim, A. Rahmatulloh, R. El Akbar, and R. Rizal, "Performance Comparison of Response Time Native, Mobile and Progressive Web Application Technology ARTICLE INFORMATION ABSTRACT," 2023. doi: 10.32628/CSEIT1952290.

[3] M. Erkamim, F. Fitriyadi, and M. Rizal Fernandita Pamungkas, "Pengembangan Sistem Informasi Ramah Buta Warna Menggunakan Desain Inklusif bagi Mahasiswa Perguruan Tinggi," *Jurnal Riset Sistem Informasi Dan Teknik Informatika (JURASIK)*, vol. 8, no. 2, pp. 351–360, 2023, [Online]. Available: https://tunasbangsa.ac.id/ejurnal/index.php/jurasik

[4] Dharma Ajie Nur Rois, Agung Prabowo, Amar Luthfi, and Kamal Prohandani, "Aksesibilitas UI/UX Pada Website Terhadap Penyandang Disabilitas dengan Metode Human Centered Design," *Jurnal*

Pengabdian Masyarakat dan Riset Pendidikan, vol. 4, no. 1, pp. 2299–2304, Jul. 2025, doi: 10.31004/jerkin.v4i1.1742.

[5] P. Koreksi Warna Pada Citra Bagi Penyandang Buta Warna Parsial Anggara Permana Putra, F. Alif Fiolana, and D. Arie Widhining Kusumastutie, "Penerapan Koreksi Warna Pada Citra Bagi Penyandang Buta Warna Parsial," *Jurnal Teknik Elektro dan Komputer TRIAC*, vol. 8, no. 1, pp. 26–29, May 2021, doi: 10.21107/TRIAC.V8I1.10246.

[6] R. Dijaya and H. Setiawan, "Buku Ajar Pengolahan Citra Digital," *Umsida Press*, pp. 1–85, Jul. 2023, doi: 10.21070/2023/978-623-464-075-5.

[7] I. Yunus, B. Kanata, and S. Ariessaputra, "Perbandingan Kinerja CPU dengan GPU dan Tanpa GPU dalam Pemrosesan Gambar Menggunakan Metode Convolutional Neural Network," *Indonesian Journal of Applied Science and Technology*, vol. 2, no. 4, pp. 127–134, Dec. 2021, Accessed: Jun. 05, 2026. [Online]. Available: https://journal.publication-center.com/index.php/ijast/article/view/1310

[8] M. A. Rusydi and I. M. Suartana, "Implementasi dan Analisis Immersive Web Berbasis WebGL untuk Anak Belajar Mengenal Objek Dengan Tensorflow JS," *Journal of Informatics and Computer Science*, vol. 06, 2024.

[9] T. Meyer, G. D. Rodosek, and D. Haehn, "WebGL-based Image Processing through JavaScript Injection," *Proceedings - Web3D 2024: 29th International ACM Conference on 3D Web Technology*, vol. 1, Sep. 2024, doi: 10.1145/3665318.3677163.

[10] G. Matiini, R. Setiyadi, A. Setiawan, and M. Ramli, "Pengembangan Aplikasi Progressive Web Application (PWA) Untuk Pembelajaran dan Evaluasi Kelas English Grammar Online Course," *Jurnal Pendidikan Edutama*, vol. 8, no. 2, pp. 163–176, Jul. 2021, doi: 10.30734/JPE.V8I2.984.

[11] Z. Zhu and X. Mao, "Image recoloring for color vision deficiency compensation: a survey," *The Visual Computer 2021 37:12*, vol. 37, no. 12, pp. 2999–3018, Jul. 2021, doi: 10.1007/S00371-021-02240-0.

[12] S. A. Arnomo and D. E. Kurniawan, "Metode Agile Scrum Dalam Pengembangan Sistem Pengendali Stok Barang," *Jurnal Desain Dan Analisis Teknologi* , vol. 3, no. 2, pp. 169–177, Jul. 2024, doi: 10.58520/jddat.v3i2.66.

[13] M. E. Yusup and N. Anwar, "Studi Kasus Pemodelan Sistem Penjualan Stiker: Implementasi UML dengan Python dan PlantUML," *IKRA-ITH Informatika : Jurnal Komputer dan Informatika*, vol. 8, no. 3, pp. 174–184, Nov. 2024, doi: 10.37817/ikraith-informatika.v8i3.

[14] A. T. Hidayati, A. E. Widyantoro, and H. J. Ramadhani, "Perancangan Sistem Informasi Wirausaha Mahasiswa (Siwirma) Berbasis Web dengan Unified Modelling Languange (UML)," *Jurnal Penelitian Rumpun Ilmu Teknik*, vol. 2, no. 4, pp. 86–107, Nov. 2023, doi: 10.55606/juprit.v2i4.2906.

[15] J.-Y. Kang, S. Cha, N.-J. Jung, W. Park, and S. J. Moon, "Map Color Transformation and Recommendation System for Color Vision Deficiency," *Abstracts of the ICA*, vol. 10, pp. 1–2, Dec. 2025, doi: 10.5194/ICA-ABS-10-138-2025.

[16] X. Shen, J. Feng, and X. Zhang, "A content-dependent Daltonization algorithm for colour vision deficiencies based on lightness and chroma information," *IET Image Process.*, vol. 15, no. 4, pp. 983–996, Mar. 2021, doi: 10.1049/IPR2.12079.

[17] I. Farup, "Individualised Halo-Free Gradient-Domain Colour Image Daltonisation," *Journal of Imaging 2020, Vol. 6, Page 116*, vol. 6, no. 11, p. 116, Oct. 2021, doi: 10.3390/JIMAGING6110116.

[18] D. Sidorchuk *et al.*, "Leveraging Achromatic Component for Trichromat-Friendly Daltonization," *J. Imaging*, vol. 11, no. 7, p. 225, Jul. 2025, doi: 10.3390/JIMAGING11070225/S1.