International Journal of Scientific Research & Engineering Trends
Volume 12, Issue 1, Jan-Feb-2026, ISSN (Online): 2395-566X

Computer Vision: Real Time Object Detection Using AI and Machine Learning Real Time Eye Strain Detection System

Snehal Pravin Pangavhane[1], Mayur Navnath Dhumal[2], Gayatri virendra patil[3], Harish Ravindra Badgujar[4], Mrs. Samiksha Gawali[5]
[1,2,3,4,5]Department of Computer Science and Engineering
Sandip University Nashik, Maharashtra, India

Abstract- The project entitled "Computer Vision: Realtime object detection using AI and Machine Learning Realtime Eye Strain Detection System", focuses on enhancing digital well-being by addressing the growing problem of Computer Vision Syndrome (CVS), commonly known as Digital Eye Strain. With the increasing dependency on digital devices for work, study, and entertainment, users often experience symptoms such as eye dryness, irritation, blurred vision, headaches, and reduced concentration. The proposed system utilizes Artificial Intelligence (AI) and Computer Vision (CV) technologies to monitor and analyze real-time indicators of visual fatigue. Using tools such as MediaPipe and OpenCV, it detects parameters like blink rate, eye aspect ratio (EAR), sitting distance, and ambient lighting. A user-friendly PyQt6 graphical interface enables seamless interaction, providing users with real-time alerts, adaptive feedback, and personalized wellness recommendations. By integrating AI APIs like Gemini or Grok, the system generates intelligent insights, preventive suggestions, and health trend reports. This promotes healthy screen habits and reduces the risk of long-term eye strain. The Vision Shield system contributes to digital wellness, productivity improvement, and AI-based health monitoring, offering a scalable solution for students, professionals, and organizations alike.

Keywords: Eye Strain, Artificial Intelligence, Computer Vision, MediaPipe, OpenCV, PyQt6, Digital Wellness, Computer Vision Syndrome (CVS).

I. INTRODUCTION

Overview

In today's digital era, almost every aspect of human life—education, work, and entertainment—depends heavily on digital screens. This excessive screen exposure has led to a rise in Computer Vision Syndrome (CVS), also known as Digital Eye Strain, which includes symptoms such as eye dryness, irritation, blurred vision, headaches, and reduced productivity. Prolonged use of computers, mobile devices, and tablets without adequate rest can cause significant visual discomfort and long-term vision issues.

The "Vision Shield – AI-Based Eye Strain Detection System" is an innovative solution designed to address this modern health challenge. The system leverages Artificial Intelligence (AI) and Computer Vision (CV) technologies to automatically monitor and analyze key indicators of eye strain in real time. It tracks parameters such as blink rate, eye aspect ratio (EAR), sitting distance, and ambient lighting conditions using MediaPipe and OpenCV.

The system provides real-time alerts and intelligent recommendations through a PyQt6-based graphical interface, ensuring smooth user interaction and visual feedback. It also integrates with advanced AI APIs such as Gemini or Grok to generate personalized insights and preventive suggestions, contributing to healthier digital habits and improved visual wellness.

Brief Description

With the rapid adoption of digital technology in professional and personal environments, individuals spend long hours interacting with display screens. Studies show that a majority of computer users experience symptoms of eye fatigue or visual discomfort after extended screen use. Traditional solutions, like fixed-timer reminders or manual breaks, fail to consider personalized user behavior or environmental factors.

The need for a real-time, intelligent monitoring system that adapts to each user's behavior has become essential. The Vision Shield project aims to fulfill this gap by using AI-powered algorithms to monitor, detect, and alert users about eye strain conditions, thus helping them maintain optimal digital wellness.

Problem Definition
The main problem addressed by this project is visual fatigue caused by prolonged screen exposure. The system must identify and respond to early indicators of eye strain, providing timely feedback and recommendations to prevent health deterioration.

Key challenges include:
- Continuous monitoring of eye activity and posture using standard webcams.
- Accurate detection of blink rates and eye aspect ratio using computer vision models.
- Analyzing environmental lighting conditions.
- Generating real-time alerts without disturbing user workflow.

Objectives of the Project
The primary objective of Vision Shield is to design and implement an AI-driven real- time monitoring system that detects and mitigates eye strain effects due to prolonged screen usage.

Additional objectives include:
- To design a computer vision module that detects blink rate, eye aspect ratio (EAR), and sitting distance using MediaPipe and OpenCV.
- To monitor ambient light and suggest optimal lighting conditions for screen usage.
- To create a user-friendly interface using PyQt6 for easy visualization of results and statistics.
- To integrate AI APIs for personalized recommendations and wellness insights.
- To generate real-time alerts and adaptive feedback to prevent visual fatigue.

Scope of the Project
The project focuses on software-based detection and prevention of digital eye strain. It is applicable to students, professionals, educators, and organizations who spend extended time on digital devices.

The scope includes:
- Real-time detection of eye strain parameters using standard webcams.
- Data visualization and alert system through the PyQt6 GUI.
- Integration of AI-based recommendations and insights.
- Secure storage of usage data and analysis reports.

II. LITERATURE SURVEY

Existing System
In the present digital environment, people spend prolonged hours in front of screens for professional, educational, and personal activities. This leads to Computer Vision Syndrome (CVS), also known as Digital Eye Strain, which causes symptoms like dryness, irritation, blurred vision, headaches, and fatigue.

Existing systems designed to address these problems are mostly screen-time tracking applications or basic reminder tools. For example, features such as Digital Wellbeing on Android or Screen Time on iOS record the total duration of device usage and display daily or weekly summaries. Similarly, software-based timers or browser extensions send notifications to take short breaks.

However, these traditional systems are time-based rather than behavior-based. They cannot analyze physiological indicators such as blink rate, eye aspect ratio (EAR), or ambient light level, which are direct causes of visual fatigue. The alerts generated are generalized and do not adapt to user habits or environmental conditions.

Another limitation of existing solutions is the lack of intelligent feedback. They do not provide personalized recommendations or data-driven insights to improve long-term eye health. These systems also lack real-time monitoring and rely heavily on user compliance, reducing their effectiveness in preventing eye strain.

Thus, while the existing systems raise awareness about screen usage, they fail to deliver proactive, intelligent, and adaptive monitoring necessary for modern digital wellness.

Proposed System
The proposed project, "Vision Shield – AI-Based Eye Strain Detection System," aims to overcome the limitations of current systems by integrating Artificial Intelligence (AI) and Computer Vision (CV) techniques to monitor visual fatigue parameters in real time.

This system uses MediaPipe and OpenCV to detect and analyze important indicators like blink rate, eye aspect ratio (EAR), sitting distance, and ambient lighting conditions. These parameters are processed by intelligent algorithms that can identify early signs of eye strain and provide real-time alerts to the user.

The Graphical User Interface (GUI), developed using PyQt6, displays visual insights such as usage statistics, strain levels, and activity trends in an interactive and user- friendly manner. The system also incorporates AI APIs like Gemini or Grok to generate personalized recommendations and adaptive feedback based on user behavior and environmental conditions.

In addition to real-time alerts, the proposed system maintains usage history and analytical reports, allowing users to understand their long-term screen habits and wellness

improvements. The approach is proactive, focusing not only on detecting strain but also on preventing it through continuous monitoring and intelligent suggestion mechanisms.
The project's innovation lies in its combination of computer vision-based detection, AI- driven personalization, and intuitive visualization, making it a scalable solution for students, professionals, and organizations seeking to promote digital wellness and reduce the impact of prolonged screen exposure.

III. SOFTWARE REQUIREMENTS SPECIFICATION

The Vision Shield AI-Based Eye Strain Detection System focuses on identifying and reducing the effects of Computer Vision Syndrome (CVS) through continuous and intelligent monitoring. The project requirements are divided into functional and non- functional aspects to ensure a reliable, efficient, and scalable system.

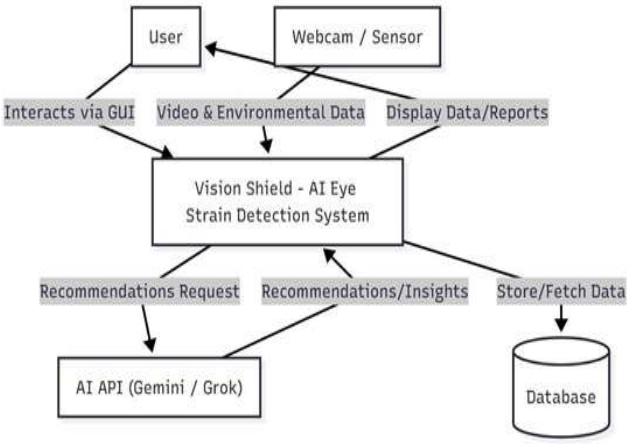
Purpose
The purpose of the Vision Shield system is to continuously monitor a user's eye activity using AI-based computer vision techniques and detect early signs of Computer Vision Syndrome (CVS). The system aims to provide real-time alerts, personalized recommendations, and health insights that help users reduce eye strain during prolonged digital screen usage. It also serves as an assistive tool for students, professionals, and organizations to promote digital eye well-being.

Project Scope
The scope of the project includes real-time face and eye detection, extraction of blink rate, eye closure levels, gaze direction, and screen exposure duration, and generation of eye-strain scores. The system will provide break reminders, guided micro-exercises, and weekly usage reports. It supports multi-device integration including laptops, desktops, and mobile phones. The project focuses on local processing for privacy, with optional cloud-based analytics. Administrative dashboards and exportable reports fall within the extended scope.

Design and Implementation Constrains
The system relies on camera availability with sufficient frame quality to detect micro-eye movements. Real-time inference requires optimized ML models compatible with mid- range hardware. Battery consumption limits the continuous monitoring duration on mobile devices. The design must follow privacy, consent, and data protection guidelines, especially for video-based biometric data. Cross-platform compatibility also imposes UI, API, and hardware constraints.

Assumptions and Dependencies
It is assumed that users will grant camera permissions and sit within the camera's field of view. The system depends on stable lighting conditions for accurate detection and on device processing power for smooth inference. Cloud modules depend on a stable internet connection. External libraries, computer vision frameworks, and OS-level APIs are assumed to remain supported throughout the lifecycle.

System Features (Use Case Diagrams) 3.2.1System Feature 1(Functional Requirement)
The system continuously captures real-time video frames, detects the user's eyes and face, calculates blink rate, eye aspect ratio, and gaze deviation, and generates an eye-strain score. If the score exceeds a threshold, the system provides immediate alerts and recommendations.



System Feature2 (Functional Requirement)
The system records usage duration, tracks daily and weekly screen exposure, and generates visual reports. It allows users to customize alert frequency, notification style, break intervals, and language preferences.

External Interface Requirements

User Interfaces
The interface includes a dashboard displaying real-time monitoring status, eye-strain levels, session timer, and alerts. The settings page provides configuration controls, and the reports page shows trends and summaries. The UI is simple, clean, and accessible to all users.

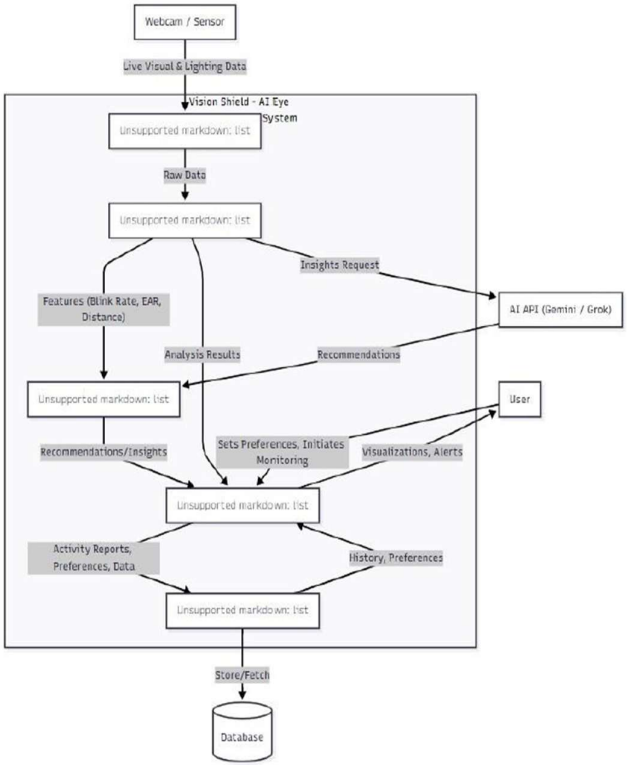
Hardware Interfaces
The system interacts with device cameras, ambient light sensors, and optional screens or smart displays. It requires minimum camera resolution for accurate detection.

Software Interfaces
The system depends on operating system APIs for camera access, notification services, and power management. It

interacts with ML frameworks such as OpenCV, TensorFlow Lite, or MediaPipe. Cloud services may include authentication and analytics APIs.

Communication Interfaces

For cloud-based features, the system uses secure HTTPS communication. It supports REST APIs for uploading anonymized data or retrieving reports. Offline mode is supported for local detection.



Nonfunctional Requirements

Performance Requirements

Real-time eye detection must be processed within 100–200 ms per frame to maintain smooth monitoring. The system should handle continuous operation for hours without performance degradation.

Safety Requirements

The system must avoid displaying alerts that distract the user at unsafe times, such as when driving. It must include disclaimers clarifying that it is not a medical diagnostic tool. The system should ensure safe data handling with retention limits.

Security Requirements

All collected data must be encrypted in transit and at rest. Sensitive data should preferably remain on-device unless the user opts into cloud storage. The system must prevent unauthorized access through authentication and secure logging.

Software Quality Attribute

The system must be reliable, scalable, maintainable, and portable. The architecture should support easy model updates, modular UI, and efficient resource usage. Usability and accessibility must be prioritized.

Other Requirements (If Applicable)

Database Requirements

For cloud deployment, the system stores anonymized user metrics, historical statistics, and admin analytics using a secure database. For local mode, only minimal configuration and temporary logs are saved.

Internalization Requirements

The system must support multilanguage interfaces and allow easy addition of new languages. Number formats, date formats, and reading directions must follow locale standards.

Legal Requirements

The system must comply with data protection laws such as GDPR, CCPA, and local IT regulations. Explicit consent must be taken before processing camera-based biometric data.

Reuse Objectives for the Project

The modular design should allow reuse of the AI detection engine, notification system, and reporting modules in other health-monitoring or productivity applications.

Analysis Model

Data Flow Diagrams

The main flow includes capture of video input, processing by the AI engine, generation of strain metrics, decision-making, and user feedback through alerts or reports.

Class Diagrams

Main classes include UserProfile, CameraManager, EyeDetectionEngine, FeatureExtractor, StrainAnalyzer, NotificationManager, SettingsManager, ReportGenerator, and AdminDashboard.

State-Transition Diagrams or Entity Relationship Diagrams

The user session states include Idle State, Monitoring State, Alert State, and Report State. The ER model includes tables such as User, Session, Metrics, Preferences, and Alerts.

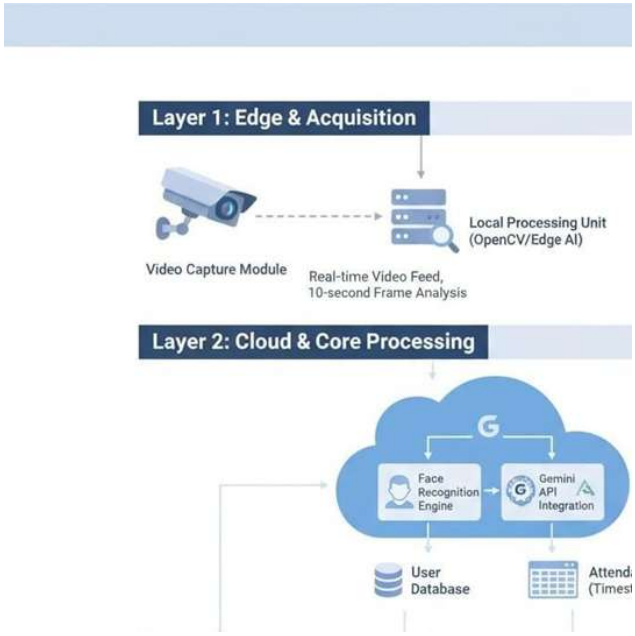
System Implementation Plan

The implementation begins with UI/UX design, dataset collection, and model training. The next phase integrates camera access, detection pipeline, and real-time inference. Following this, the alerting system, settings module, and reporting engine are developed. Cloud modules, admin panel, and security features are added in the later phase. Final steps

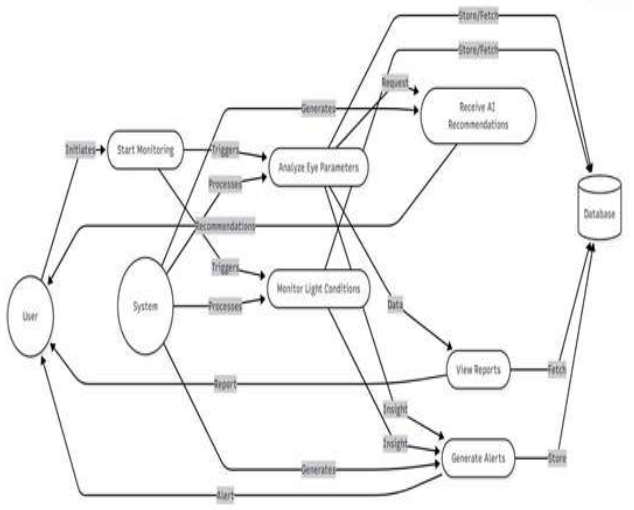
include testing, optimization, deployment, and user feedback-based refinement.

IV. SYSTEM DESIGN

System Architecture
The overall design follows a modular architecture, where each component performs a specific function. The data flows between these components in a sequential and logical manner, ensuring efficiency and maintainability.



UML Diagram
- Use Case Diagram

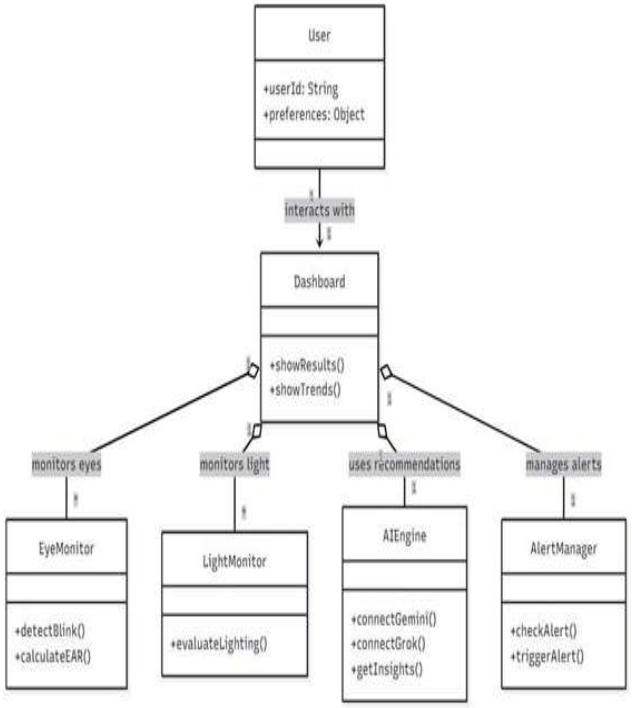


Actors:
- User
- System (AI Engine)
- Database

Use Cases:
- Start Monitoring
- Analyze Eye Parameters
- Monitor Light Conditions
- Generate Alerts
- View Reports
- Receive AI Recommendations

Explanation:
The user initiates monitoring; the system tracks real-time parameters and interacts with AI APIs to generate personalized insights. The results are displayed on the dashboard and stored for future analysis.

Class Diagram

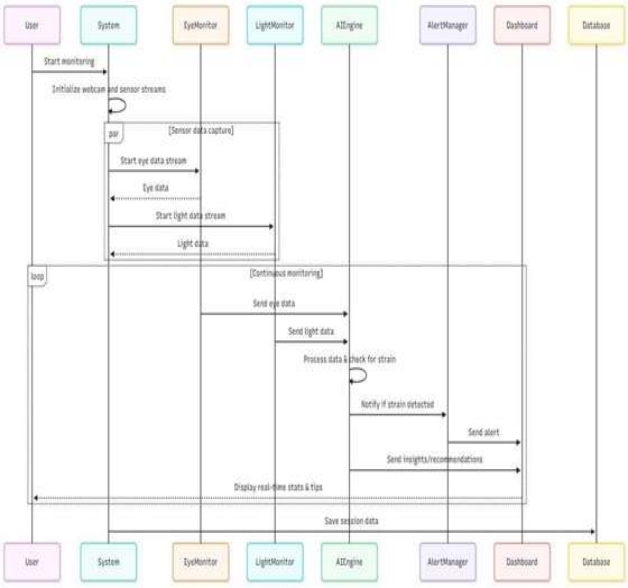


Main Classes:
- User – stores user details and preferences.
- EyeMonitor – handles blink detection and EAR calculations.
- LightMonitor – evaluates ambient lighting.
- AIEngine – connects with Gemini/Grok APIs for insights.
- AlertManager – manages real-time alert conditions.
- Dashboard – visualizes results and trends.

Relationships:

Each user instance interacts with one Dashboard, which depends on multiple monitoring classes (EyeMonitor, LightMonitor) and an AIEngine instance for recommendations.

Sequence Diagram



Flow Description:

1. User starts monitoring → System initializes webcam and sensor streams.
2. Eye and light data are continuously captured.
3. AI engine processes data and checks for strain indicators.
4. Alerts are generated and sent to the dashboard.
5. User views statistics and recommendations in real time.
6. System saves the session data in the database.

V. TECHNICAL SPECIFICATIONS

Technology details used in the project

The Vision Shield – AI-Based Eye Strain Detection System is developed using a modern and efficient technology stack to ensure real-time performance, accuracy, and cross- platform usability. The system runs on Windows 11 (Version 24H2) as the primary development and execution environment. Python 3.13.3 is used as the core programming language for implementing AI models, computer vision pipelines, data processing modules, and backend logic.

The frontend dashboard is designed using HTML5, CSS3, and JavaScript (ECMAScript 2025) to provide a clean, responsive, and interactive interface for users. Frameworks like OpenCV and MediaPipe are used for eye and face detection, blink analysis, and real-time image processing. PyQt6 is used for building rich desktop UI components when required.

For visualization of reports and usage statistics, the project integrates D3.js and Chart.js, enabling dynamic and interactive charts. The system also utilizes powerful AI APIs such as Gemini or Grok to generate intelligent insights, personalized recommendations, and behavioral pattern summarization for eye-strain prevention.

References to technology

The technologies used in this project are well-documented and widely supported in the AI and software development industry. Windows 11 (24H2) provides an optimized platform for high-performance AI computation and secure data handling. Python 3.13.3 supports advanced AI libraries, improved performance, and modern language features.

OpenCV and MediaPipe are referenced for computer vision algorithms, landmark detection, and tracking tasks essential for eye-strain monitoring. PyQt6 serves as a reference for desktop interface design, while HTML5, CSS3, and JavaScript (ES2025) are standard technologies referenced for modern web UI development.

D3.js and Chart.js are used as references for creating interactive and real-time visualizations. AI APIs like Gemini or Grok are referenced for integrating advanced AI- driven analytics and generating contextual, user-specific feedback.

VI. PROJECT ESTIMATE

Project Estimate

The expected estimate for the Vision Shield project includes time, effort, and cost distribution for major development phases. Expected estimate refers to the planned effort, while actual estimate is calculated after completion.

Expected Estimate

The expected estimate includes planning, UI design, dataset preparation, model development, feature integration, testing, documentation, and deployment. The team allocates fixed hours per module and estimates additional buffer time for unexpected delays.

Actual Estimate

Actual estimate includes the real time spent on development activities. Model training, optimization, and debugging generally take longer than expected because of data inconsistencies and performance issues. Team meetings, integration conflicts, and testing cycles also add additional hours beyond initial expectations.

Project Schedule (Sem I & Sem II) Semester I

The first semester focuses on requirement analysis, research, model selection, design, and basic prototype development. Activities include literature survey, finalization of scope, architecture design, and initial implementation of the eye

detection module. A mid-semester prototype consists of basic face detection and user interface layout.

Semester II
The second semester focuses on system completion, integration of all modules, cloud connectivity (if applicable), testing, performance tuning, result analysis, report documentation, and final deployment. The full working system includes real-time eye-strain alerts, reports, settings module, and the final optimized AI model.

Team Structure (Diagram) Team Members
The project team includes a project leader, AI developer, frontend developer, backend developer, UI/UX designer, tester, and documentation lead. Each member manages one or more responsibilities based on skill sets and workload distribution.

VII. SOFTWARE IMPLEMENTATION

Introduction
The implementation phase translates the system design into an operational software product. The Vision Shield AI-Based Eye Strain Detection System is developed using Python and AI-based computer vision libraries. Each module designed in earlier phases has been implemented as an independent, reusable component to ensure modularity and scalability.

database (Data Dictionary)

Table 1: User
Purpose: Stores basic user information and system preferences. Attributes:
User_ID — Primary key, unique identifier for each user (INTEGER). Full_Name — Name of the user (TEXT).
Email — Email address for login or recovery (TEXT). Age — User's age (INTEGER).
Uses_Glasses — Indicates if user uses glasses or contacts (BOOLEAN). Created_At — Date and time when user account was created (DATETIME).

Table 2: Preferences
Purpose: Stores individual user settings and notification customizations. Attributes:
Pref_ID — Primary key (INTEGER).
User_ID — Foreign key referencing User table (INTEGER). Alert_Sensitivity — Sensitivity level for strain detection (TEXT). Break_Interval — Preferred break interval in minutes (INTEGER). Language — Selected system language (TEXT). Theme_Mode — Light or Dark theme (TEXT).

Table 3: Session
Purpose: Stores information about each monitoring session. Attributes:
Session_ID — Primary key (INTEGER).
User_ID — Foreign key referencing User table (INTEGER). Start_Time — Session start timestamp (DATETIME).
End_Time — Session end timestamp (DATETIME). Total_Duration — Duration of session in minutes (INTEGER). Average_Strain_Score — Session-level computed strain score (FLOAT).

Table 4: Eye_Metrics
Purpose: Stores extracted AI metric values used to detect strain. Attributes:
Metric_ID — Primary key (INTEGER).
Session_ID — Foreign key referencing Session table (INTEGER). Blink_Rate — Blinks per minute (FLOAT).
Eye_Aspect_Ratio — Eye openness measurement (FLOAT). PERCLOS — Percentage of eye closure (FLOAT).
Gaze_Stability — Gaze focus deviation score (FLOAT). Ambient_Light — Light level around the user (INTEGER).
Timestamp — Exact moment when metrics were recorded (DATETIME).

Table 5: Alerts
Purpose: Records all strain-related warnings and notifications shown to users. Attributes:
Alert_ID — Primary key (INTEGER).
Session_ID — Foreign key referencing Session table (INTEGER).
Alert_Type — Type of alert (Low Blink Rate / Long Session / High Strain) (TEXT). Alert_Message — Prompt shown to the user (TEXT).
Alert_Time — Timestamp of the alert (DATETIME).

Table 6: Reports
Purpose: Stores summarized data for weekly or monthly reports. Attributes:
Report_ID — Primary key (INTEGER).
User_ID — Foreign key referencing User table (INTEGER). Report_Type — Weekly or Monthly (TEXT).
Average_Exposure — Average screen exposure time (FLOAT). Average_Strain — Average strain score (FLOAT).
Generated_At — Timestamp when report was created (DATETIME).

Important module, Mathematical model and algorithm

Each module of the system is described below according to the prescribed structure.

Module 1: Eye Detection Module
Purpose:
To detect and analyze the user's eye parameters, calculate the blink rate and EAR, and identify signs of eye strain.
Inputs:
Live video feed from webcam.
Outputs:
Blink count, EAR value, eye strain level.

Files Used:
Temporary image/frame data, user configuration file.
Algorithm:
1. Capture video stream.
2. Detect facial landmarks using MediaPipe.
3. Extract eye region coordinates.
4. Calculate EAR value per frame.
5. Determine blink rate and strain threshold.
6. Generate alert if thresholds are crossed.

User Interface:
Display of blink count and EAR graph on GUI dashboard.

Module 2: Ambient Light Monitoring Module

Purpose:
To measure the lighting conditions of the surrounding environment and recommend optimal brightness levels.
Inputs:
Sensor data or webcam image brightness.
Outputs:
Light intensity value, lighting suggestion message.
Algorithm:
1. Capture frame or sensor input.
2. Calculate brightness level.
3. Compare with optimal illumination thresholds.
4. Display suitable feedback on GUI.

User Interface:
Light indicator with suggestions such as "Increase room brightness" or "Reduce screen glare."

Module 3: AI Recommendation Module
Purpose:
To generate intelligent insights and preventive actions using AI APIs.
Inputs:
Processed parameters from Eye and Light modules.
Outputs:
Customized wellness tips, exercise suggestions, break reminders.

Algorithm:
1. Send monitoring data to AI API (Gemini/Grok).
2. Receive adaptive recommendations.
3. Display summarized suggestions to user.

User Interface:
Recommendation panel on GUI dashboard.

Module 4: Alert & Feedback Module
Purpose:
To provide real-time notifications when strain indicators exceed healthy limits.
Inputs:
Threshold violation from detection modules.
Outputs:
Visual pop-ups, audio alerts, vibration (optional).
Algorithm:
1. Continuously monitor EAR and brightness values.
2. Trigger alert event when limits exceed.
3. Reset timer once user responds.

User Interface:
Alert window showing messages like "Blink more often" or "Take a short break."

Module 5: Data Storage and Visualization Module

Purpose:
To store monitoring data securely and visualize wellness progress through reports.
Inputs:
Daily/weekly monitoring data.
Outputs:
Charts, logs, and historical reports.
Algorithm:
1. Store user session data in the local database.
2. Generate usage trends and graphs.
3. Display data on GUI using charts and analytics widgets.

User Interface:
Statistical dashboard showing progress over time.

Business logic and Software Architecture
The system integrates the following main components:
1. Computer Vision Module:
Implemented using OpenCV and MediaPipe. This module captures the live video stream through a webcam and detects eye landmarks to compute the Eye Aspect Ratio (EAR) and blink rate in real time.

2. Ambient Light Monitoring Module:
Uses webcam brightness or connected sensors to measure the illumination of the environment and determine whether it meets recommended lighting standards.

3. AI Recommendation Module:
Connects to AI APIs such as Gemini or Grok to generate personalized insights and preventive wellness suggestions. These recommendations are displayed on the dashboard as adaptive feedback.

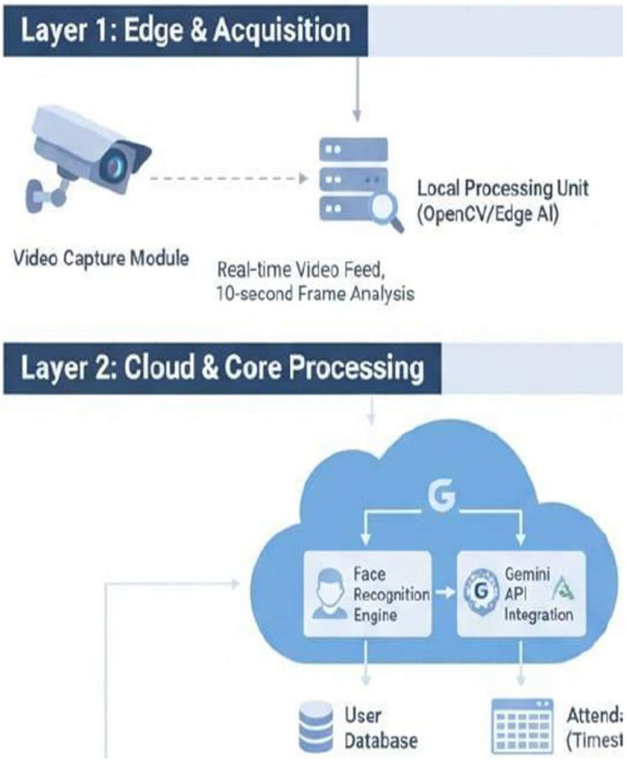
4. Graphical User Interface (GUI):
Developed using PyQt6, it provides a user-friendly interface where users can start or stop monitoring, view real-time results, and analyze historical data through charts and statistics.

5. Alert & Notification Module:

Generates audio and visual alerts when the system detects potential eye strain or poor lighting conditions, prompting the user to take breaks or make adjustments.

6. Database & Storage Module:

Stores user profiles, preferences, monitoring history, and usage reports for future reference and analytics



Advantages and Disadvantages

Advantages

1. Provides real-time monitoring of eye activity to detect early signs of strain.
2. Helps prevent Computer Vision Syndrome (CVS) with timely alerts and reminders.
3. AI-based detection increases accuracy compared to manual observation.
4. Works on normal devices without needing any specialized hardware.
5. Generates weekly and monthly reports to help users track habits.
6. Supports privacy through on-device processing of camera data.
7. Offers customizable settings such as alert interval, sensitivity, and theme.
8. Modular design allows future expansion and easy updating of the AI model.
9. Improves productivity by encouraging breaks and healthy screen usage.
10. Useful for students, remote workers, employees, and computer-intensive users.
11.

Disadvantages

1. Accuracy decreases in poor lighting or low-quality camera conditions.
2. Requires continuous camera access, which may raise privacy concerns.
3. High camera usage may drain battery on laptops or tablets.
4. Tracking may fail if the user frequently moves out of the camera frame.
5. Anti-reflective or tinted glasses can reduce detection accuracy.
6. Older systems may struggle to run real-time detection smoothly.
7. Requires stable internet for cloud-based reports or AI insights.
8. Not a medical diagnostic tool; can only assist, not replace eye specialists.
9. User discomfort may arise due to continuous monitoring.
10. Excessive alerts can annoy users if settings are not properly configured.

Applications

1. Personal Health Monitoring

Helps individuals track their screen-usage habits and detect eye strain early during daily computer or mobile use.

2. Educational Institutions

Useful for students who spend long hours attending online classes or completing assignments, helping them maintain healthy screen habits.

3. Corporate Offices & IT Companies

Employees working long hours on computers can receive timely alerts to prevent CVS and improve productivity.

4. Remote Workers & Freelancers

Supports professionals working from home by monitoring screen time and ensuring regular breaks.

5. Gaming Community

Gamers who play for extended periods can avoid fatigue and maintain better eye health.

6. Digital Creators & Designers

People involved in photo/video editing, coding, UI/UX design, or content creation can manage strain during long creative sessions.

7. Healthcare & Wellness Platforms

The system can integrate with wellness apps to provide eye-care analytics and personalized recommendations.

8. HR and Employee Wellness Programs

Companies can use aggregated (anonymized) data to track team wellness and reduce eye-strain-related complaints.

9. Libraries, Labs, and Learning Centers
Useful in public or institutional spaces where users frequently access computers for long durations.

10. Assistive Tool for Eye-Care Awareness
Helps spread awareness about CVS, healthy screen usage, and preventive eye-care practices.

VIII. SOFTWARE TESTING

Introduction
The implementation phase translates the system design into an operational software product. The Vision Shield AI-Based Eye Strain Detection System is developed using Python and AI-based computer vision libraries. Each module designed in earlier phases has been implemented as an independent, reusable component to ensure modularity and scalability.

Test cases (Unit test, Integration Test, Acceptance Test, Product Test)
Testing ensures that all components of the system perform correctly and meet the specified functional and non-functional requirements. The project underwent unit testing, integration testing, and system testing to verify performance, accuracy, and reliability.

Types of Testing

1. Unit Testing:
Each module (Eye Detection, Light Monitoring, Alerts, etc.) was tested individually to ensure correct logic, accurate detection, and proper data handling.

2. Integration Testing:
Verified that all modules communicate effectively—such as the transfer of data from the detection module to the dashboard and alert systems.

3. System Testing:
The entire application was executed under real conditions with multiple users to validate the system's behavior, response time, and user interface performance.

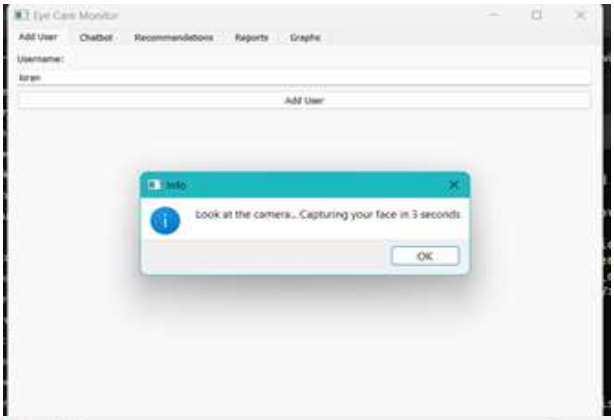
4. Performance Testing:
The system was tested on various hardware setups to ensure real-time monitoring without noticeable delay. Processing rate maintained at 25–30 frames per second.

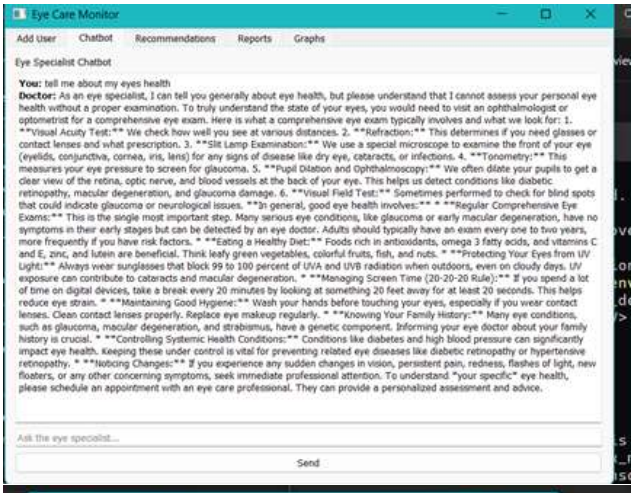
5. User Acceptance Testing (UAT):
Feedback was collected from a small group of students and professionals who used the system. The users confirmed that the alerts and reports improved their awareness of screen-time habits and reduced discomfort.

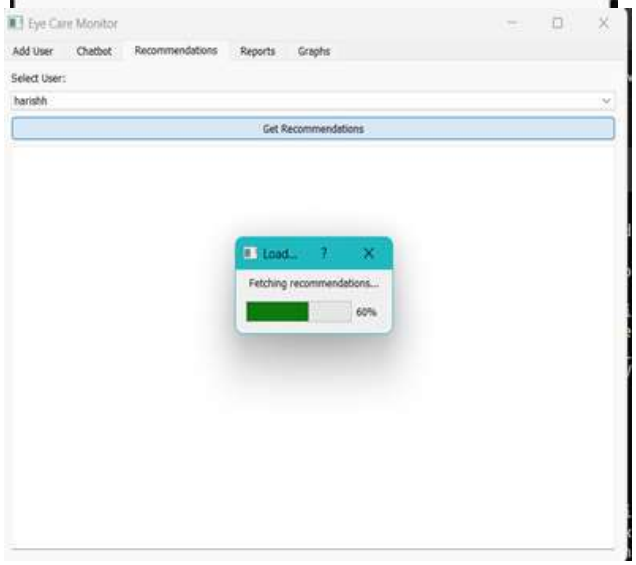
Snap shots of the test cases and test plans

Test Case ID	Test Description	Expected Output	Actual Output	Stat u s
TC01	System detects blink and calculates EAR	EAR value updates per frame	EAR calculated accurately	Pass
TC02	Ambient light below threshold	"Increase Lighting" alert displayed	Alert generated correctly	Pass
TC03	Prolonged focus detected (no blink)	"Take a Break" notification displayed	Alert displayed on time	Pass
TC04	AI Recommendation request	Personalized suggestion shown	Suggestions received from API	Pass
TC05	Data stored after session	Usage history saved successfully	Log entries visible in reports	Pass

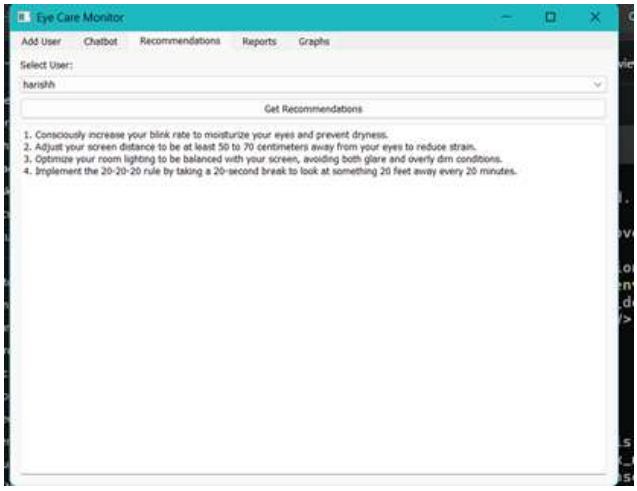
IX. RESULT AND DISCUSSION

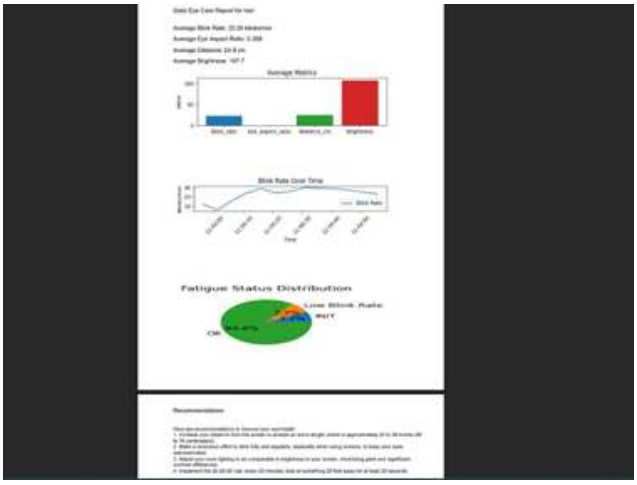

Eye Care Monitor
Add User
Chatbot
Recommendations
Reports
Graphs
Username:
Add User
Info
Look at the camera.. Capturing your face in 3 seconds
OK

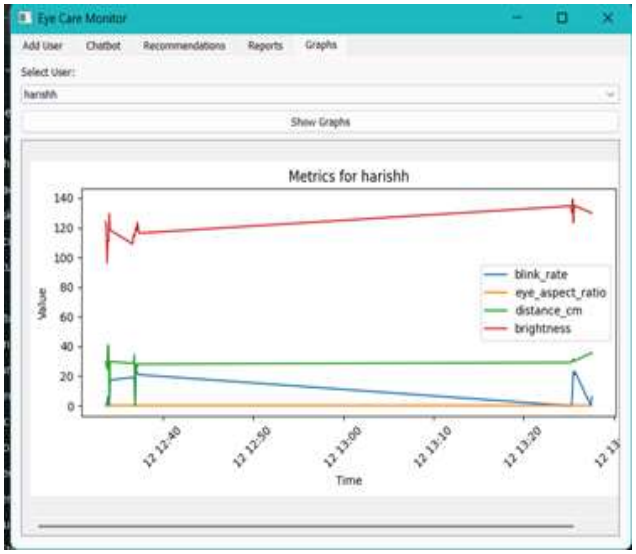

Eye Care Monitor
Add User
Chatbot
Recommendations
Reports
Graphs
Eye Specialist Chatbot
You: tell me about my eyes health
Doctor: As an eye specialist, I can tell you generally about eye health, but please understand that I cannot assess your personal eye health without a proper examination. To truly understand the state of your eyes, you would need to visit an ophthalmologist or optometrist for a comprehensive eye exam. Here is what a comprehensive eye exam typically involves and what we look for: 1. **Visual Acuity Test:** We check how well you see at various distances. 2. **Refraction:** This determines if you need glasses or contact lenses and what prescription. 3. **Slit Lamp Examination:** We use a special microscope to examine the front of your eye (eyelids, conjunctiva, cornea, iris, lens) for any signs of disease like dry eye, cataracts, or infections. 4. **Tonometry:** This measures your eye pressure to screen for glaucoma. 5. **Pupil Dilation and Ophthalmoscopy:** We often dilate your pupils to get a clear view of the retina, optic nerve, and blood vessels at the back of your eye. This helps us detect conditions like diabetic retinopathy, macular degeneration, and glaucoma damage. 6. **Visual Field Test:** Sometimes performed to check for blind spots that could indicate glaucoma or neurological issues. **In general, good eye health involves:** * **Regular Comprehensive Eye Exams:** This is the single most important step. Many serious eye conditions, like glaucoma or early macular degeneration, have no symptoms in their early stages but can be detected by an eye doctor. Adults should typically have an exam every one to two years, more frequently if you have risk factors. * **Eating a Healthy Diet:** Foods rich in antioxidants, omega 3 fatty acids, and vitamins C and E, zinc, and lutein are beneficial. Think leafy green vegetables, colorful fruits, fish, and nuts. * **Protecting Your Eyes from UV Light:** Always wear sunglasses that block 99 to 100 percent of UVA and UVB radiation when outdoors, even on cloudy days. UV exposure can contribute to cataracts and macular degeneration. * **Managing Screen Time (20-20-20 Rule):** If you spend a lot of time on digital devices, take a break every 20 minutes by looking at something 20 feet away for at least 20 seconds. This helps reduce eye strain. * **Maintaining Good Hygiene:** Wash your hands before touching your eyes, especially if you wear contact lenses. Clean contact lenses properly. Replace eye makeup regularly. * **Knowing Your Family History:** Many eye conditions, such as glaucoma, macular degeneration, and strabismus, have a genetic component. Informing your eye doctor about your family history is crucial. * **Controlling Systemic Health Conditions:** Conditions like diabetes and high blood pressure can significantly impact eye health. Keeping these under control is vital for preventing related eye diseases like diabetic retinopathy or hypertensive retinopathy. * **Noticing Changes:** If you experience any sudden changes in vision, persistent pain, redness, flashes of light, new floaters, or any other concerning symptoms, seek immediate professional attention. To understand *your specific* eye health, please schedule an appointment with an eye care professional. They can provide a personalized assessment and advice.
Ask the eye specialist...
Send


Eye Care Monitor
Add User
Chatbot
Recommendations
Reports
Graphs
Select User:
harishh
Get Recommendations
Load...
Fetching recommendations...
60%


Eye & Fatigue Monitor
User: hari
EAR: 0.28
Blinks: 7
Blinks/min: 13.5
Distance: 42.0 cm
Light: 124.8
0593 24184 inference_feedback_manager.cc:114] Feedback manager requires a mo
abling support for feedback tensors.


Eye Care Monitor
Add User
Chatbot
Recommendations
Reports
Graphs
Select User:
harishh
Get Recommendations
1. Consciously increase your blink rate to moisturize your eyes and prevent dryness.
2. Adjust your screen distance to be at least 50 to 70 centimeters away from your eyes to reduce strain.
3. Optimize your room lighting to be balanced with your screen, avoiding both glare and overly dim conditions.
4. Implement the 20-20-20 rule by taking a 20-second break to look at something 20 feet away every 20 minutes.


Fatigue Status Distribution



X. DEPLOYMENT AND MAINTENANCE

The Vision Shield system is designed for smooth deployment on Windows-based machines and requires minimal technical expertise for installation or maintenance. Regular updates ensure accuracy, improved AI performance, and enhanced user experience.

Installation and Un-installation Installation

1. The user downloads the Vision Shield setup file from the official distribution link or installer package.
2. The installation wizard guides the user through simple steps such as selecting the installation folder and agreeing to the terms.
3. Required dependencies, including Python runtime and system libraries, are automatically configured during installation.
4. After installation, the application creates a shortcut on the desktop and in the start menu for easy access.
5. On first launch, the system verifies the camera, initializes AI modules, and allows the user to set preferences such as alert interval and theme.

Un-installation

1. The user navigates to Control Panel → Programs and Features or Settings → Apps.
2. Vision Shield is selected from the installed applications list.
3. Clicking "Uninstall" removes all system files, logs, temporary datasets, and related dependencies safely.
4. User preferences and reports can be optionally retained or deleted based on user choice.
5. After completion, the system verifies that no background services or residual files remain.

User Help

The system provides multiple user-support options to ensure a smooth experience.

1. **Help Menu Inside Application**

A built-in help section explains features such as how to enable monitoring, adjust sensitivity, change settings, and interpret reports.

2. **Tooltips and On-Screen Guidance**

Small pop-up explanations appear when the user hovers over buttons or settings, making the interface beginner-friendly.

3. **User Manual (PDF)**

A downloadable manual provides step-by-step guidance on installation, setup, usage, troubleshooting, and common FAQs.

4. **In-App Alerts and Notifications**

Smart alerts guide the user when lighting is poor, the camera is blocked, or the system cannot detect eyes.

5. **Technical Support (Email/Chat)**

Users can contact support for issues related to installation, configuration, or performance problems.

6. **Tutorial Video (Optional)**

A short walkthrough video helps users understand all features visually.

XI. CONCLUSION AND FUTURE SCOPE

Conclusion

The Vision Shield AI-Based Eye Strain Detection System addresses one of the most common challenges of the digital era—Computer Vision Syndrome (CVS)—by combining artificial intelligence, computer vision, and data analytics.

Through the use of AI-powered monitoring, real-time alerting, and adaptive recommendations, the system successfully demonstrates that preventive digital wellness solutions can be automated efficiently and reliably.

Key achievements include:

- Accurate real-time fatigue detection through visual parameters.
- Integration of AI for personalized recommendations.
- Simple and intuitive user interface suitable for all users.
- Modular and scalable architecture for future expansion.

The project thus fulfills its objective of developing an intelligent tool that enhances productivity, reduces fatigue, and promotes long-term eye health in daily screen-based activities

Future Scope

The system, though fully functional, can be expanded with several enhancements to make it more powerful, accurate, and user-centered in future versions. The potential future developments include:

1. **Integration with Wearable Devices:**
Support for smart glasses or headbands to collect more accurate eye movement and posture data.
2. **Mobile Application Version:**
Development of an Android/iOS version for broader accessibility and background monitoring on portable devices.
3. **Cloud-Based Data Storage:**
Implementation of cloud analytics for storing large-scale data, enabling comparison across time and generating predictive reports.
4. **Advanced AI and Machine Learning Models:**
Use of deep learning models for detecting fatigue patterns with higher precision and early symptom prediction.
5. **Voice Assistant Integration:**
Inclusion of a voice interface for verbal reminders and wellness suggestions.
6. **Healthcare Collaboration:**
Integration with optometrists or wellness professionals for data-driven preventive care and digital wellness recommendations.

Summary

The Vision Shield Project has successfully demonstrated how AI-driven technology can be applied to real-world health and wellness problems. It establishes a base for future research and development in AI-assisted digital health monitoring, creating opportunities for safer and more sustainable screen usage.

The system not only fulfills all academic and functional requirements but also provides a meaningful step toward improving digital well-being in the modern world.

REFERENCES

1. Smith, J., & Brown, L. (2022). "Effects of Prolonged Screen Time on Eye Health,"
Journal of Ophthalmology, Vol. 15, No. 3, pp. 45–58.
2. A. Rezi and M. Allam, "Techniques in Array Processing by Means of
Transformations," in Control and Dynamic Systems, Vol. 69, Multidimensional Systems,
C. T. Leondes, Ed. San Diego: Academic Press, 1995, pp. 133–180.
3. G. O. Young, "Synthetic Structure of Industrial Plastics," in Plastics, 2nd ed., Vol. 3,
J. Peters, Ed. New York: McGraw-Hill, 1964, pp. 15–64.
4. Microsoft Developer Network, "Windows Vision and Sensor APIs Documentation,"
5. Google Developers, "MediaPipe Framework – Face and Eye Landmark Detection,"
6. Apple Developer Documentation, "App Analytics and Screen Time API,"
7. OpenCV Documentation, "Real-Time Computer Vision Library for AI-Based Detection
8. Chart.js Documentation, "JavaScript Charting Library for Web Visualization,"
9. D3.js Documentation, "Data-Driven Documents for Dynamic Visualization,"
10. Visual Paradigm Knowledge Base, "UML Diagram Types and Design Guidelines,"
11. PlantUML Official Guide, "Open-Source UML Diagram Syntax and Examples,"
12. K. K. Wagh Polytechnic, Department of Computer Technology, Project Report Format 2025–26, Internal Document, Nashik, 2025.
13. MediaPipe Team, "Eye Aspect Ratio (EAR) and Facial Landmark Detection Using AI Models,"
14. Gemini AI API Documentation, "Adaptive AI API for Personalized Insights,"
15. OpenAI Research, "AI-Powered Recommendation Systems and Human-Centered Design,"