Software Engineering Project

Human Computer Interaction Using Eye Movement Tracking and Predictability Computation

SUBMITTED BY

Nitesh Verma (23001570034)

Awani Yadav (23001570071)

Tanisha Verma (23001570057)

B.Sc. (Hons) Computer Science, IIIrd Year

2025

Under the guidance of
Prof. Vibha Gaur

Department of Computer Science
ACHARYA NARENDRA DEV COLLEGE

ACKNOWLEDGEMENT

We, Nitesh Verma, Awani Yadav and Tanisha Verma, carried out this project as part of the Software Engineering Project under the guidance of Prof. Vibha Gaur. We have made efforts in this project. However, it would not have been possible without the kind support and help of many individuals.

We are highly thankful to Prof. Vibha Gaur for their guidance and constant supervision as well as for providing necessary information regarding the project and also for their support in completing the project. Our thanks and appreciation also go to all members of the Department of Computer Science for their support and encouragement throughout this project.

Nitesh Verma	Awani Yadav	Tanisha Verma
23001570034	23001570071	23001570057
B.Sc. (Hons) CS	B.Sc. (Hons) CS	B.Sc. (Hons) CS
IIIrd Year	IIIrd Year	IIIrd Year

ACHARYA NARENDRA DEV COLLEGE

(University of Delhi)

CERTIFICATE

This is to certify that Nitesh Verma, Awani Yadav and Tanisha Verma, students of B.Sc. (Hons) Computer Science IIIrd Year, Acharya Narendra Dev College, University of Delhi, have completed the Software Engineering project titled 'Human Computer Interaction Using Eye Movement Tracking and Predictability Computation" to fulfill the credit of the Software Engineering project making for practical marks of the subject. The project submitted is a report of original work carried out by Nitesh Verma, Awani Yadav and Tanisha Verma.

Nitesh Verma Awani Yadav Tanisha Verma

Supervisor
Prof. Vibha Gaur

Abstract

Human Computer Interaction Using Eye Movement

Tracking and Predictability Computation The project "Human Computer Interaction Using Eye Movement Tracking and Predictability Computation" introduces a hands-free interaction system that enables users to operate a computer solely by using their eye movements.

The system works by:

- Capturing real-time video through a standard webcam.
- Processing the video with computer-vision techniques to detect facial landmarks, track gaze direction, and identify user intent.
- Providing a virtual keyboard with large keys for text input, utilizing gaze focus combined with blink-based or dwell-time selection.
- Integrating a predictability computation module that analyzes past user interactions to suggest the most probable next character or action.

This innovative approach is designed to enhance accessibility for individuals with motor impairments, support touch-free computing, and improve typing speed and efficiency by reducing user effort. The system demonstrates a low-cost solution, relying on readily available, open-source tools.

Contents

List of Figures

Chapter 1

Problem Statement

Traumatic bilateral hand loss is extremely rare, affecting fewer than 4 people per 100,000 worldwide [1]. Despite its rarity, the impact on daily life is profound, highlighting the critical need for supportive technologies, advanced prosthetics, and inclusive design to enhance independence and quality of life.

This also leads to the need for better technologies which help people to interact with computers without using their hands. One of the most commonly accepted ways is to track eye movement and use it as input for human-computer interaction. [2]

Eye trackers have existed for a number of years, but in the early stage when eye tracking systems developed they are confined for laboratory experiments. The Eye trackers that time was only, used in the study of the nature of human eye movements rather than using it as a real-time input medium for Human-Computer Interaction(HCI). [3].

Initially Eye trackers were too expensive, but with the development of better and cheaper components for gaze interaction, low-cost eye trackers have been produced by several high-profile companies, such as Tobii's EyeX tracker [4], GazePoint's GP3 tracker [5],and the EyeTribeTracker [6]

As eye tracking gear gets cheaper, new applications with the concept of using eye tracking in Human-Computer Interaction are clearly beginning to blossom. [7] [8] [9]

Although, all of the currently existing eye tracking software used in Human-Computer Interaction requires high quality eye-tracker hardwares, for better accuracy and precision. This is a extra harware cost for users.

This led to the need of a software which can perform better on general available cameras.

The given project focuses on making software for eye tracking, which can work using only general available cameras. Since the use of general available cameras might reduce the eye-tracking input quality, the project aims to add predictive modeling which helps computers to predict the next click or the next action to be performed by the user. This way the accuracy _ precision of software can be enhanced and in last it helps in reducing extra hardware cost for users.

Chapter 2

Process Model

The software development for Human–Computer Interaction using Eye Movements with Predictive Modeling will follow the Agile Process Model. Agile allows the system to be developed incrementally, where the core eye-tracking functionality is implemented first, and predictive modeling is added in later sprints. This ensures that a working prototype is available early and can be continuously improved through user feedback.

2.1 Project Vision

The goal of the project is to create a system where users can control a computer interface using eye movements (cursor movement, blink/dwell selection, and typing on a virtual keyboard). Later, predictive modeling will be integrated to predict the user's next action or click, improving efficiency and usability.

2.2 Product Backlog (User Stories)

As a user, I want to control the cursor using my eye movements, I want to select items by blinking or hovering my gaze, I want to type using a virtual keyboard controlled by my eyes, I want the system to predict my next action/click so that interaction becomes faster. As a researcher, I want to record and analyze user gaze data and prediction logs.

2.3 Sprint Plan

A Sprint plan is a detailed outline created at the start of a sprint in Agile development. It defines the sprint goal, the tasks to be completed, and how the team will achieve them within the sprint duration. The plan is made during the Sprint Planning Meeting, where the team selects items from the product backlog and commits to delivering them by the end of the sprint. A Sprint plan made by development team of the project is :

Sprint 1 – Eye Tracking Basics

Implement real-time eye gaze tracking (left, right, up, down).
Show cursor movement according to gaze direction.
Deliverable: Prototype with only eye-controlled cursor movement.

Sprint 2 – Eye Selection Methods

Add blink detection for selection (blink = click).
Add dwell-time selection (staring for fixed time = click).
Deliverable: Eye-controlled clicking demo.

Sprint 3 – Eye-Controlled Typing

Develop an on-screen virtual keyboard.
Enable typing using eye gaze + blink/dwell selection.
Deliverable: Functional typing interface controlled by eyes.

Sprint 4 – Predictive Modeling

Collect history of user gaze and actions.
Implement a basic predictive model (e.g., statistical or ML-based).
Deliverable: Prototype that predicts and highlights likely next action/click.

Sprint 5 – Advanced Prediction & Optimization

Improve prediction accuracy with more advanced algorithms.
Integrate predictions into UI (suggest next button, predictive text in keyboard).
Deliverable: Fully integrated predictive HCI system.

Sprint 6– Continuous Improvement

Add personalization so that prediction adapts to each user.
Optimize performance for accuracy and low latency.
Deploy across multiple platforms (Windows/Linux).
Deliverable: Production-ready HCI system with prediction.

2.4 Agile Practices

Sprint Duration: 2–3 weeks.

Daily Scrum Meetings: Short discussions to track progress.

Sprint Review: Demo of completed features to users and stakeholders.

Sprint Retrospective: Identify improvements for the next sprint.

Continuous Feedback: Collect feedback from real users to refine features.

2.5 Outcome

By following the Agile Process Model, the project will deliver: A working eye-tracking prototype early in development.

A fully interactive HCI system with cursor, click, and typing support.

A predictive modeling extension that makes the interaction faster and smarter.

A system that evolves through user feedback and continuous improvement.

Chapter 3

REQUIREMENT ANALYSIS & MODELING

Requirement analysis and modeling is a crucial phase in software development where the needs and expectations of users are identified, analyzed, and clearly documented. It involves gathering detailed information about what the system should do and defining both functional and non-functional requirements. Once the requirements are understood, modeling techniques such as use case diagrams, data flow diagrams, and entity-relationship models are used to represent the system's structure and behavior visually. This helps developers, stakeholders, and designers gain a shared understanding of the system before actual development begins, reducing errors and improving project efficiency.

3.1 Data Flow Diagram

A Data Flow Diagram (DFD) is a visual tool used to show how data moves through a system. It illustrates the flow of information between processes, data stores, and external entities, helping to understand how inputs are transformed into outputs. DFDs simplify complex systems by breaking them into smaller, manageable parts, making analysis and design clearer and more efficient.

3.1.1 Context Level DFD

A Context Level DFD is the highest-level data flow diagram that shows the entire system as a single process. It highlights the system's interactions with external entities and the flow of data between them, providing a clear overall

view of the system's boundaries. A similar context level DFD is made while requirement analysis of the project and is shown below in figure 3.1

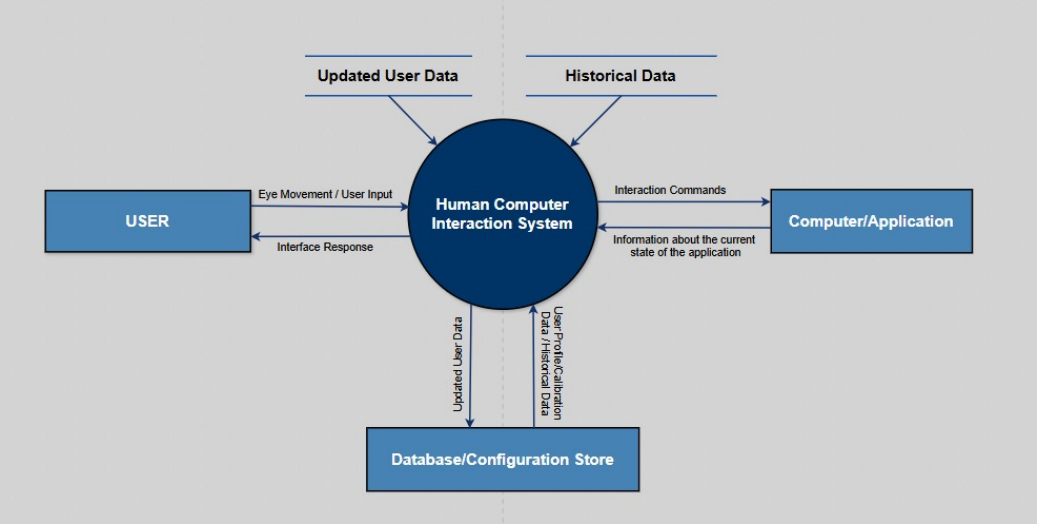

Figure 3.1: Context Level Data Flow Diagram for Eye-Movement Based HCI with Predictive Modeling

3.1.2 Level 1 DFD

A Level 1 DFD breaks down the main process from the context diagram into smaller sub-processes. It shows detailed data flows between these processes, data stores, and external entities, providing a clearer view of how the system's internal operations work. A similar Level 1 DFD is made while requirement analysis of the project and is shown below in figure 3.2

3.2 Data Dictionary

Raw Eye Data = Gaze Coordinate + Timestamp + Pupil Size + Confidence Score

Processed Gaze Data = Filtered Gaze Coordinate + Timestamp + Fixation Status + Saccade Status

User Intention & Prediction = Predicted Action + Target Object + Confidence Level

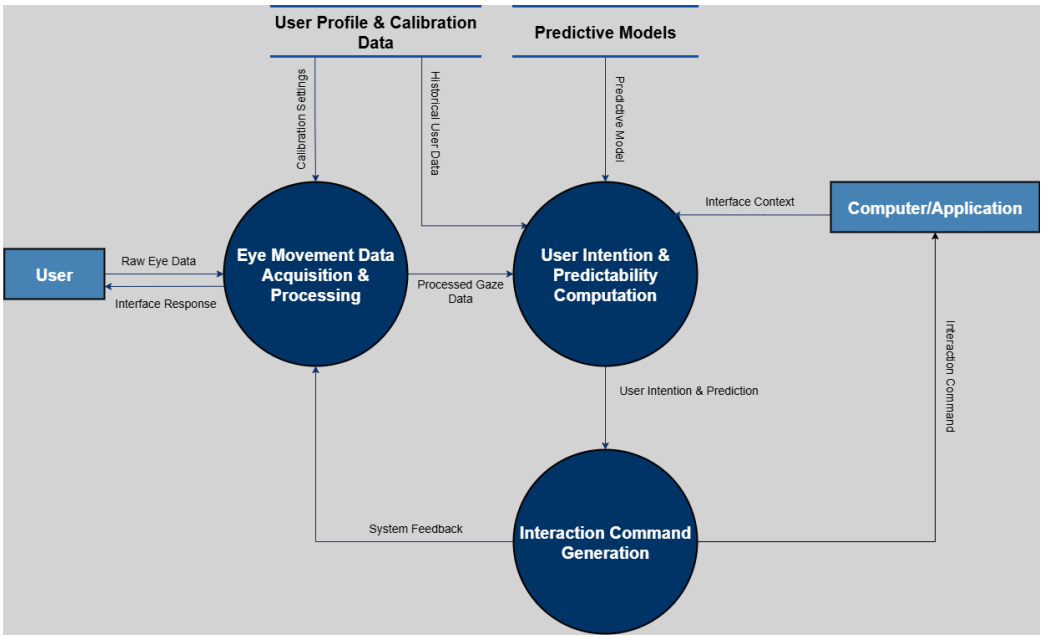

Figure 3.2: Level 1 Data Flow Diagram for Eye-Movement Based HCI with Predictive Modeling

Interaction Command = Command Type + Command Parameters

Interface Context = Screen Layout + UI Element Properties + UI Element Events

Historical User Data = Processed Gaze Data + Interaction Command + Date

System Feedback = Feedback Type + Message

Calibration Settings = User ID + Calibration Points + Screen Resolution

Predictive Model = Model File + Model Metadata

User Profile Record = User ID + Demographics + Calibration Settings + Historical User Data

3.3 Use Case Diagram

A Use Case Diagram is a visual representation of a system's functional requirements, showing how different users (actors) interact with the system. It identifies the main functions or "use cases" and the relationships between actors and these functions. This diagram helps developers and stakeholders understand what the system should do from a user's perspective, making it useful for capturing requirements and clarifying system boundaries.

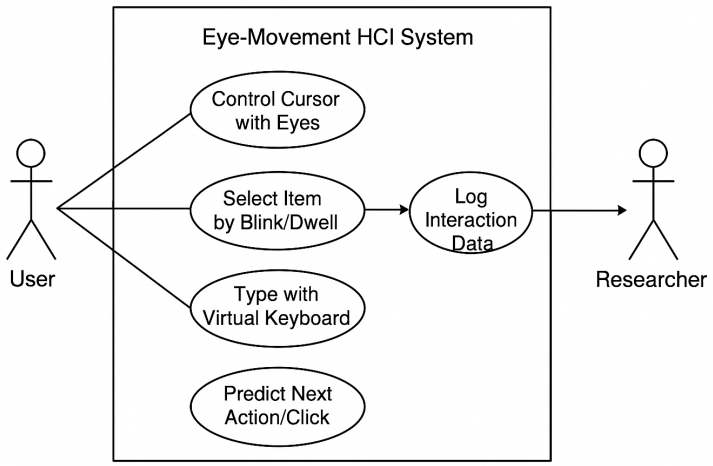

Figure 3.3: Use Case Diagram for Eye-Movement Based HCI with Predictive Modeling

Chapter 4

SOFTWARE REQUIREMENT SPECIFICATION (SRS)

4.1 Overall Description

The system "Human Computer Interaction Using Eye Movement Tracking and Predictability Computation" is designed to allow users to operate a computer using only their eye movements. The system detects gaze direction using a camera, tracks eye-movement patterns, identifies the intended screen element (such as a virtual keyboard key or clickable button), and performs actions such as selecting keys, opening files, or navigating the interface. This system aims to improve accessibility, hands-free interaction, and ease of use for individuals with motor impairments as well as provide an innovative HCI technique.

4.1.1 Product Functions

- Capture real-time video stream from the front camera. [10].
- Detect and track facial landmarks, especially eyes and pupils. [11, 12]
- Determine gaze direction and map it to on-screen coordinates.
- Provide a virtual keyboard with large keys for eye-based selection.
- Perform click selection using blink detection or dwell time. [13]
- Predict most likely next action using movement history and computation (predictability model). [14]
- Allow users to navigate and control laptop functions hands-free.

4.2 External Interface Requirements

4.2.1 User Interface

- A clean GUI with a virtual keyboard containing large, spaced-out keys for easy eye-based selection.
- Highlighting of keys or UI elements when the user focuses their gaze on them.
- Visual feedback for selection (e.g., color change or magnification).
- Calibration screen for eye-tracking accuracy. [13]
- Settings panel to modify dwell time, blink sensitivity, and keyboard size.

4.2.2 Hardware Interface

- Standard laptop/PC webcam (720p or higher recommended).
- CPU capable of running real-time computer-vision algorithms.

Supported Hardware

- Webcam
- Laptop keyboard/mouse (for backup)
- GPU (optional for better performance)

4.2.3 Software Interface

- Python 3.8+
- OpenCV for video processing [10]
- Dlib or Mediapipe for facial landmark detection [11, 12]
- PyAutoGUI for system-level actions
- Tkinter / PyQt / OpenCV GUI for on-screen keyboard
- Operating System: Windows / Linux

4.3 Functional requirements

4.3.1 FR1: Eye Tracking Requirement

- The system shall detect eyes and track gaze in real time. [13]
- The system shall map gaze direction to specific screen areas.
- The system shall recognize blinks or dwell-time focus as a "click" action.

4.3.2 FR2: Virtual Keyboard Requirement

- The system shall display a virtual keyboard with large, high-contrast keys.
- The system shall allow typing using eye gaze.
- The system shall show predictive text suggestions based on typed characters. [14]

4.3.3 FR3: Predictability Computation Requirement

- The system shall store user's previous gaze selections.
- The system shall predict the next likely character or UI action using a simple predictive model. [14]
- Predicted elements shall be highlighted for faster selection.

4.3.4 FR4:System Navigation Requirement

- The system shall allow opening/closing applications using gaze-based UI controls.
- The system shall allow scrolling using eye movement patterns.
- The system shall allow selecting menu options without physical touch.

4.4 Performance Requirements

4.4.1 Performance Requirement 1

- The system shall process camera input at a minimum of 15 frames per second to maintain smooth gaze tracking.

4.4.2 Performance Requirement 2

- Gaze detection accuracy should be at least 85% under normal lighting conditions.

4.4.3 Performance Requirement 3

- The system should register a selection within 500–800 milliseconds of focused gaze (dwell time) or blink detection.

4.5 Design Constraints

- System must run on hardware with standard webcams; no special sensors required.
- Real-time processing must be optimized to reduce CPU usage.
- Lighting conditions may affect detection accuracy, so environment constraints apply.
- The system must adhere to accessibility design standards. [15]
- Should use open-source libraries only, to reduce project cost.

Chapter 5

ESTIMATIONS

5.1 Function Points

Function Point Analysis (FPA) is used to estimate the size of the software based on the functionalities provided to the user. For the project **"Human Computer Interaction Using Eye Movement Tracking and Predictability Computation"**, the function points come from:

- Inputs such as webcam video stream, calibration data, and user settings
- Outputs including gaze detection, key selection, predictive suggestions, and UI feedback
- Internal files like gaze history and user configuration
- External interfaces such as pre-trained eye-tracking models and OpenCV [10]/Dlib libraries. [11, 12]

Based on these components, the project includes several user-focused functions such as gaze tracking, eye-blink selection [13], virtual keyboard display, prediction module [14], and navigation controls. Therefore, the system's total function points indicate that this project falls under a medium-sized software module in terms of complexity.

5.2 Effort

Effort estimation gives an idea of how much work (in person-hours or person-days) is required to develop the system. Considering the modules developed—video processing, facial landmark detection, gaze estimation [13], virtual keyboard design, prediction logic [14], and system integration—the project requires:

- Effort for research and study of eye-tracking frameworks
- Effort for implementing and testing computer-vision models
- Effort for GUI creation, predictive logic, and calibration features
- A significant amount of testing under different lighting conditions

Overall, the project requires moderate development effort, typically ranging from 4–6 weeks for a small academic team, depending on familiarity with computer vision and Python frameworks.

5.3 Cost

Cost estimation defines the resources required to complete the project. Since this is an academic project developed using open-source tools, the cost is minimal.

The project uses free tools such as:

- Python
- OpenCV [10]
- Dlib / Mediapipe. [12]
- PyAutoGUI
- Tkinter or PyQt
- Standard laptop webcam

Thus, the cost mainly includes:

- Developer time
- System hardware availability
- Testing environment such as proper lighting for accurate eye tracking

Overall, the project cost is very low because no paid software, licensed tools, or specialized hardware were required. [10]

Chapter 6

SCHEDULING

Scheduling is a crucial aspect of project management that involves planning, organizing, and controlling all project activities within a defined timeframe. It specifies when each task should begin and end, who is responsible for completing it, and how resources such as time, manpower, and equipment will be allocated. A well-structured schedule helps coordinate team efforts, track progress, and identify potential delays or bottlenecks early. By providing a clear timeline and sequence of activities, scheduling ensures that the project moves smoothly, stays on track, and meets its goals efficiently within the planned duration.

6.1 Timeline Chart

A timeline chart is a visual tool used to represent the sequence and duration of tasks or events over a specific period. It helps illustrate when activities start and finish, making it easier to understand the overall project schedule and dependencies between tasks. Timeline charts are often used in project management to track progress, set milestones, and ensure that deadlines are met. By providing a clear, time-based overview, they help teams stay organized and maintain focus on key deliverables throughout the project. The timeline of the project is of approx 5 weeks and is shown below in figure 6.1 in detail.

Legend for the chart given below is:

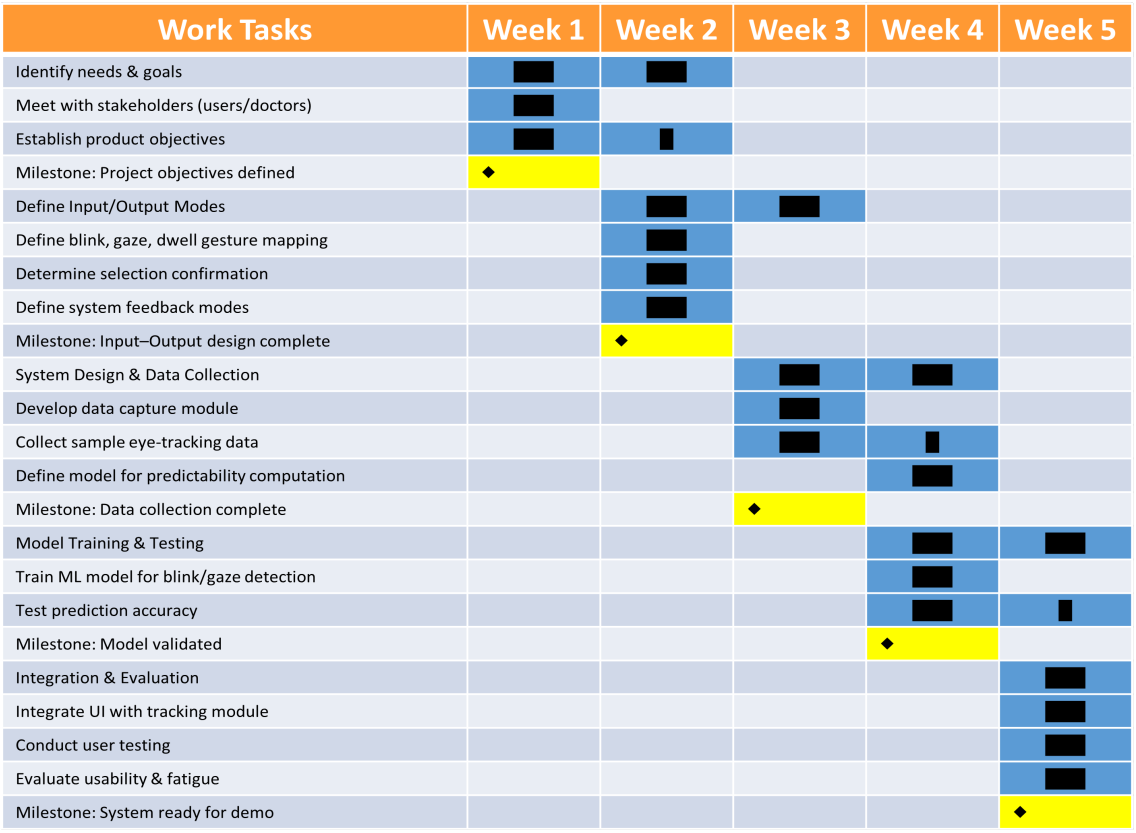

Figure 6.1: Timeline Chart – HCI using Eye Movement Tracking

Chapter 7

RISK MANAGEMENT

Risk management is the process of identifying, assessing, and controlling potential risks that could affect a project's success. It involves analyzing the likelihood and impact of risks, developing strategies to minimize or mitigate them, and monitoring risks throughout the project lifecycle. Effective risk management helps ensure projects stay on track, reduces surprises, and improves decision-making and overall project outcomes.

7.1 Risk Table

The project is gone through under risk management analysis, where some risks, their categories, their probability to occur and some of the related solutions were identified. The Risk table made is shown in figure 7.1

Impact Legend & Key Priorities

Impact values:
1 — Catastrophic (system unusable, legal/privacy harm)
2 — Critical (major functionality degraded)
3 — Marginal (noticeable but manageable)
4 — Negligible (minor inconvenience)

Top 5 Risk Priorities:
1. Reduce misdetections and false positives
2. User calibration and adaptive models
3. Privacy and data protection
4. Usability and fatigue testing
5. Training and fallbacks

Risk	Category	Probability	Impact	RMMM
Eye-tracking mis-detection (blinks misread)	TE	60%	1	
High false positives for natural blinks	US	55%	1	
Physiological variation reduces accuracy	HW/US	45%	2	
Poor performance in lighting variation	EN	50%	2	
Latency too high – sluggish experience	TE	40%	2	
Privacy/data leakage risk	SE	30%	1	
Regulatory/accessibility compliance issues	RG	25%	2	
User fatigue/resistance to adoption	US	50%	2	
Inadequate training of users/caregivers	PS	60%	3	
Staff inexperienced in ML/eye-tracking	ST	40%	2	
Hardware supply/cost issues	BU/HW	35%	3	
Integration issues with OS/assistive tech	TE/PS	30%	2	
Funding shortfall or deadline cuts	CU/BU	20%	1	
Security vulnerabilities (updates)	SE	20%	1	
Dataset bias causing unfair predictions	TE/Ethics	35%	2	
Eye strain/medical discomfort risk	US/Safety	10%	1	

Figure 7.1: Risk Table : made under risk management analysis

Bibliography

[1] Bei Yuan, Dong Hu, Suxi Gu, Songhua Xiao, and Fei Song. The global burden of traumatic amputation in 204 countries and territories. *Frontiers in Public Health*, 11:1258853, October 2023.

[2] Q. Sun, J. Xia, N. Nadarajah, T. Falkmer, J. Foster, and H. Lee. Assessing drivers' visual-motor coordination using eye tracking, gnss and gis: a spatial turn in driving psychology. *Journal of Spatial Science*, 61(2):299–316, 2016.

[3] R. J. K. Jacob and K. S. Karn. Eye tracking in human-computer interaction and usability research: Ready to deliver the promises. In J. Hyönä, R. Radach, and H. Deubel, editors, *The Mind's Eye: Cognitive and Applied Aspects of Eye Movement Research*, volume 2, pages 573–605. Elsevier Science, Amsterdam, 2003.

[4] Tobii EyeX. Eyex. `http://www.tobii.com/eyex/`, 2014. Accessed: YYYY-MM-DD.

[5] GazePoint. Gazept. `http://www.gazept.com/category/gp3-eye-tracker`, 2013. Accessed: 2025-09-01.

[6] The Eye Tribe. Eyetribe. `http://www.theeyetribe.com`, 2014. Accessed: 2025-09-01.

[7] Mohamad A. Eid, Nikolaos Giakoumidis, and Abdulmotaleb El Saddik. A novel eye-gaze-controlled wheelchair system for navigating unknown environments: case study with a person with als. *IEEE Access*, 4:558–573, 2016.

[8] Y.-M. Cheung and Q. Peng. Eye gaze tracking with a web camera in a desktop environment. *IEEE Transactions on Human-Machine Systems*, 45(4):419–430, 2015.

[9] L. Sun, Z. Liu, and M.-T. Sun. Real time gaze estimation with a consumer depth camera. *Information Sciences*, 320:346–360, 2015.

[10] Opencv documentation. https://docs.opencv.org, 2023.

[11] Dlib facial landmark detection. http://dlib.net/, 2023.

[12] Google mediapipe eye and face tracking. https://developers.google.com/mediapipe, 2023.

[13] Robert J.K. Jacob. The use of eye movements in human–computer interaction techniques. In *ACM SIGGRAPH*, 1993.

[14] I. Scott MacKenzie. *Text Entry Research*. Morgan & Claypool, 2012.

[15] Alan Dix. *Human-Computer Interaction*. Pearson, 3 edition, 2004.