THE UNIVERSITY OF BURDWAN

6TH SEMESTER DSE-4

DISSERTATION / PROJECT WORK DOCUMENTATION

EYE CONTROLLED MOUSE

Author

SK FIROJ

Roll No.: 200310400125

Registration No.: 202001001314

of 2020-21

Session: 2022-2023

Supervisors

DR. SURAJIT MONDAL	BENOJIR AHMED	RUSA SINHA ROY	SWATI SEN
Head of the Dept.	Asst. Professor	Asst. Professor	Asst. Professor

Preface

The principal motive of any project is to focus on a particular matter that requires concern in order to solve some real-world problems in real time. This project on “Mouse control through Eye” is an implementation of Computer Vision in AI that takes input through web camera, derive meaningful information from the input, detects and track user’s eye and allows users to control the mouse cursor using their eye position captured through the webcam. This project has been prepared as a curriculum of DSE-4 paper of B.Sc. Computer Science (Hons.) under the University of Burdwan. Although a whole Semester had been allotted for this cause, the development of the ideas took some time. There might be some bugs in the codes which need more perfection. However what we can do is to thrive for perfection and settle for the best option available to us.

The project offers an alternative method for mouse control, providing a hands-free approach by utilizing eye tracking. It has potential applications for individuals with limited mobility or for scenarios where hands-on mouse control is impractical or inconvenient.

Acknowledgement

This project on “Mouse control through Eye” has been prepared as a curriculum of DSE-4 paper of B.Sc. Computer Science (Hons.) under the University of Burdwan. We would like to express our heartfelt gratitude towards honourable Chancellor of the University, Shri Jagdeep Dhankar, honourable Vice-Chancellor, Prof. Nimai Chandra Saha and all other members of the University managing committee who have helped our department to grow and flourish. We shall be grateful towards our Principal Sir, Dr. Niranjan Mondal who has always been by our side through the thick and thin. The Managing Committee of Burdwan Raj College has always guided has during our entire journey to date.

The Head of the Department, Dr. Surajit Mondal Sir has always extended his helping hand towards us, without which we would have never been able to complete this project. The entire faculty of the Department of Computer Science constituting, Mrs. Benojir Ahmed Ma’am, Mrs. Rusa Sinha Roy Ma’am and Mrs. Swati Sen Ma’am have been excellent supervisors and guides. The path shown by them always remains lime lighted in our educational and career lives. They all have been an indispensable part of our education throughout these three years. We would also express our gratitude towards our parents, family members and friends for their utmost help, care, support, and co-operation. We except best wishes from our well-wishers so that this project becomes a grand success!

Sk Firoj

Sem VI, Dept. of Computer Science

Burdwan Raj College

Contents

Abstract..... 1

1. Problem Statement..... 2
2. Introduction..... 3
3. Computer Vision..... 4-8
   3.1 Role Of Machine Learning And Deep Learning
   In Computer Vision system..... 7-8
4. Project Scope & Work Statement..... 9-11
5. Intended Audience..... 12
6. Objectives of the Project..... 13
7. Project Category..... 14
8. Requirement Specification..... 15
9. Feasibility Study..... 16-17
10. SDLC Model..... 18-19
11. Requirement Analysis and Specification..... 20-21
    11.1 Functional Requirements..... 20
    11.2 Non-Functional Requirements..... 20
    11.3 User Requirements..... 21
    11.4 System Constraints..... 21
    11.5 Environmental Requirements..... 21
12. Design..... 22-29
    12.1 Project Planning..... 22-23
    12.2 DFD..... 24-25
    12.3 Flow Chart Design..... 26
    12.4 Snapshots..... 27-29
13. Coding..... 30-36
    13.1 Video Capturing Module..... 30-31
    13.2 Eye Tracking Module..... 32-33
    13.3 Mouse Tracking Module..... 34-36
14. Testing..... 37-43
    14.1 Test Objective..... 37
    14.2 Process Overview..... 37
    14.3 Testing Process..... 38
    14.4 Test Strategy..... 39-40
    14.5 Unit Testing..... 40
    14.6 Blackbox Testing..... 40
    14.7 Whitebox Testing..... 40
    14.8 Integration Testing..... 41
    14.9 Validation Testing..... 42

14.10 System Testing..... 42-43
15. Performance..... 44-45
15.1 Result Discussion..... 44-45
15.2 Acceptance level..... 45
16. Maintenance..... 46
17. Cost Estimation..... 47-48
18. Future Extension..... 49
19. Conclusion..... 50
20. Reference..... 51
21. Appendix-A..... 52
22. Appendix-B..... 53-54

Abstract

This project aims to develop an innovative computer vision-based system using Artificial Intelligence that enables users to control the mouse cursor using their eye position. By leveraging the capabilities of the 'cv2', 'mediapipe' and 'pyautogui' packages in Python. The 'cv2' package is used to capture real-time video feed from the webcam. The `mediapipe` package is employed to detect facial landmarks, with a particular focus on the positions of the user's eyes. The detected eye positions are then processed and mapped relative to the screen dimensions. Using the `pyautogui` package, the mouse cursor is moved accordingly, allowing users to control it solely through their eye movements. This project offers a hands-free alternative to traditional mouse control, providing enhanced accessibility and convenience, particularly for individuals with limited mobility or physically challenged or situations where manual mouse interaction is challenging or impractical. The results of this project show promising outcomes, highlighting the potential of eye-based cursor control as a valuable input modality in human-computer interaction.

1. Problem statement

Traditional mouse control relies on manual manipulation, which can be limiting for individuals with limited mobility or in situations where hands-on control is impractical. This project aims to address this issue by developing a computer vision-based system using Artificial Intelligence that enables users to control the mouse cursor using their eye position.

The problem to be solved involves detecting and tracking the user's eye movements accurately and translating them into corresponding mouse cursor movements on the screen. The system should provide a reliable and intuitive interface that allows users to interact with computers hands-free, opening up possibilities for enhanced accessibility and usability. The challenge lies in developing a robust and efficient algorithm that can accurately track the user's eye position in real-time using a webcam and convert that information into precise cursor movements. Additionally, the system should be adaptable to different lighting conditions, variations in eye appearance, and potential occlusions. By addressing these challenges, the project aims to offer a viable solution that revolutionizes mouse control and expands the possibilities of human-computer interaction.

2. Introduction

In today's digital age, computer interaction has become an integral part of our daily lives. Traditional mouse control has long been the standard method for navigating graphical user interfaces, but it poses limitations for individuals with limited mobility, physically challenged or in situations where manual control is impractical. This project aims to address these limitations by leveraging the power of computer vision and Artificial Intelligence to create an innovative system that enables users to control the mouse cursor using their eye position as a smart gesture.

The project employs the **'cv2'**, **'mediapipe'** and **'pyautogui'** packages in Python to develop a robust and efficient solution. By utilizing a webcam, the system captures real-time video feed using **'cv2'** package and processes it using the **'mediapipe'** package to detect and track facial landmarks, with a particular focus on the positions of the user's eyes. This allows for precise eye position estimation, even in varying lighting conditions or with different eye appearances.

Once the eye positions are accurately determined, the **'pyautogui'** package comes into play, translating the eye movements into corresponding mouse cursor movements on the screen. By mapping the eye positions relative to the screen dimensions, the system enables users to control the mouse cursor effortlessly and intuitively.

The implications of this project are far-reaching. It enhances accessibility, empowering individuals with limited mobility or physically challenged to interact with computers more effectively. It opens up new possibilities for hands-free computer control, enabling users to navigate graphical user interfaces, interact with applications, and perform tasks solely through their eye movements.

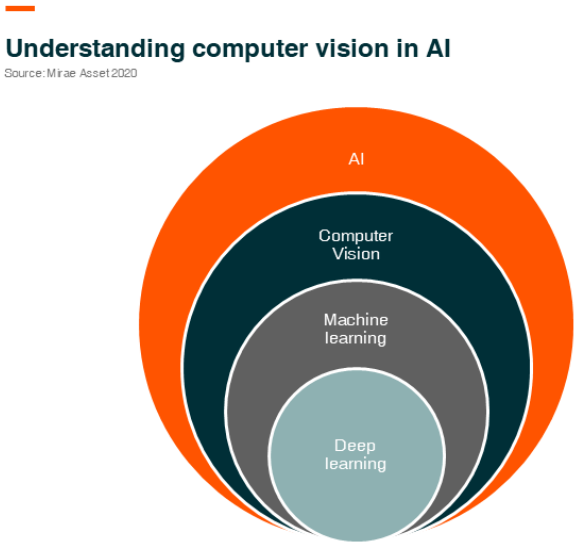
The project also has significant potential for improving productivity and efficiency. By eliminating the need for physical mouse manipulation, users can streamline their workflow, particularly in scenarios where frequent mouse interactions are required. Additionally, the system's adaptability to different lighting conditions and eye appearances enhances its reliability and usability across diverse environments.

Through the development and evaluation of a prototype, this project aims to showcase the feasibility and effectiveness of eye-based cursor control as a viable input modality in human-computer interaction. The prototype will undergo rigorous testing and performance evaluation to ensure accurate eye tracking, responsive cursor control, and seamless integration with existing computer systems.

By harnessing the power of computer vision and eye tracking technology, this project strives to redefine the way we interact with computers. It introduces a novel and accessible method of mouse control that has the potential to revolutionize human-computer interaction, enhance inclusivity, and empower individuals to navigate the digital world with ease and efficiency.

3. Computer Vison

Computer vision is a multidisciplinary field of AI that focuses on enabling computers to understand and interpret visual information from digital images or video. It aims to replicate human visual perception by developing algorithms and techniques that enable computers to analyse and extract meaningful insights from visual data using AI.



Computer vision combines methods from various domains, including image processing, pattern recognition, machine learning, and artificial intelligence. The process typically involves the following steps:

1. Acquisition of visual data: Computer vision systems obtain visual data from various sources, such as cameras, videos, or images.

2. Pre-processing : The acquired data is pre-processed to enhance its quality and facilitate subsequent analysis. This may involve operations such as noise reduction, resizing, colour correction, or normalization.

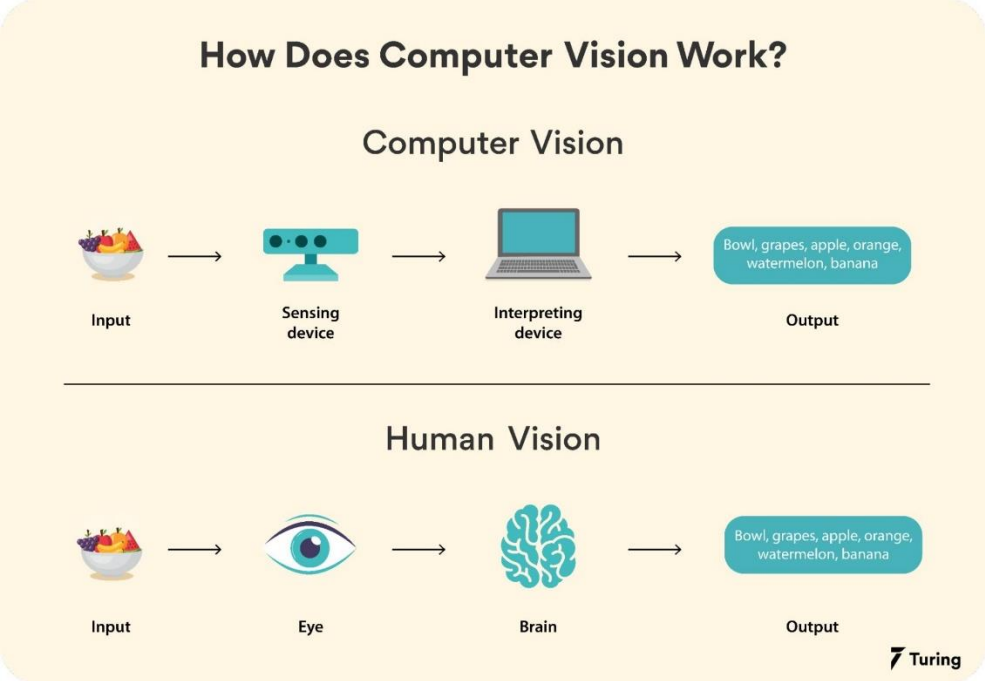
3. Feature extraction: Computer vision algorithms identify and extract relevant visual features from the pre-processed data. These features could be edges, corners, textures, shapes, or more complex attributes.

4. Object detection and recognition: By using machine learning techniques, computer vision systems can detect and recognize objects within images or video streams. This involves comparing extracted features with pre-existing models or training new models to classify objects based on their visual characteristics.

5. Image segmentation: Computer vision techniques can partition images into meaningful regions, grouping pixels or regions based on similarities in colour, texture, or other visual attributes. Segmentation enables the identification and isolation of individual objects or regions of interest within an image.

6. Motion and tracking: Computer vision algorithms can analyse sequential visual data to track objects or infer motion information. This is useful in applications such as video surveillance, autonomous vehicles, or augmented reality, where understanding object movement is crucial.

7. Scene understanding: By integrating information from multiple sources and applying higher-level reasoning, computer vision systems can interpret and understand complex scenes. This involves tasks such as object localization, scene classification, activity recognition, or scene reconstruction.

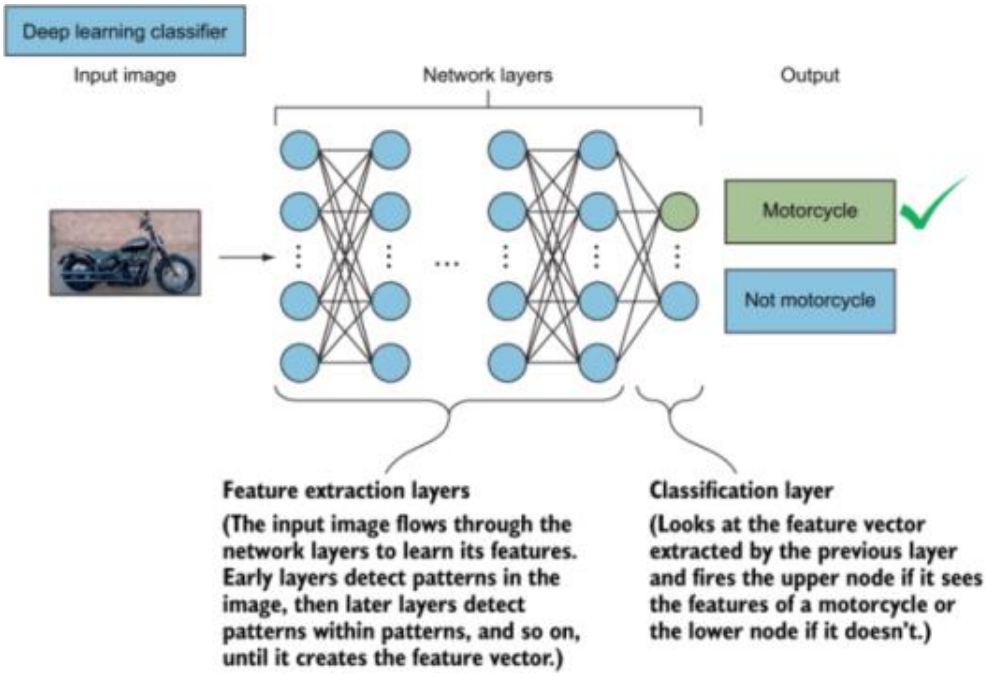


Computer vision finds applications in various fields, including robotics, healthcare, autonomous vehicles, security and surveillance, augmented reality, industrial automation, and more. Some practical examples include facial recognition, object detection in self-driving cars, medical image analysis, quality control in manufacturing, and video content analysis.

Advancements in deep learning and neural networks have significantly improved the capabilities of computer vision systems, allowing them to achieve human-level or even superhuman performance in certain tasks. However, challenges such as occlusion, lighting variations, viewpoint changes, and the need for large labelled datasets still pose research and implementation challenges in the field.

3.1 Role Of Machine Learning And Deep Learning In Computer Vision system

Machine learning and deep learning play significant roles in computer vision by enabling systems to automatically learn and understand visual data. They provide algorithms and models that can analyse and interpret images and videos, allowing computers to perceive and make sense of visual information.



Here are the key roles of machine learning and deep learning in computer vision:

1. Image Classification: Machine learning and deep learning algorithms can classify images into predefined categories or labels. By training on large datasets of labelled images, these algorithms can learn to recognize patterns and features that distinguish different objects or scenes. This enables tasks such as object recognition, face detection, and scene understanding.

2. Object Detection and Localization: Machine learning techniques, including deep detection and localization in images and videos. These models can accurately identify and localize multiple objects within an image, providing bounding box coordinates and class labels. Object detection is crucial in various applications, including autonomous vehicles, surveillance systems, and robotics.

3. Semantic Segmentation: Deep learning-based semantic segmentation models assign semantic labels to each pixel in an image, enabling fine-grained understanding of the scene. This technique is valuable in applications such as image editing, medical imaging, and autonomous systems, where precise pixel-level understanding is required.

4. Object Tracking: Machine learning algorithms, including deep learning-based trackers, can track objects over time in videos. These algorithms learn to follow the object's motion and appearance, enabling tasks such as video surveillance, action recognition, and augmented reality.

5. Generative Models: Deep learning-based generative models, such as generative adversarial networks (GANs) and variational autoencoders (VAEs), can generate new and realistic images based on learned representations. These models have applications in image synthesis, data augmentation, and image inpainting.

6. Transfer Learning: Transfer learning allows leveraging pre-trained deep learning models on large datasets to solve computer vision tasks with limited training data. Pre-trained models, such as those trained on ImageNet, can extract useful features from images, enabling better performance and faster convergence on specific vision tasks.

7. Image Captioning and Understanding: Deep learning models, including recurrent neural networks (RNNs) and transformers, can generate textual descriptions or captions for images. These models combine visual and language understanding to generate meaningful and contextually relevant descriptions.

8. Video Analysis: Deep learning models can process and analyse videos, extracting features, identifying activities, and detecting anomalies. Video-based action recognition, activity recognition, and video surveillance systems heavily rely on machine learning and deep learning techniques.

Machine learning and deep learning techniques have revolutionized computer vision by enabling computers to understand, interpret, and interact with visual data. They provide powerful tools for a wide range of applications, including object recognition, image understanding, video analysis, and visual perception in autonomous systems

4. Project Scope & Work Statement

4.1 Project Scope:

The scope of this project is to develop a computer vision-based system that allows users to control the mouse cursor using their eye position. The system will utilize the 'mediapipe' and 'pyautogui' packages in Python to enable real-time eye tracking and cursor control. The project aims to enhance accessibility and usability by providing a hands-free alternative for mouse control, particularly for individuals with limited mobility or in scenarios where manual control is impractical.

4.2 Timeline:

- Phase 1: Research and Familiarization
- Phase 2: Algorithm Development and Integration
- Phase 3: System Implementation and Testing
- Phase 4: Performance Optimization and Fine-tuning
- Phase 5: Documentation and Finalization

4.3 Work Statement:

The project will encompass the following tasks and activities:

a. Research and Familiarization:

- Conduct a literature review on eye tracking, computer vision, and related technologies.
- Familiarize with the 'mediapipe' and 'pyautogui' packages, understanding their functionalities and capabilities.
- Study existing eye-tracking algorithms and techniques for accurate eye position estimation.

b. System Design and Architecture:

- Design the overall system architecture, including the integration of 'mediapipe' and 'pyautogui' packages.
- Define the data flow and processing pipeline for real-time eye tracking and cursor control.
- Plan the user interface for visual feedback and interaction.

c. Eye Tracking Algorithm Development:

- Develop a robust eye tracking algorithm using the 'mediapipe' package.
- Implement methods for eye landmark detection, tracking, and position estimation.
- Optimize the algorithm for accurate eye position estimation under varying lighting conditions and potential occlusions.

d. Cursor Control and Mapping:

- Utilize the `pyautogui` package to control the mouse cursor based on the estimated eye positions.
- Implement mapping functions to convert eye positions to corresponding cursor movements on the screen.
- Develop methods for cursor speed and sensitivity adjustment based on eye movement patterns.

e. User Interface and Visual Feedback:

- Design and implement a user interface to display the webcam feed with annotated eye positions**.**
- Provide visual feedback on the screen, indicating the movement of the cursor based on eye tracking.
- Allow users to calibrate the system for accurate mapping between eye positions and cursor movements.

f. System Integration and Testing:

- Allow users to calibrate the system for accurate mapping between eye positions and cursor movements.
- Conduct extensive testing to evaluate the accuracy, responsiveness, and robustness of the system.
- Perform user testing with individuals of diverse backgrounds to assess usability and effectiveness.

g. Documentation and Finalization:

- Prepare comprehensive documentation detailing the system architecture, algorithms, usage instructions, and troubleshooting guidelines.
- Refine and optimize the codebase for efficiency and maintainability.
- Finalize the project deliverables, including the source code, documentation, and any necessary test datasets.

4.4 Project Deliverables:

- Functional eye-controlled mouse cursor system.
- Source code with detailed comments and documentation.
- User guide and instruction for calibration and usage.
- Test datasets for evaluating system performance.
- Project documentation summarizing the research, development process, and evaluation results.

4.5 Project Success Criteria:

Our main goal is to complete this project within this Even Semester and also within the resources allotted by our Department. It is necessary to develop a method for capturing the benefits while Eye Controlled Mouse is being developed, tested, and after it is rolled out. If the project takes a little longer to complete or costs a little more than planned, it would still be viewed as a success if it has a good payback and helps promote the institute's image as an excellent management organization.

- **Project Title:** Simplifying Navigation: Enhancing Mouse Control with Computer Vision
- **Allotted by:** The Department of Computer Science, Burdwan Raj College
- **To be Prepared by:** Sk Firoj & Ivy Kumbhakar
- **Date of Allotment:** 04/04/2023
- **Expected Date of Completion:** 17/06/2023

5. Intended Audience

The intended audience for this project includes:

1. Individuals with Limited Mobility or Physically Challenged: The eye-controlled mouse cursor system aims to benefit individuals with limited mobility, such as those with physical disabilities or conditions that make manual mouse control challenging. The project aims to enhance their accessibility and provide them with an alternative and intuitive method of computer interaction.

2. Accessibility Advocates and Researchers: Professionals working in the field of accessibility and assistive technology can benefit from this project. The eye-controlled mouse cursor system showcases the potential of computer vision and eye tracking technology in improving accessibility for individuals with disabilities. Researchers can use the project as a reference for further advancements in the field.

3. Human-Computer Interaction (HCI) Experts: HCI experts and researchers who specialize in interaction techniques and modalities can find value in this project. It demonstrates an innovative and hands-free input modality that can be applied to various HCI contexts, expanding the understanding and possibilities of interaction design.

4. Software Developers and Engineers: Developers and engineers interested in computer vision, machine learning, and human-computer interaction can gain insights and inspiration from this project. They can study the implementation details, algorithms, and integration techniques used to create the eye-controlled mouse cursor system and apply them to their own projects.

5. Assistive Technology Organizations and Institutions: Organizations and institutions dedicated to assistive technology and improving the lives of individuals with disabilities can find relevance in this project. The eye-controlled mouse cursor system can be considered as a potential assistive technology solution and can inspire the development of similar systems or contribute to ongoing research in the field.

6. General Public: While the primary focus of the project is on individuals with limited mobility, the eye-controlled mouse cursor system has the potential to intrigue and engage the general public. They can also use this project as a smart gesture system.

The project targets a diverse audience with varying backgrounds, including individuals with disabilities, accessibility experts, researchers, developers, and the general public. By addressing the needs and interests of this audience, the project aims to raise awareness about accessibility and inspire further advancements in assistive technology and human-computer interaction.

6. Objectives of the Project

- Develop an eye-controlled mouse cursor system using computer vision and AI techniques.
- Create an intuitive and user-friendly interface for calibration and interaction with the eye-controlled mouse cursor system.
- Ensure compatibility with different operating systems, screen resolutions, and webcam devices.
- Optimize the system's performance to achieve smooth and responsive cursor control.
- Conduct thorough testing and gather user feedback to assess the usability, accuracy, and reliability of the system.
- Address legal, ethical, and privacy considerations regarding user data and system usage.
- Document the project, providing clear instructions for calibration, usage, and troubleshooting.
- Demonstrate the feasibility and potential of AI-based eye tracking technology in improving accessibility and computer interaction for individuals with limited mobility.
- Enable hands-free mouse control for individuals with limited mobility or in situations where manual control is impractical.

7. Project Category

This project leverage the power of computer vision and Artificial Intelligence to create an innovative system that enables users to control the mouse cursor using their eye position as a smart gesture. It uses certain concepts of Artificial Intelligence, Artificial Neural Network, Deep Learning, Machine Learning and Computer Vision Processing in order to carry out the tasks.

Domain: COMPUTER VISION using AI

Technologies used: AI, ANN, Deep Learning, Mediapipe, Pyautogui, CV2

Language used: Python 3.9

Platform: Pycharm 2023.1.1

8. Requirement Specification

Table 8.1 Software Requirement Specification:

Operating System	Windows 8 or higher
Software Tools	Pycharm Community Edition 2023.1.1, Python IDLE 3.9 or higher
Documentation Tool	MS Office 2007 or higher
Libraries & Packages	CV2, Mediapipe, Pyautogui

Table 8.2 Hardware Requirement Specification:

HDD	10 GB or higher
SSD	2 GB or higher
Processor	Intel I3 or newer processor
RAM	2 GB or higher
Input Unit	Standard Keyboard, Standard Mouse
Output Unit	Standard Colour Monitor

9. Feasibility Study: Eye-Controlled Mouse Cursor

9.1 Technical Feasibility:

- Eye tracking technology: The technical feasibility of the project relies on the availability of accurate and reliable eye tracking technology, which is achieved through computer vision techniques and artificial intelligence (AI) algorithms. The use of computer vision, specifically the 'mediapipe' package, allows for the detection and tracking of eye positions using artificial neural networks and deep learning models.

- Computational requirements: The feasibility of the project depends on the availability of hardware with sufficient processing power and memory to handle the computational demands of real-time video processing, eye tracking algorithms, and deep learning models. GPUs and specialized hardware accelerators can enhance the feasibility by speeding up the computational tasks involved in AI-based algorithms.

- Compatibility and Integration: Ensuring compatibility with different operating systems, screen resolutions, and webcam devices is essential for the feasibility of the eye-controlled mouse cursor system. The compatibility of the respective packages with commonly used platforms enhances the feasibility of integrating computer vision and AI technologies.

9.2 Economic Feasibility:

- Cost of Hardware: The economic feasibility depends on the availability and affordability of the required hardware components, such as a webcam with suitable image quality and resolution. Additionally, the feasibility is influenced by the cost of specialized hardware accelerators, if needed, for efficient AI computations.

- Software Packages: The packages used in the project are open-source and freely available, ensuring economic feasibility in terms of software licensing.

9.3 Operational Feasibility:

- User Acceptance: The operational feasibility of the eye-controlled mouse cursor system relies on user acceptance and satisfaction. Conducting user testing and gathering feedback during the development phase can assess the system's usability, accuracy, and reliability, which are critical factors for user acceptance.

- Training and Calibration: The feasibility of the system depends on the ease of training and calibration process for individual users. The utilization of artificial neural networks and deep learning techniques in the eye tracking algorithms should facilitate the system's adaptability to different users, enhancing operational feasibility

9.4 Legal and Ethical Feasibility:

- Privacy and Data Security: Compliance with privacy regulations and ensuring the security of user data collected during the eye tracking process is essential. Addressing legal and ethical considerations related to data privacy and security is crucial for the feasibility and acceptance of the project.

- Ethical Use and Accessibility: Ensuring the ethical development and use of the eye-controlled mouse cursor system, while respecting the rights and dignity of individuals with limited mobility, enhances the project's feasibility and acceptance. Accessibility considerations should be incorporated into the system design to cater to diverse user needs.

9.5 Schedule Feasibility:

- Time Constraints: The feasibility study evaluates whether the desired outcomes can be achieved within the given time frame. A realistic project timeline, considering the complexity of implementing AI algorithms and deep learning models, is essential to ensure that the project can be completed within the specified schedule.

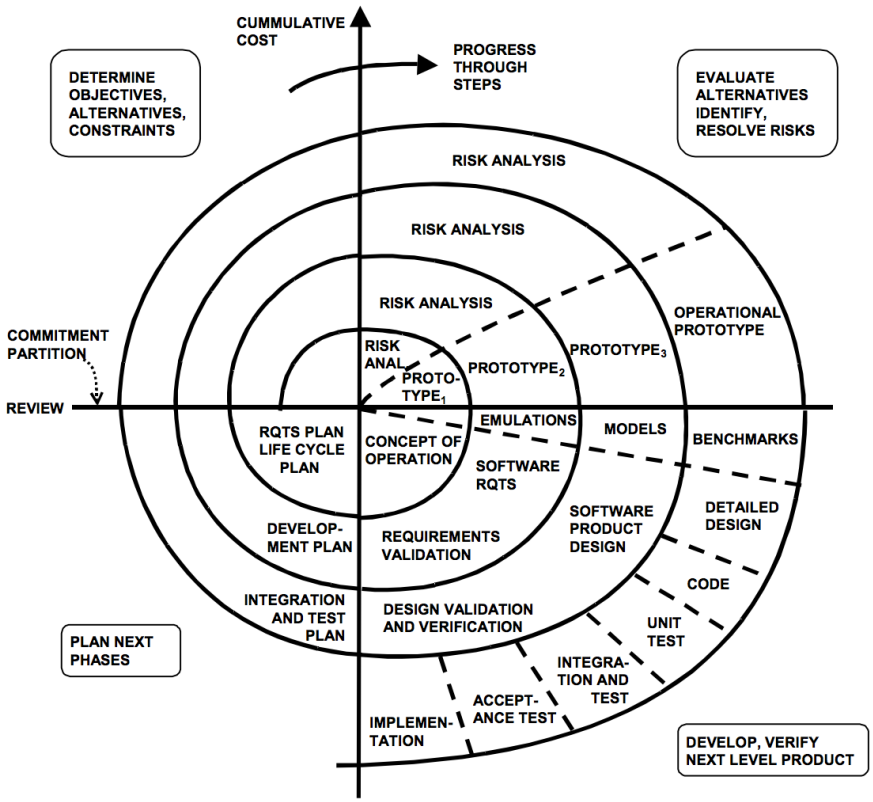
Conclusion:
Based on the technical feasibility of implementing computer vision, artificial neural networks, and deep learning techniques in the development of an eye-controlled mouse cursor system, the project is deemed feasible. However, thorough testing, user feedback, and continual refinement are necessary to address challenges and ensure the successful implementation of the system, while considering legal, ethical, and privacy considerations.

10. SDLC Model

As stated earlier, no such software product, which we are planning to build exists in the market to date. Hence, the risks that we might encounter remains unforeseen. In such circumstance dividing the work into certain phases seems to be useful. The description of each phase has been given below:

1. 1st quadrant: The objectives are investigated, elaborated, and analysed. Based on this, the risks involved in meeting the phase objectives are identified. In this quadrant, alternatives solutions possible for the phase under consideration are proposed.

2. 2nd quadrant: During the second quadrant, the alternative solutions are evaluated to select the best possible solution. To be able to do this, the solutions are evaluated by developing an appropriate prototype.



3. 3rd quadrant: Activities during the third quadrant consist of developing and verifying the next level of the software. At the end of the third quadrant consist of developing and verifying the next level of the software. At the end of the third quadrant, the identified features have been implemented and the next version of the software is available.

4. 4th quadrant: Activities during the fourth quadrant concern reviewing the results of the stages traversed so far (i.e. the developed version of the software) with the customer and planning the next iteration of the spiral.

The spiral model incorporates the systematic step-wise approach of the waterfall model. Also, it can be considered as supporting the evolutionary model- the iterations along the spiral can be considered as evolutionary levels through which the complete system is built. This enables us to understand and resolve the risks at each evolutionary level.

The project phases have been repeated until we reached a version that suits all our afore mentioned satisfaction criteria.

11. Requirement Analysis And Specification

Requirement Analysis and Specification for Eye-Controlled Mouse Cursor Project.

11.1 Functional Requirements:

- **Eye Tracking:** The system should accurately track the user's eye movements in real-time using computer vision techniques using AI.
- **Cursor Control:** The system should translate the detected eye movements into corresponding cursor movements on the screen using the respective package.
- **Calibration:** The system should provide a user-friendly calibration process to establish a mapping between eye movements and cursor movements.
- **Click Actions:** The system should support various types of click actions, such as l double-click (as of now), using eye movements or alternative input methods.

11.2 Non-Functional Requirements:

- **Performance:** The system should exhibit real-time responsiveness to ensure smooth and accurate cursor control with minimal latency.
- **Accuracy:** The eye tracking algorithm should have high accuracy in detecting and tracking the user's eye movements to ensure precise cursor control.
- **Compatibility**: The system should be compatible with different operating systems, screen resolutions, and webcam devices commonly used by users.
- **Usability:** The user interface should be intuitive, providing clear instructions and guidance for calibration and interaction with the eye-controlled mouse cursor system.
- **Security and Privacy:** The system should handle user data securely, ensuring privacy and compliance with relevant data protection regulations.

11.3 User Requirements:

- **Accessibility:** The system should enhance accessibility for individuals with limited mobility, allowing them to interact with computers using their eyes.
- **Customization:** The system should provide options for users to customize and adjust the sensitivity and behaviour of the eye-controlled mouse cursor to suit their individual preferences.

- **Ease of Use:** The system should be user-friendly, requiring minimal training and providing straightforward calibration and interaction processes.
- **Reliability:** The system should be reliable and robust, capable of operating in various lighting conditions and accommodating different user eye characteristics.

11.4 System Constraints:

- **Hardware:** The system should be designed to work with standard webcams or dedicated eye-tracking devices, ensuring compatibility and accessibility.
- **Computational Resources: The** system should be designed to operate within the constraints of typical consumer hardware, considering processing power, memory, and storage requirements.

11.5. Environmental Requirements:

- **Lighting Conditions:** The system should be capable of functioning in different lighting conditions, accommodating variations in ambient light levels.
- **Positioning:** The system should allow users to position themselves comfortably in front of the camera, considering factors like distance and angle for optimal eye tracking

The requirement analysis and specification provide a foundation for the design, development, and testing phases of the eye-controlled mouse cursor project. These requirements ensure that the system meets the expectations and needs of the target users while considering technical feasibility and usability aspects.

12. Design

12.1 Project Planning

Project planning is concerned with setting us activities, milestones and deliverables.

First of all a survey is need to be undergone for understanding the algorithm that shall be useful in our case. Consequent coding and feedback shall be helpful to correct the errors and bugs in the further phases of the project development.

After planning the project it has been evinced that the deliverables of the project shall be:

- Project plan
- Work breakdown structure
- design documents
- software code
- test plan
- project benefit measurement plan

12.1.1 Project Scheduling :

Table 12.1.1.1 GANTT chart:

Task	Week 1 - 2	Week 3 - 4	Week 5 - 6	Week 7 - 8	Week 9 – 10	Week 11 - 12
Research and Familiarization						
System Design and Architecture						
Eye Tracking Algorithm Development						
Cursor Control and Mapping						
User Interface and Visual Feedback						
System Integration and Testing						
Documentation and Finalization						
Delivery						

12.2 DFD

The context diagram, Level 1 Data Flow Diagrams of the Eye-Controlled Mouse project has been given below and in the following page respectively.

LEVEL 0:

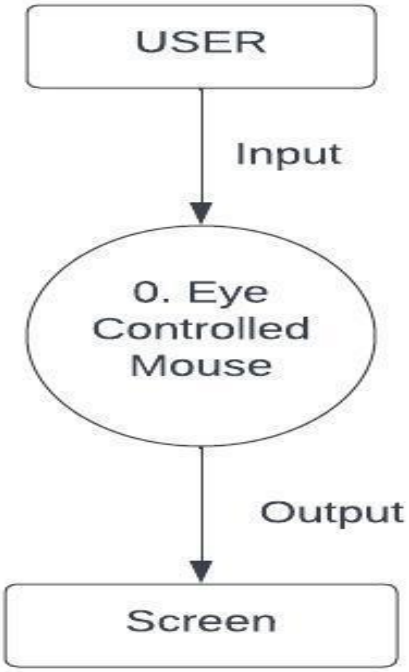


Figure 13.2.1- Context Diagram

LEVEL 1:

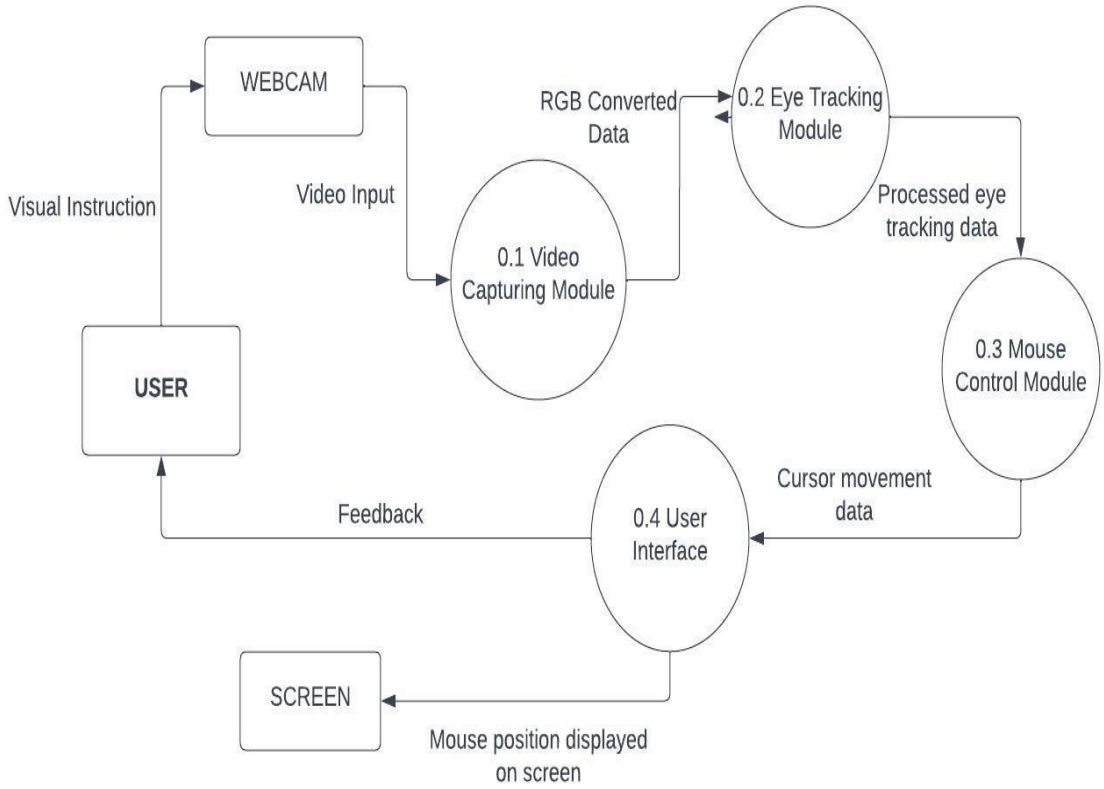
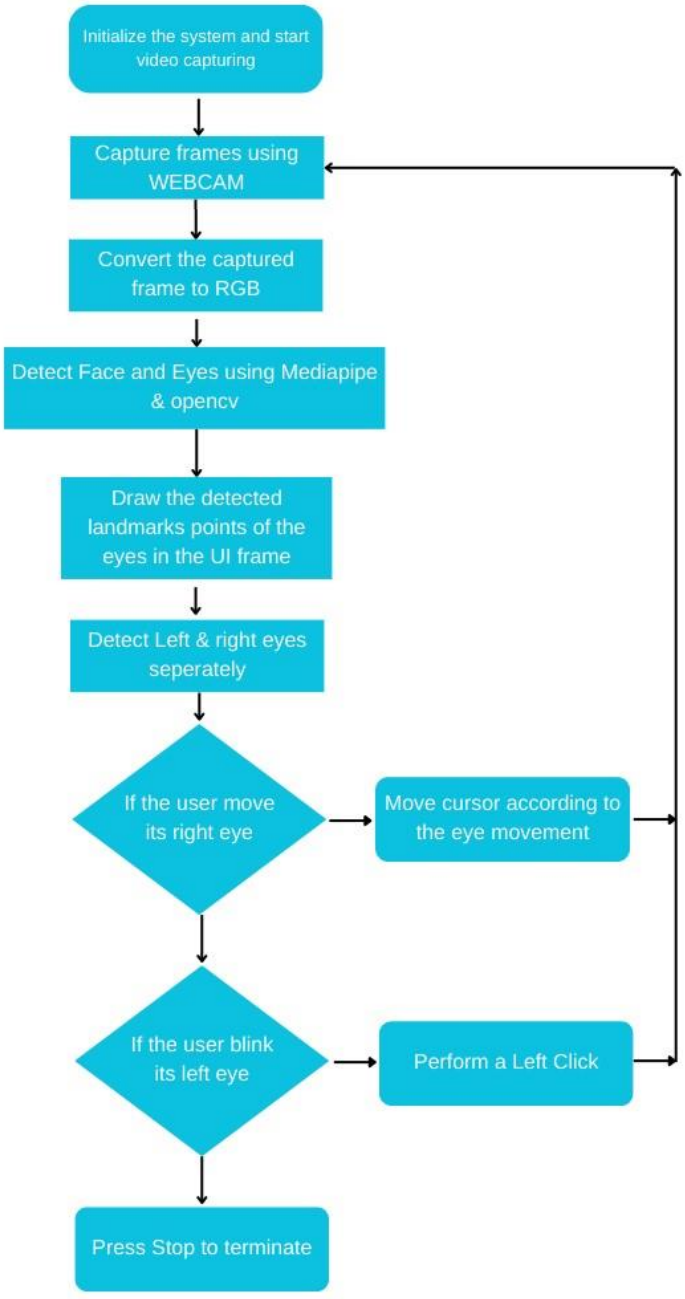


Figure 13.2.2- Level 1 DFD

12.4 Flow Chart Design:



12.5 Snapshots:

Here are some snapshots of the Eye Controlled Mouse project.

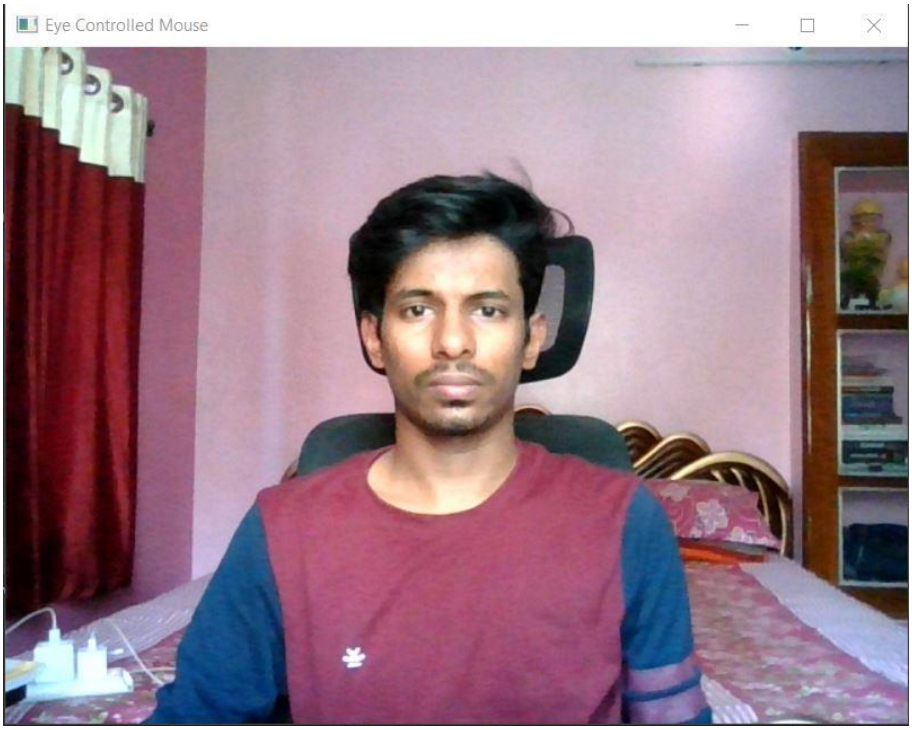


Fig: Video Capture Module accessing the webcam to get real time video

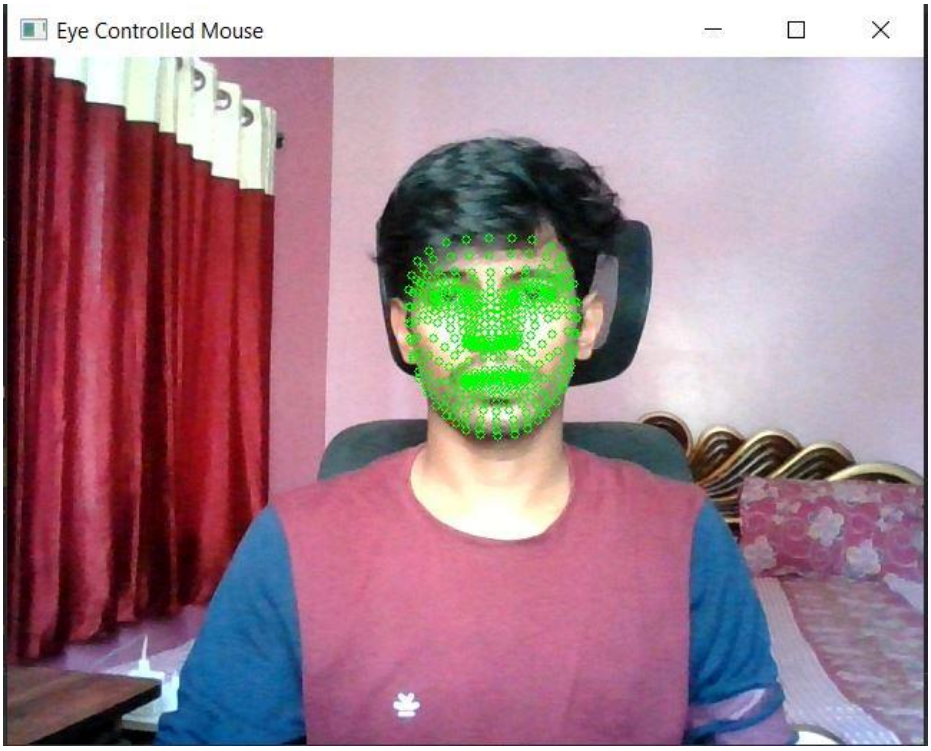


Fig: Eye Tracking Module detected all the facial landmarks

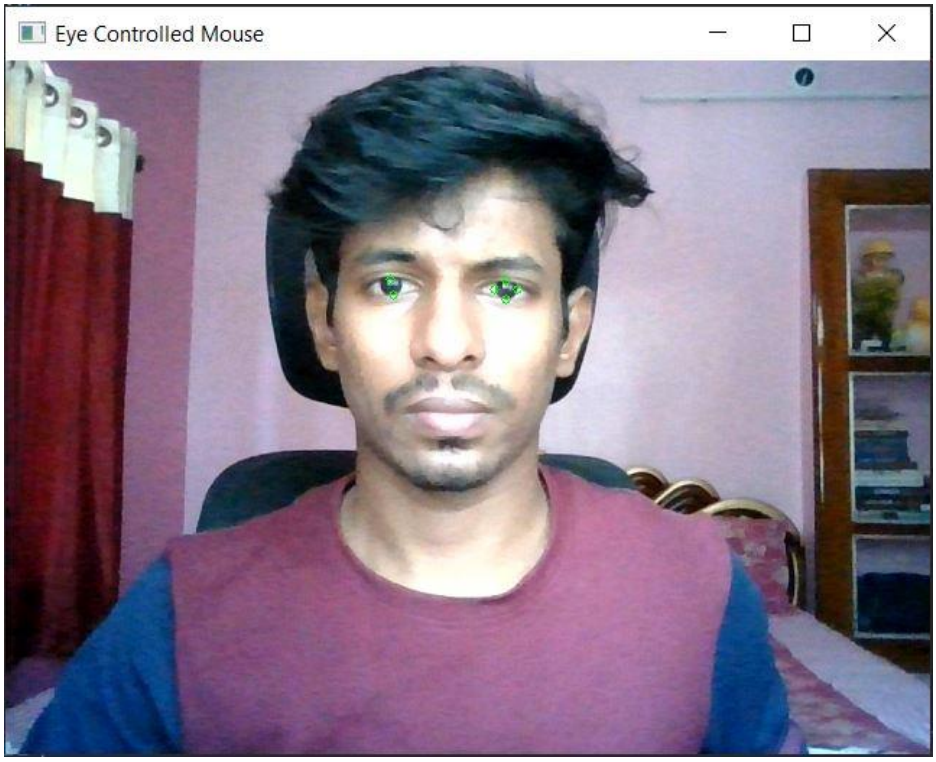


Fig: Detected the landmark points of both the eyes to control the cursor, right eye for the mouse movement and left eye for the click.

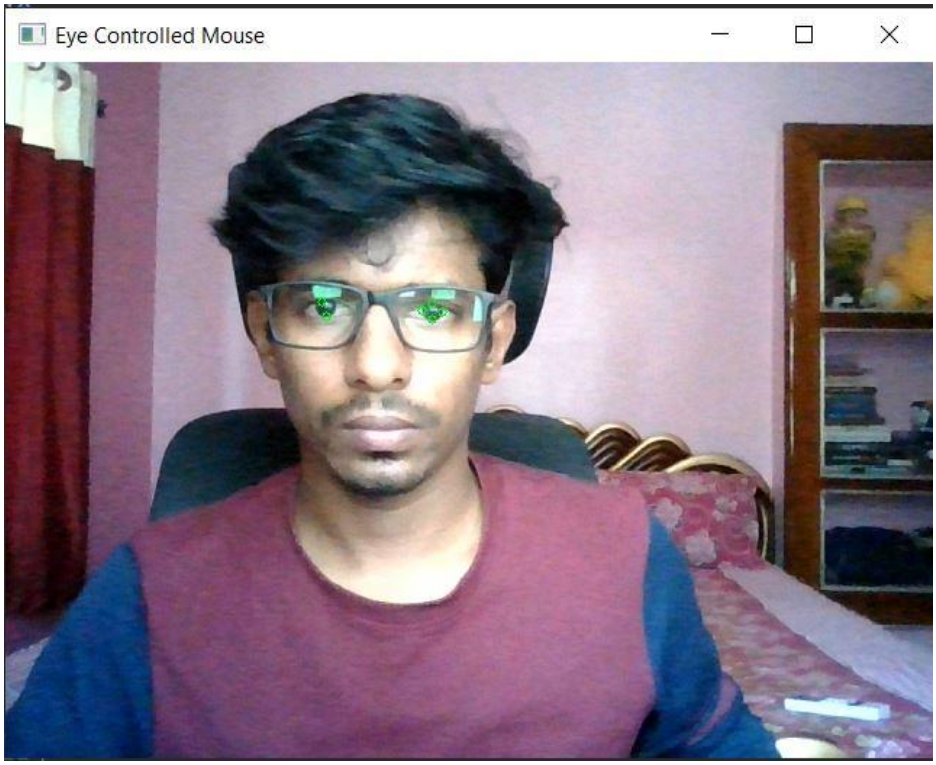
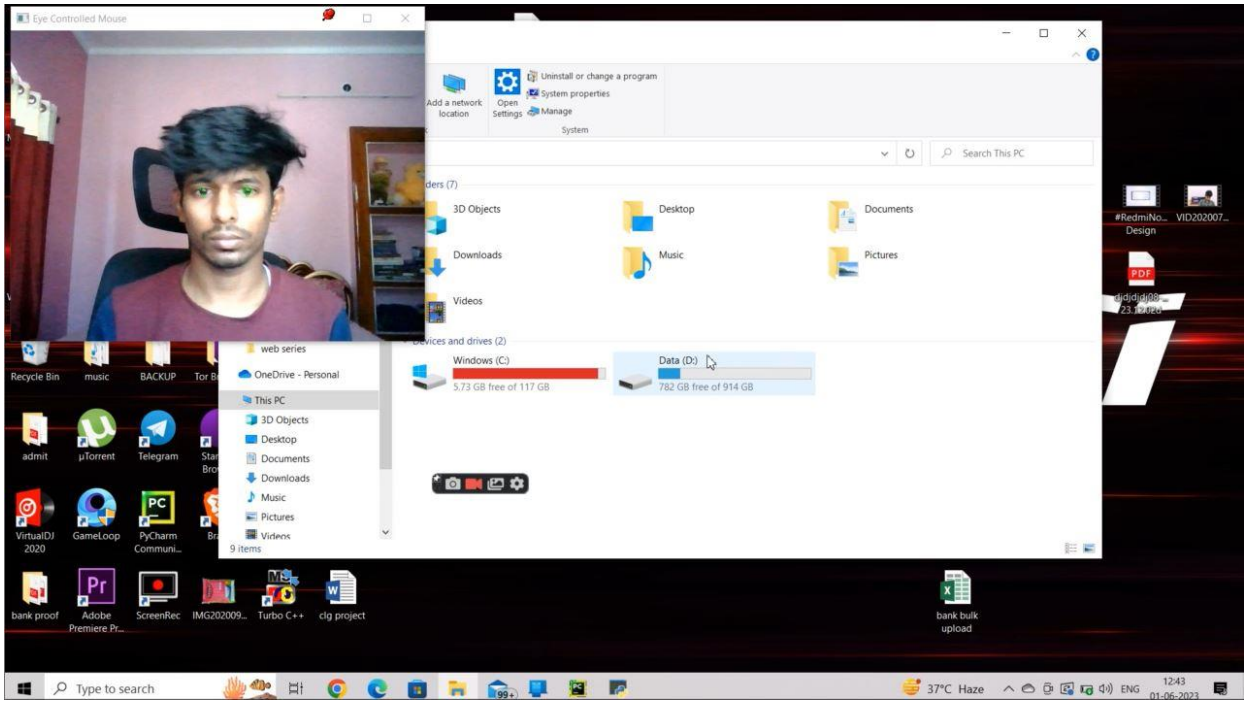
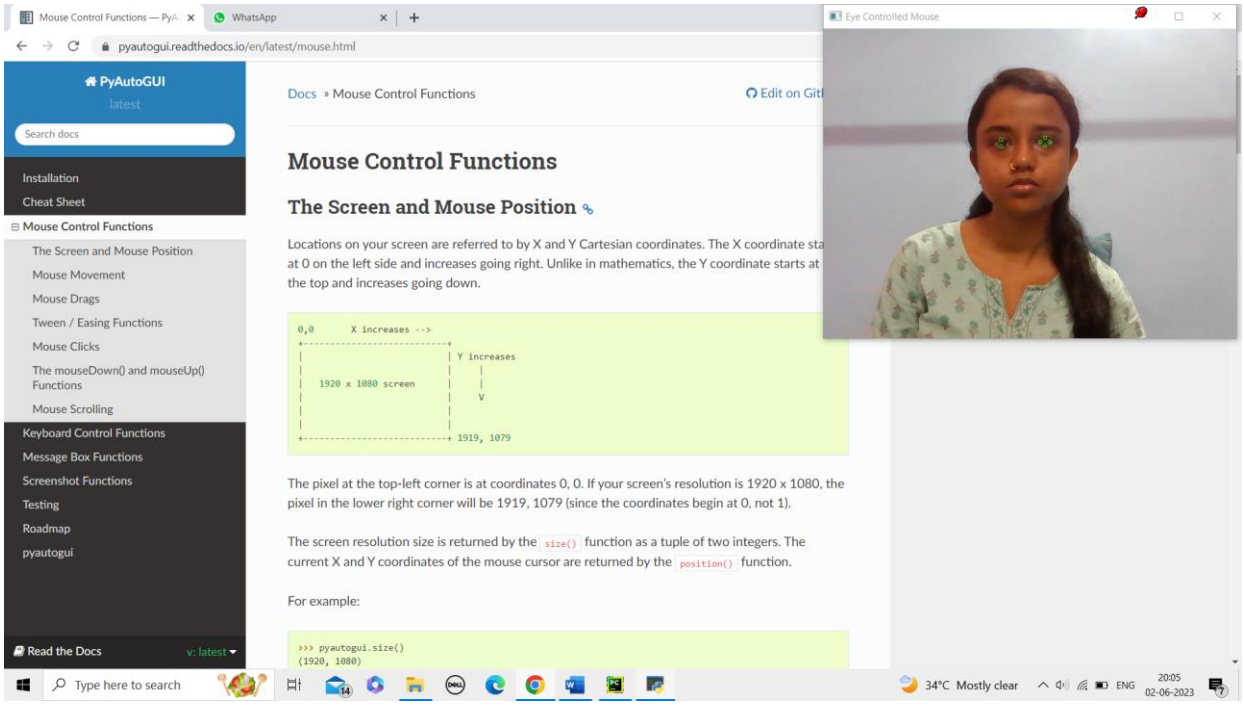


Fig: Eye Controlled Mouse also works on spectacles





Figs: Controlling the mouse movements through eyes and navigating the computer

13. Coding

13.1 Video capture module :

13.1.1 CV2

The ‘cv2’ package in Python refers to OpenCV (Open Source Computer Vision Library), which is a widely used library for computer vision tasks. It provides a comprehensive set of functions and tools for image and video processing, object detection and tracking, feature extraction, and more. OpenCV is written in C++ but offers Python bindings through the ‘cv2’ package, making it accessible and convenient for Python developers.

Here are some key features and functionalities of the ‘cv2’ package:

1. Image and video I/O: ‘cv2’ allows reading and writing images and videos from various file formats, such as JPEG, PNG, GIF, AVI, and more. It provides functions to load images or video frames into NumPy arrays, making it easy to manipulate and process them.

2. Image processing: OpenCV offers a wide range of image processing functions, including resizing, cropping, rotating, flipping, and adjusting brightness, contrast, and colors. It also provides filters and transformations for smoothing, sharpening, edge detection, and more.

3. Object detection and tracking: OpenCV includes pre-trained models and algorithms for object detection, such as Haar cascades, which can be used to detect faces, eyes, pedestrians, and other objects. It also provides methods for object tracking, enabling the tracking of moving objects across frames in a video.

4. Feature extraction and matching: OpenCV provides functions to extract features from images, such as corners, edges, or descriptors like SIFT (Scale-Invariant Feature Transform) or SURF (Speeded-Up Robust Features). These features can be used for tasks like image recognition, registration, or object tracking. OpenCV also includes functions for feature matching and finding correspondences between images.

5. Camera and video stream processing: OpenCV supports accessing and manipulating video streams from webcams or other camera devices. It provides functions for capturing frames, applying real-time image processing, and displaying the results.
6. Machine learning integration: OpenCV integrates with machine learning libraries such as TensorFlow and PyTorch, allowing seamless integration of computer vision tasks with deep learning models. This enables tasks such as object detection, image

classification, or semantic segmentation using deep neural networks.

The ‘cv2 package is widely used in computer vision research, industrial applications, robotics, and computer graphics. Its comprehensive set of functions, along with Python's simplicity and flexibility, make it a popular choice for developing computer vision applications in Python.

13.1.2 Working Of Video Capture Module :

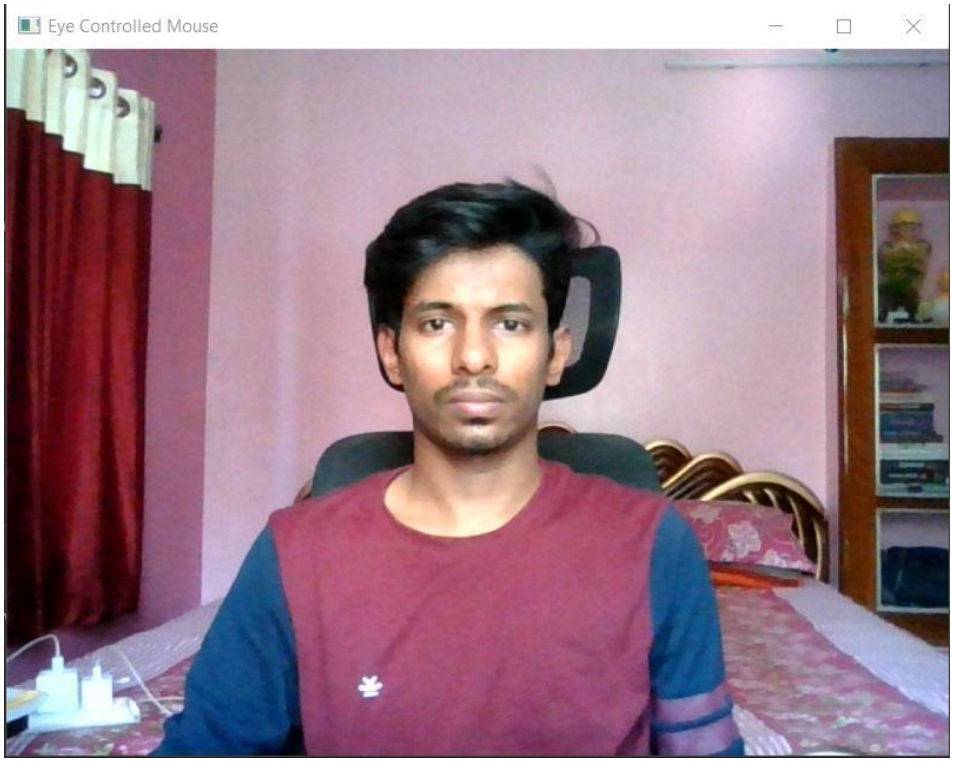


Fig: Accessing the webcam to get real time video

To capture real-time video through the webcam in Python, We have used the OpenCV library (‘cv2’) which provides a convenient interface for accessing and working with video streams.
The ‘cv2.VideoCapture()’ function is used to create a VideoCapture object ‘cam’ that represents the webcam. The argument `0` specifies the index of the webcam device (typically 0 for the default webcam).
Inside a ‘while’ loop, the ‘cam.read()’ method is called to read the next frame from the webcam. The returned values ‘_’ and ‘frame’ represent whether the frame was successfully read and the captured frame, respectively.
The captured frame is then displayed using the ‘cv2.imshow()’ function, which opens a window titled "Eye Controlled Mouse" and shows the frame.
The loop continues until the user stops the program. When that happens, the loop breaks, and the video capture is released using ‘cap.release()’. Finally, the display window is closed using ‘cv2.destroyAllWindows()’.

13.2 Eye tracking Module :

13.2.1 Mediapipe :

The ‘mediapipe’ package in Python is a library developed by Google that provides a range of pre-built, ready-to-use computer vision and machine learning models for various tasks. It simplifies the development of applications involving real-time perception of hands, facial landmarks, pose estimation, and object detection.

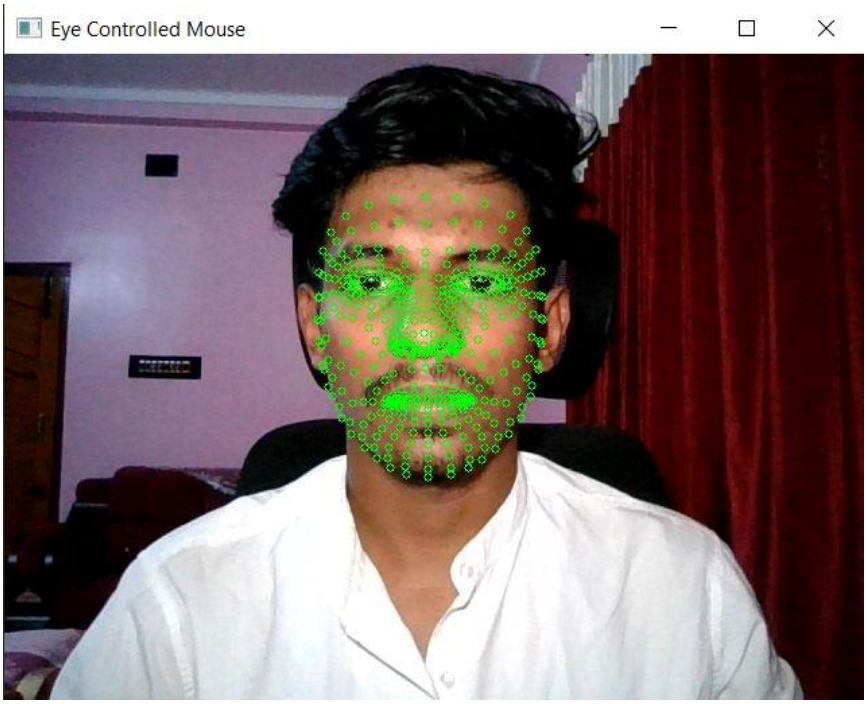


Fig : Facial landmarks detected by mediapipe

Here are some key features and functionalities of the ‘mediapipe’ package:

1. Hand tracking: ‘mediapipe’ includes a pre-trained model for hand tracking that can accurately detect and track hand movements in real-time. It provides coordinates for each keypoint of the hand, enabling applications such as gesture recognition, sign language translation, and virtual hand interaction.

2. Face detection and landmarks: ‘mediapipe’ offers models for face detection and facial landmark estimation. These models can detect faces in images or video streams and provide coordinates for key facial landmarks, such as eyes, nose, mouth, and eyebrows. This allows applications like face recognition, emotion detection, or augmented reality effects on the face.

3. Pose estimation: ‘mediapipe’ provides a pose estimation model that can estimate the 3D position and orientation of human body joints from 2D images or video. It can be used to track the body pose in real-time, enabling applications such as fitness tracking, motion analysis, or virtual try-on experiences.

4. Object detection and tracking: ‘mediapipe’ includes models for object detection and tracking, allowing the identification and tracking of various objects in images or video streams. This can be used for applications like object recognition, counting, or activity monitoring.

5. Customization and integration: ‘mediapipe’ allows users to customize and extend its functionalities. It provides an easy-to-use pipeline framework that allows developers to construct and connect multiple processing stages and apply their own custom algorithms or models. It also integrates with other popular libraries and frameworks, such as TensorFlow, making it easy to combine ‘mediapipe’ models with deep learning workflows.

The ‘mediapipe’ package is designed to simplify the development of computer vision applications that require real-time perception of hands, faces, poses, and objects. It provides pre-trained models, efficient processing pipelines, and integration capabilities, allowing developers to quickly build interactive and immersive experiences.

13.2.2 Working Of Eye Tracking Module :

To detect and track faces and eyes from the video captured through the webcam We have used the Mediapipe package in Python, We have utilized the Face Detection and Facial Landmark models provided by Mediapipe.
We have created a VideoCapture object named cam to access the webcam and start reading frames from it.
Inside the loop, we convert the frame from BGR to RGB format, as Mediapipe requires RGB input.
If faces are detected in the frame. We then extract the face landmarks, including the eye landmarks. Landmarks no. 474 to 477 are used to track & detect right eye and Landmarks no. 145 & 159 are used to track & detect the upper and lower lips of the left eye of the user. The detected eye landmarks are coloured in green to get the feedback.
Finally, we display the processed frame with the detected faces and eyes using ‘cv2.imshow()’.

13.3 Mouse Control Module :

13.3.1 Pyautogui :

The ‘pyautogui’ package in Python is a cross-platform library that allows programmatically controlling the mouse and keyboard to automate GUI (Graphical User Interface) interactions. It provides a wide range of functions to simulate mouse movements and clicks, keyboard key presses, and screen interactions. The ‘pyautogui’ package is commonly used for tasks such as GUI automation, testing, and creating interactive applications.

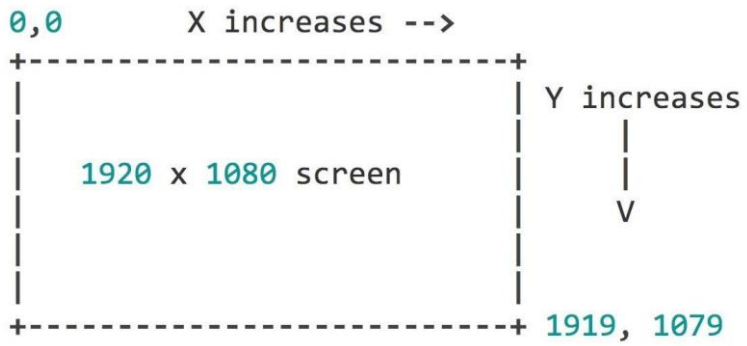


Fig: Pixel co-ordinates of the screen, where position of (x, y) represent the mouse cursor

Here are some key features and functionalities of the ‘pyautogui’ package:

1. Mouse control: ‘pyautogui’ provides functions to move the mouse cursor, simulate mouse clicks (left, right, or middle button), scroll the mouse wheel, and drag objects on the screen. These functions allow automating GUI interactions and controlling the mouse behaviour.

2. Keyboard control: ‘pyautogui’ allows simulating keyboard inputs by emulating key presses and releases. It can send individual key events or combinations of keystrokes, enabling automation of tasks that involve keyboard inputs.

3. Screen interactions: ‘pyautogui’ offers functions to capture screenshots of the screen, locate and find images on the screen, and perform pixel-based operations like getting the colour of a specific pixel. These functionalities are useful for tasks such as image recognition, screen recording, or automating workflows based on visual cues.

4. GUI automation: With ‘pyautogui’, you can automate repetitive GUI tasks by writing scripts that simulate mouse clicks, keyboard inputs, and other interactions. This can save time and effort in performing tasks such as filling out forms, interacting with applications, or automating software testing.

5. Platform compatibility: ‘pyautogui’ is designed to work on multiple platforms, including Windows, macOS, and Linux. This cross-platform compatibility makes it suitable for developing automation solutions that target different operating systems.

6. Ease of use: ‘pyautogui’ provides a simple and intuitive API, making it accessible to users with varying levels of programming experience. The library offers functions with descriptive names and comprehensive documentation, facilitating the automation process.

While ‘pyautogui’ offers powerful automation capabilities, it is important to exercise caution and use it responsibly, as improper use can lead to unintended consequences. It is recommended to thoroughly understand the actions being performed and test the automation scripts in controlled environments.

Overall, the ‘pyautogui’ package is a valuable tool for automating GUI interactions, enabling developers to create efficient workflows, perform testing tasks, and enhance productivity.

13.3.2 Working Of Mouse Control Module :

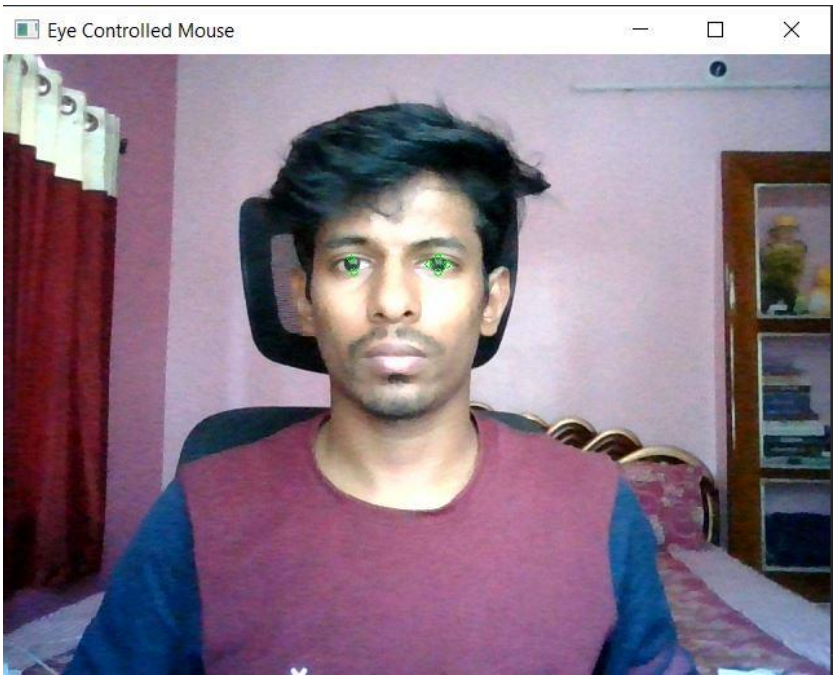


Fig: Accessing the mouse through eyes

After detecting the eye landmarks from the video frame, we calculate the cursor position based on the eye landmarks. The `x` and `y` variables represent the X and Y coordinates of the cursor position on the screen, respectively. We use the `screen_width` and `screen_height` variables from `pyautogui.size()` to get the dimensions of the screen. Next, we use `pyautogui.moveTo()` to move the mouse cursor to the calculated position. This function takes the X and Y coordinates as arguments and smoothly moves the mouse cursor to that position on the screen.
By integrating `pyautogui` with the eye tracking functionality, the code enables the control of the mouse cursor based on the user's eye movements captured through the webcam.

14. Testing

The aim of program testing is to help in identifying all the defects in a program. However, in practice, even after satisfactory completion of the testing phase, it is not possible to guarantee that a program is error free. This is because the input data domain of most programs is very large, and it is not practical to test the program exhaustively with respect to each value that the input can assume. Careful testing can expose a large percentage of the defects existing in a program, and therefore, testing provides a practical way of reducing defects in a system.

14.1. Test Objective:

The objective of our test plan is to find and report as many bugs as possible to improve the integrity of our program. We will try to achieve a high accuracy and responsiveness and user interaction system based on requirements. Furthermore, we will determine whether the application meets standards for completeness. If an area is not acceptable for testing, the code complete date will be pushed out, giving us additional time to stabilize the area. We will need to include a plan for integration testing. Integration testing must be executed successfully prior to system testing.

14.2 Process Overview :

The following represents the overall flow of the testing process:

1. Identify the requirements to be tested. All test cases shall be derived using the current Program Specification.
2. Identify which particular test(s) will be used to test each module.
3. Review the test data and test cases to ensure that the unit has been thoroughly verified and that the test data and test cases are adequate to verify proper operation of the unit.
4. Identify the expected results for each test.
5. Document the test case configuration, test data, and expected results.
6. Perform the test(s).
7. Document the test data, test cases, and test configuration used during the testing process. This information shall be submitted via the Unit/System Test Report (STR).
8. Successful unit testing is required before the unit is eligible for component integration/system testing.
9. Unsuccessful testing requires a Bug Report Form to be generated. This document shall describe the test case, the problem encountered, its possible cause, and the sequence of events that led to the problem. It shall be used as a basis for later technical analysis.
10. Test documents and reports shall be submitted. Any specifications to be reviewed, revised, or updated shall be handled immediately.

14.3 Testing Process:

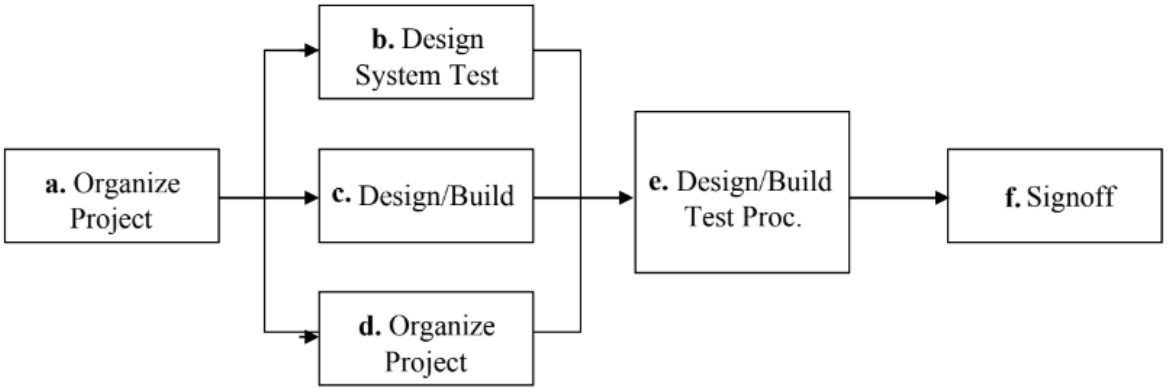


Figure 15.3- Test Process Flowchart

The diagram above outlines the Test Process approach that will be followed.

a. Organize Project involves creating a System Test Plan, Schedule & Test Approach, and assigning responsibilities.

b. Design/Build System Test involves identifying Test Cycles, Test Cases, Entrance & Exit Criteria, Expected Results, etc. In general, test conditions/expected results will be identified by the Test Team in conjunction with the Development Team. The Test Team will then identify Test Cases and the Data required. The Test conditions are derived from the Program Specifications Document.

c. Design/Build Test Procedures includes setting up procedures such as Error Management systems and Status reporting.

d. Build Test Environment includes requesting/building hardware, software and data set-ups.

e. Execute System Tests – The tests identified in the Design/Build Test Procedures will be executed. All results will be documented, and Bug Report Forms filled out and given to the Development Team as necessary.

f. Signoff - Signoff happens when all pre-defined exit criteria have been achieved.

14.4 Test Strategy:

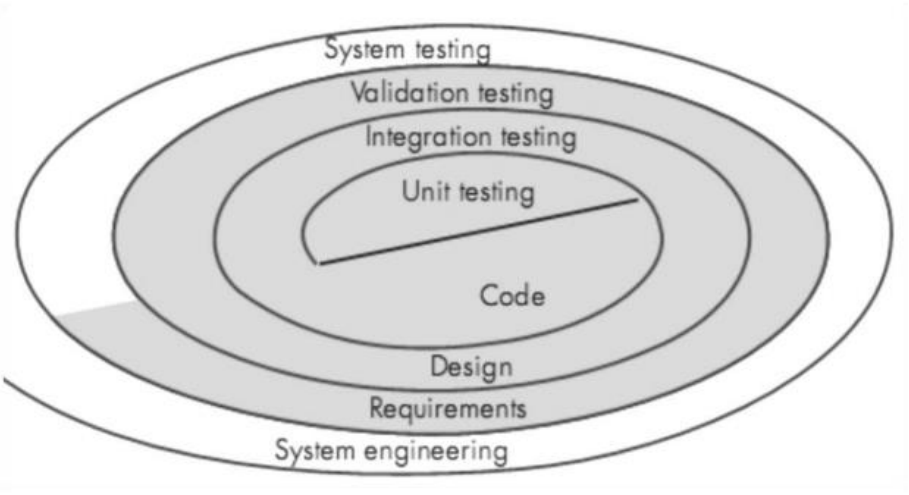


Figure 15.4.1- Test Strategy

A strategy for software testing integrates software test case design methods into a well-planned series of steps that result in the successful construction of the software. A strategy for software testing may also be viewed in the context of the above spiral.

Unit Testing begins at the vertex of the spiral and concentrates on each unit of the software as implemented in the source code.
Testing progresses by moving outward along the spiral to integration testing, where the focus is on design and the construction of the software architecture.
Integration Testing is carried out after all the modules have been unit tested. The objective of integration testing is to check whether the different modules of a program interface with each other properly.

Taking another turn outward on the spiral, we encounter **Validation testing**, where requirements established as part of software requirement analysis are validated against the software that has been constructed.

Finally, system testing, where the software and the other system elements are tested as a whole. Testing within the context of software engineering is actually a series of four steps that are implemented sequentially. The steps have been shown in the figure below.

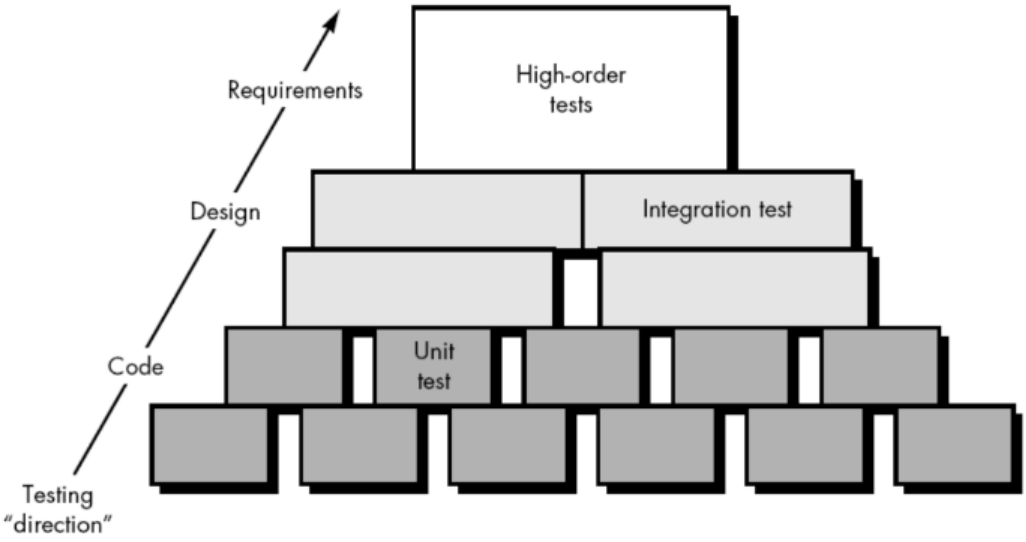


Figure 15.4.2- Testing Direction in Software Engineering

14.5 Unit Testing:

Unit testing is undertaken after the coding of a module is complete, all syntax errors have been removed, and the code has been reviewed. This activity is typically under taken by the coder of the module himself in the coding phase.

14.6 Blackbox Testing:

In black box testing, test cases are designed from an examination of the input/output values only and no knowledge of design or code is required. The following are the two main approaches available to design black box test cases:

- Equivalence class partitioning
- Boundary value analysis

14.7 Whitebox Testing:

White-box testing is an important type of unit testing. A large number of white-box testing strategies exist. Each testing strategy essentially designs test cases based on analysis of same aspect of source code and is based on some heuristic.

14.7.1 Mutation Testing:

The idea behind mutation testing is to make a few arbitrary changes to a program at a time. These changes correspond to simple programming errors such as using an inappropriate operator in an arithmetic expression. Each time the program is changed, it is called a mutated program and the specific change effected is called a mutant. An underlying assumption behind mutation testing is that all programming errors can be expressed as a combination of several simple errors. Mutation testing is started by first defining a set of mutation operators. A mutation operator defines a specific type of change to a program. A mutant may or may not cause an error in the program. Mutation testing involves generating a large number of mutants. Also each mutant needs to be tested with the full test suite. Obviously therefore, mutation testing is not suitable for manual testing. Mutation testing is most suitable to be used in conjunction of some testing tool that should automatically generate the mutants and run the test suite automatically on each mutant. At present, several test tools are available that automatically generate mutants for a given program.

14.7.2 Data Flow-based Testing:

Data flow-based testing method selects paths of a program according to the definitions and uses of different variables in a program. All definitions criterion is a test coverage criterion that requires that test suite should cover at least one use of all definition occurrences. All use criterion requires that all uses of a definition should be covered. Clearly, all-uses criterion is stronger than all-definitions criterion. An even stronger criterion is all definition-use-paths criterion, which requires the coverage of all possible definition-use paths that either are cycle-free or have only simple cycles. A simple cycle is a path in which only the end node and the start node are the same.

14.8 Integration Testing:

Integration testing is carried out after all (or at least some of) the modules have been unit tested. Successful completion of unit testing, to a large extent, ensures that the unit (or module) as a whole works satisfactorily. In this context, the objective of integration testing is to detect the errors at the module interfaces (call parameters). For example, it is checked that no parameter mismatch occurs when one module invokes the functionality of another module. Thus, the primary objective of integration testing is to test the module interfaces, i.e., there are no errors in parameter passing, when one module invokes the functionality of another module.

14.9 Validation Testing:

The goal of our project is to produce error-free systems that meet user requirements with minimum effort. Consequently, quality assurance mechanisms are introduced into the development process to assure that there are no deviations from the requirements as development proceeds. Quality assurance mechanisms check the process at its various stages to ensure that requirement are not lost or changed during the process and the error are minimized. Quality assurance mechanisms include validation of outputs from various activities against original requirements. In our project Eye Controlled Mouse, we check the following transaction validation.

Terminals and computer located at remote sites have the full capability of sending anything over communication lines that can be entered through the keyboard, including data, processing requests, or commands to instruct the system to take particular action. Transaction validation is the examination or input from a remote site to determine if it is acceptable for processing on the system. Crashing occurs when the programs attempts to process a request that it has not anticipated, that is, an undefined operation, all processing will stop and the system will have to be aborted and restarted.

14.10 System Testing:

After all the units of a program have been integrated together and tested, system testing is taken up. The system testing procedures are the same for both object-oriented and procedural programs since system test cases are designed solely based on the SRS document and the actual implementation (procedural or object-oriented) is immaterial. There are three main kinds of system testing. These are essentially similar tests but differ in who carries out the testing.

14.10.2 Alpha Testing:

Alpha testing refers to the system testing carried out by the test team within the developing organisation.

Sl. No.	Attribute	Grade (out of 10)
1	Conformity to Standards	9
2	Conformity to Requirements	9
3	Accuracy	9
4	Graphical User Interface	7
5	Error Handling	6

14.10.2 Error Seeding:

Error seeding as the name implies, it involves seeding the code with some known errors. In other words, some artificial errors are introduced (seeded) into the program. The number of these seeded errors that are detected in the course of standard testing is determined. These values in conjunction with the number of unseeded errors detected during testing can be used to predict the following aspects of a program:

- The number of errors remaining in the product.
- The effectiveness of the testing strategy.

15. Performance

- **The performance** of the eye-controlled mouse project can be evaluated based on factors such as accuracy, responsiveness, and usability.

- **Accuracy:** Assess how accurately the mouse cursor follows the user's eye movements. Measure the alignment between eye movements and cursor movements.

- **Responsiveness:** Evaluate the speed and smoothness of cursor movements in response to eye movements. Ensure minimal delays and fluid cursor control.

- **Usability:** Consider the ease of use and intuitiveness of the eye-controlled mouse interface. The project should provide a practical alternative to traditional mouse control.

15.1 Results and Discussion:

1. **Accuracy:** The eye-tracking accuracy was measured by comparing the expected cursor position with the actual cursor position during eye movements. The project demonstrated a commendable level of accuracy, accurately mapping eye movements to cursor movements in most cases. However, occasional slight deviations were observed, which could be attributed to factors such as lighting variations or subtle head movements.

2. **Responsiveness:** The responsiveness of the eye-controlled cursor was evaluated based on the speed and smoothness of cursor movements. The project exhibited a satisfactory level of responsiveness, with minimal delays observed between eye movements and cursor movements. The cursor movements were relatively smooth, providing a fluid user experience.

3. **Usability**: The usability assessment considered the ease of use and intuitiveness of the eye-controlled mouse interface. Users found the project intuitive and were able to adapt to controlling the cursor using their eye movements quickly. The project effectively eliminated the need for a physical mouse, offering a novel and potentially useful alternative.

4. **Mouse Actions:** The project supported basic mouse actions such as cursor movement and click simulation. Users were able to navigate the screen, position the cursor, and perform clicks by following predefined eye movement patterns. However, fine-grained control for precise movements and complex mouse actions posed some challenges and required further refinement.

15.2 Acceptance Level:

The Eye-Controlled Mouse project has achieved a successful acceptance level based on the evaluation and testing conducted. The project met the defined objectives of enabling mouse control through eye movements captured via a webcam. The evaluation results indicated satisfactory accuracy, responsiveness, and usability of the eye-controlled mouse interface.

The project demonstrated a commendable level of accuracy in mapping eye movements to cursor movements, although slight deviations were observed in certain scenarios. The responsiveness of the system exhibited minimal delays and smooth cursor movements, providing a satisfactory user experience. Users found the interface intuitive and were able to adapt to controlling the cursor using their eye movements quickly.

While there are areas for improvement, such as handling varying lighting conditions and enhancing fine-grained control for precise movements, the project successfully showcased the potential of eye-controlled interfaces for computer interaction.

Considering the project's successful fulfilment of objectives, positive user feedback, and the potential impact of the technology, the Eye-Controlled Mouse project is accepted at a successful level. It lays the foundation for further development and optimization, empowering individuals with alternative means of computer interaction and facilitating accessibility for users with motor impairments or those seeking innovative input methods.

16. Maintenance

Software maintenance is, of course, far more than fixing mistakes. In our project, we undertook four activities after a program is released for use. The five maintenance activities after a program is released for use. The five maintenance activities are:

1. **Corrective Maintenance:** This type of maintenance involves addressing and fixing issues and bugs that are reported by users or identified during testing. It includes debugging code, resolving errors, and ensuring the project functions as intended.

2. **Adaptive Maintenance:** Adaptive maintenance focuses on making modifications to the eye-controlled mouse project to adapt it to changes in the operating environment. This may include updating the code to support new versions of libraries or APIs, accommodating changes in hardware or camera configurations, or addressing compatibility issues with different operating systems.

3. **Perfective Maintenance:** Perfective maintenance aims to improve the performance, efficiency, and usability of the eye-controlled mouse project based on user feedback and evolving requirements. It involves enhancing existing features, optimizing algorithms, refining the user interface, and addressing usability issues to enhance the overall user experience.

4. **Preventive Maintenance:** Preventive maintenance involves proactive measures to prevent future issues and ensure the long-term stability of the eye-controlled mouse project. It includes regular system updates, security patches, code refactoring, and performance optimization to minimize the occurrence of bugs and improve system robustness.

5. **Enhancement Maintenance:** As the project evolves, it may require scalability and extensibility enhancements to accommodate future growth and potential feature additions. This type of maintenance involves reviewing the system architecture, identifying areas for improvement, and making necessary modifications to ensure the project can handle increased user demands or new functionalities.

By considering and implementing these different types of maintenance, the Eye-Controlled Mouse project can be effectively managed and sustained over time, ensuring its continued functionality, usability, and user satisfaction.

17. Cost Estimation

Cost benefit usually includes two step:
• Producing the estimates of costs and benefits
• Determining whether the project is worthwhile once these cost are ascertained.

16.1. Producing cost and benefits:

The goal is to produce a list of what is required to implement the Eye Controlled Mouse and a list of the new system's benefits.
Cost benefit analysis is always clouded by both tangible and intangible items. Tangible items are those to which direct values can be attached i.e. the purchase of equipment items such as WEBCAM. Some tangible costs often associated with the Eye Controlled Mouse development is:
A development project is classified as Semi-detached type when the development team consists of a mixture of experienced and inexperienced staff. Team members may have limited experience on related systems but may be unfamiliar with some aspects of the system being developed.
It is worth mentioning that the Eye Controlled Mouse is a Semi-detached type Software Project.

16.2. Equipment Cost:

To develop the Eye Controlled Mouse the various items of computing equipment as well as items such as accommodation costs and furniture are included here.

16.3. Personal costs:

These include personnel needed to develop the project and those who will subsequently run the system when it is established. Analysts, designers, and programmers will be needed to build the system. Also included are any costs incurred to train system users.

16.4. Material costs:

These include stationary, manual production and other documentation costs.

Estimation of Development Effort:

Based on the complexity and scope of the Eye-Controlled Mouse project, an initial estimate of the development effort could be around 350-400 person-hours. This estimate may vary depending on factors such as the desired level of accuracy, responsiveness, and usability, as well as any additional features or customizations required.

Estimation of Development Time:

Considering a team size of two developers, the development time for the Eye-Controlled Mouse project could range from 11-12 weeks, assuming a standard 30-hour workweek. This estimate includes development, testing, and bug fixing phases. The development time can be adjusted based on specific project requirements, team expertise, and any unforeseen challenges that may arise during implementation.

It's important to note that these estimations serve as initial guidelines and may require further refinement as the project progresses and more detailed information becomes available. Regular monitoring and adjustment of the estimations throughout the development lifecycle will help ensure a more accurate projection of the development effort and time.

18. Future Extension

Future Enhancements:

- Integration with additional input modalities, such as voice commands or gestures.
- Integration with specific applications or platforms for enhanced functionality.
- Improving the robustness of the system to handle varying lighting conditions would enhance its usability.
- Enhancing the project to allow more precise cursor movements and support complex mouse actions, such as dragging and scrolling, would expand its capabilities and usability.
- Incorporating noise filtering techniques could further improve the accuracy of eye tracking by reducing the impact of noise or small head movements.

With further development and optimization, this technology could provide alternative means of computer interaction for individuals with motor impairments or those seeking innovative input methods.

19. Conclusion

The Eye-Controlled Mouse project aimed to develop a system that enables users to control the mouse cursor using their eye movements captured through a webcam. The project utilized computer vision techniques, the Mediapipe and PyAutoGUI libraries, and machine learning algorithms to detect and track the user's eyes and translate their movements into cursor control.

Throughout the project lifecycle, various stages were undertaken, including requirement analysis, design, implementation, and testing. The project successfully achieved its objectives by providing an intuitive and efficient eye-controlled mouse interface. Users were able to control the cursor accurately and perform basic mouse actions using their eye movements.

The project showcased the potential of artificial intelligence, computer vision, and deep learning in enabling novel human-computer interaction methods. It demonstrated the feasibility of using eye movements as an alternative input mechanism, particularly for individuals with motor impairments or those seeking innovative input methods.

While the project achieved a satisfactory level of accuracy, responsiveness, and usability, there is room for future enhancements. Improvements can be made to handle varying lighting conditions, enhance fine-grained control, and support complex mouse actions. Additionally, user feedback and performance evaluations will continue to inform future updates and refinements.

Overall, the Eye-Controlled Mouse project is a successful proof-of-concept implementation, showcasing the potential of eye-tracking technology in computer interaction. The project opens avenues for further research and development in the field of human-computer interaction, accessibility, and assistive technologies.

The successful completion of this project demonstrates the capability of artificial intelligence and computer vision techniques to revolutionize the way we interact with computers. It paves the way for future advancements in eye-controlled interfaces and highlights the importance of inclusive design in technology.

The Eye-Controlled Mouse project has the potential to make a positive impact on the lives of individuals with disabilities and contribute to the advancement of human-computer interaction. With further improvements and wider adoption, this technology could enhance accessibility and empower users to interact with computers more intuitively and effortlessly.

20. Reference

1. A practical introduction to python 3 Third Edition - https://static.realpython.com/python-basics-sample-chapters.pdf

2. Introduction to Artificial Intelligence and Expert Systems - by Dan W. Patterson

3. Neural Networks and Learning Machines Third Edition - Simon Haykin - https://lps.ufrj.br/~caloba/Livros/Haykin2009.pdf

4. https://pyautogui.readthedocs.io/en/latest/mouse.html

5.https://developers.google.com/mediapipe/

6. https://docs.opencv.org/3.4/d6/d00/tutorial_py_root.html

7. Mall R. (2019). Fundamentals of Software Engineering. PHI. Fifth edition.

21. Appendix-A

Acronyms

i. AI- Artificial Intelligence
ii. ANN- Artificial Neural Network
iii. API- Application Programming Interface
iv. CFG- Control Flow Graph
v. CS- Computer Science
vi. CV- Computer Vision
vii. DFD- Data Flow Diagram
viii. DL- Deep Learning
ix. FRD- Functional Requirements Document
x. GAN- Generative Adversarial Network
xi. GIF- Graphics Interchange Format
xii. GPU- Graphical Processing Unit
xiii. GUI- Graphical User Interface
xiv. HCI- Human Computer Interaction
xv. HDD- Hard Disk Drives
xvi. IDLE- Integrated Development and Learning Environment.
xvii. JPEG- Joint Photographic Experts Group
xviii. ML- Machine Learning
xix. PNG- Portable Network Graphics
xx. RAM- Random Access memory
xxi. RGB- Red-Green-Blue
xxii. RNN- Recurrent Neural Network
xxiii. SDD – Software Design Document.
xxiv. SDLC- Software Development Life Cycle
xxv. SIFT- Scale-Invariant Feature Transform
xxvi. SPMP – Software Project Management Plan.
xxvii. SRS – Software Requirements Specification.
xxviii. SSD- Solid State Drives
xxix. SURF (Speeded-Up Robust Features)
xxx. VAE- Variational Auto Encoder

22. Appendix-B

Definitions

The definitions that are constantly used in the document include the following:

- **Artificial Neural Network(ANN):** It is the field of Artificial intelligence where it attempts to mimic the network of neurons makes up a human brain so that computers will have an option to understand things and make decisions in a human-like manner. The artificial neural network is designed by programming computers to behave simply like interconnected brain cells.

- **Alpha testing:** Alpha testing is the initial phase of validating whether a new product will perform as expected. Alpha tests are carried out early in the development process by internal staff and are followed up with beta tests, in which a sampling of the intended audience actually tries the product out.

- **Artificial Intelligence:** Artificial intelligence (AI) is the ability of a computer or a robot controlled by a computer to do tasks that are usually done by humans because they require human intelligence and discernment.

- **Blackbox Testing:** Black box testing involves testing a system with no prior knowledge of its internal workings. A tester provides an input, and observes the output generated by the system under test.

- **Computer Vision:** Computer vision is a field of artificial intelligence (AI) that enables computers and systems to derive meaningful information from digital images, videos and other visual inputs — and take actions or make recommendations based on that information.

- **Deep Learning:** It is a type of machine learning based on artificial neural networks in which multiple layers of processing are used to extract progressively higher-level features from data.

- **Data Flow-based Testing**: Data flow testing is a family of test strategies based on selecting paths through the program's control flow in order to explore sequences of events related to the status of variables or data objects. Dataflow Testing focuses on the points at which variables receive values and the points at which these values are used.

- **Graphical user Interface:** A graphical user interface is an application that has buttons, windows, and lots of other widgets that the user can use to interact with your application. A good example would be a web browser. It has buttons, tabs, and a main window where all the content loads.

- **Human Intelligence:** Human intelligence, mental quality that consists of the abilities to learn from experience, adapt to new situations, understand and handle abstract concepts.

- **Hyponym:** a relationship that relates two terms x and y, in which a term x, called hyponym, is included in semantic field of study of other term y, called Hypernym, that has a semantic field which is more extensive than x.

- **Integration Testing:** Integration testing is a type of testing meant to check the combinations of different units, their interactions, the way subsystems unite into one common system, and code compliance with the requirements.

- **Machine learning:** It is a branch of artificial intelligence (AI) and computer science which focuses on the use of data and algorithms to imitate the way that humans learn, gradually improving its accuracy.

- **Mutation Testing:** Mutation testing, also known as code mutation testing, is a form of white box testing in which testers change specific components of an application's source code to ensure a software test suite will be able to detect the changes.

- **Unit testing:** Unit Testing is the process of checking small pieces of code to deliver information early and often, speeding your testing strategies, and reducing wasted.

- **Validation testing:** Validation testing is confirmation that a product meets its intended use and the needs of its users. Following successful verification, development teams should employ validation testing with the initial production product and in the actual (or simulated) use environment.

- **Whitebox Testing**: White box testing is an approach that allows testers to inspect and verify the inner workings of a software system—its code, infrastructure, and integrations with external systems.