Graduation Project Report

for

<EASIEST>

Prepared by
Barış Aydınay-2452977
Furkan Duman-2453173
Kerem Keptiğ-2453355
Daniil Belousov-2491827

Middle East Technical University Northern Cyprus Campus

Computer Engineering Department

Supervised by Assist. Prof. Dr. Sükrü Eraslan
Co-supervised by Prof. Dr. Yeliz Yesilada
07/06/2024

Contents

List of Figures

1 Introduction

1.1 Purpose

The traditional process of diagnosing Autism Spectrum Disorder (ASD) in children relies heavily on timeconsuming methods due to several factors. Initially, healthcare professionals typically conduct comprehensive interviews and assessments with the individual to gather detailed behavioral and developmental history[1]. These assessments often involve observing and interacting with the individual to evaluate social communication skills, repetitive behaviors, and sensory sensitivities. These labor-intensive methods are aimed at ensuring a thorough and accurate evaluation, but they inherently require significant time and resources to complete. In addition to clinical assessments, methods such as MRI scans have also been used for detection[2], but these can be expensive and time-consuming[3]. The high cost of MRI scans is due to several factors, including the expensive machinery, maintenance costs, and the need for specialized personnel to operate the equipment and interpret the results. Recently, the integration of eye-tracking technology with machine learning (ML) has emerged as a more objective and cost-effective method[4]. However, the use of commercial eye-tracking devices remains prohibitively expensive, as noted in various resources[5].

In response to these challenges, our innovative project introduces a web application named EASIEST. This application harnesses gaze-tracking technology to streamline the detection of autism in adults. The primary aim is to enhance the quality of life for individuals with autism and their families by offering a user-friendly and easily accessible tool for the screening process. Unlike traditional diagnostic methods, EASIEST seeks to significantly reduce both the time and cost associated with the initial screening for Autism Spectrum Disorder. The web application achieves this by collecting data on eye movements and behavior using a basic computer camera. Subsequently, this data is analyzed to identify potential signs of autism.

By leveraging cutting-edge technology, EASIEST stands as a promising solution that not only addresses the limitations of traditional methods but also provides a more efficient and accessible approach to autism screening for adults.

1.2 Scope

The EASIEST project aims to create an affordable system for detecting Autism Spectrum Disorder using web interactions. To achieve this, the project will follow a series of sequential steps. Initially, the project will explore and test available webcam-based eye trackers. These trackers will then be integrated into a web application, which will allow for the administration of pre-screening tests to individuals. During these tests, participants will be asked to complete specific tasks on designated web pages while their eye-tracking data is recorded. The collected data will be used to compute essential metrics required for predictive models. These models, which have already been developed[6], will be integrated into the web application to predict whether an individual has ASD or not based on the computed metrics. Finally, the project will evaluate the usability of the web application through a comprehensive user study.

In addition to the primary objectives of the EASIEST project, it also aims to address financial and accessibility constraints that often hinder the use of traditional eye-tracking devices in medical institutions. The cost of devices like the Tobii eye tracker (`https://www.tobii.com/`) can be substantial, making it difficult for healthcare providers to incorporate them into their practice.

However, by utilizing standard webcam technology, the EASIEST project significantly reduces the cost associated with autism detection. This makes it more accessible for healthcare providers and eases financial pressures on medical institutions. Ultimately, the EASIEST project has the potential to be highly beneficial in both financial and accessibility contexts for healthcare providers

1.3 Identification of constraints

Economic Constraints

Operational Costs: Expenses related to maintaining the web application, including server costs, data storage, and regular updates.

Firstly, the application will be dispatched on the server that is being mainly controlled from one place, so we have to consider the server costs to accommodate all the users. The server should have enough storage capability as we need not to only store the information of the clients and their patients, but also the intermediate information of the gaze data before it is processed. Lastly, the server should not be limited to the current structure of the application, but also can cope with the system updates.

Affordability for Users: Ensuring the application remains affordable for healthcare providers and patients.

To supply the services for the customers, the fee for its use should still be affordable for the clients but be able to cover the basic expenses of server maintenance, as well as the website admin's personal salary coverage.

Social Constraints

Accessibility: Ensuring the web application is accessible to users with different abilities, including providing clear instructions and a comfortable testing environment.

The application should be available for people of any social class, meaning the application should be available in any hospital that wills to have it with the fulfillment of application requirements.

Political Constraints

Regulatory Compliance Adhering to local and international regulations related to medical data, patient privacy, and digital health applications.

The application should not keep any personal information without the consent of the patients and clients. Anyone should be able to fully delete their personal information if the person demands it.

Ethical Constraints

Data Privacy: Ensuring stringent data privacy measures to protect patient information, including eye-tracking data.

The information obtained from the client's gaze tracking should be not be stored but processed and then deleted. The accessibility to the client's dashboards should be protected and passwords should be stored in an encrypted way.

Informed Consent: Obtaining clear, informed consent from patients before collecting and analyzing their data.

The patients should be fully aware of what kind of data is being obtained, what kind of test they are taking, and how that data is being utilized.

Bias and Fairness: Ensuring the machine learning models used do not exhibit bias and provide fair results across different demographic groups.

The result of the test should not be biased in noones favor and give the result ultimately from the data it received from the gaze tracking.

Sustainability Constraints
Scalability: Ensuring the application can scale efficiently without a disproportionate increase in resource consumption. In case of the system structure improvement, or clients increase, the system should be able to handle the load.

Manufacturing Constraints
Compatibility: Ensuring the software is compatible with widely used hardware and operating systems to avoid the need for custom manufacturing. The application should be accessible from any computer which anyoperating system as long as it has any browser installed. The screen of the client, where the patientwill take the test should be big enough for the patient to see everything. The camera should be functional to record the eye-tracking data and record with a fair quality.

Quality Control: Maintaining high standards for software performance and reliability, particularly in terms of the accuracy of the eye-tracking and ASD detection algorithms. The recorded information should be reliable and belonging to the actual person taking the test.

1.4 Related Work

Detecting autism in young children is a pivotal responsibility shared by parents and healthcare professionals. The first step in identifying autism is the analysis of social behaviors. Early identification opens the door to timely interventions and improved developmental outcomes[7]. While methods such as behavioral observation and parent interviews play a crucial role in early diagnosis, it's essential to acknowledge that advanced diagnostic tools like MRI machines can also be valuable. MRI's role in detecting autism is a significant development in the field[8]. However, gaining access to MRI-based diagnoses may pose challenges, both in terms of limited availability and the associated costs, rendering it less accessible for widespread use

Understanding the significance of enhancing the lives of individuals with undiagnosed autism, we have committed to the development of an affordable and user-friendly web-based eye tracking system. This innovative solution aims to enrich the quality of life for those who may have missed the benefits of early diagnosis by offering a means to comprehend their needs, enhance communication, tailor interventions, and ultimately diagnose autism in undiagnosed adults. Importantly, it does so in a manner that is not only cost-effective but also secure.

Let's delve into our research on webcam-based eye tracking devices and the machine learning techniques we initiated for this project, seeking to enhance our understanding and capabilities in these areas.

1.4.1 Webcam-based eye trackers

According to our research, there are free web-based eye trackers on the market [4]. For our application, we reviewed the literature on eye trackers, which are open source and can be easily integrated into our web application, taking into account other functions. In comparing five eye trackers that could be suitable for our application, Our main goal was to find an eye tracker that would provide a huge cost advantage for doctors to diagnose autism in their patients.

Software	**Device**	**Method**	**Model**	**Training Data**	**Accuracy**
Webgazer[9]	desktop	feature	RR	Pupil location and eye features	4.06 cm
TurkerGaze[10]	all	feature	RR + SVR	Eye appearance feature vector	1.04 cm
GazeRecorder[11]	all	n/d	n/d	n/d	1.75 cm
RealEye[12]	all	feature	RR	Pupil location & eye features	n/d

Table 1: Shown in Table 1 (Heck, Becker, & Deutscher, 2023, Table 1), which contains web-based eye-trackers comparison.

Table 1 gives a complete summary of various eye-tracking methods.It outlines key characteristics based on device, method, model, training data, and accuracy. The "Device" column refers to the platform or hardware each method is designed for. "Desktop" means it can be used on computers or laptops, while "All" means it can be used on different platforms. In the "Method" column, techniques such as "Feature" extract high-level features for gaze estimation. The "Model" column shows predictive models, including Ridge Regression (RR) and RR + SVR. "Training Data" gives information about the data sources. The "Accuracy" column reports gaze estimation accuracy in centimeters.

Additionally, as seen in this table, we encountered various options but faced challenges regarding code integration and compatibility. Notably, some alternatives, like TurkerGaze[10], are no longer operational. We also opted not to proceed with RealEye[12] and GazeRecorder[11] due to their requirement for collaboration on using their libraries and APIs, as we aimed to complete this project independently. Furthermore, apart from the table above, our attempts to integrate pyGaze[13] turned out to be overly complex for our application.

However, WebGazer[9] stands out as an attractive choice for several reasons. It openly shares its library and undergoes continuous improvement. The paper associated with WebGazer introduces a fixation algorithm that combines fixations and cursor activity (RR+F+C) with a reported error rate of 210.6 pixels (Heck, M., Becker, C., Deutscher, V., 2023, 3843). Additionally, the paper suggests that an extended or customized fixation algorithm could be implemented. Therefore, in order to improve WebGazer's fixation algorithm, we can use a special fixation algorithm ourselves or choose and use another fixation algorithm. Below are the fixation algorithms that may be useful for us.

Method	Accuracy	Speed	Robustness	Implementation Ease
Velocity Threshold (I-VT)	good	very good	not as good	very good
Hidden Markov Model (I-HMM)	very good	very good	very good	not as good
Dispersion Threshold (I-DT)	very good	good	very good	good
Minimum Spanning Tree (I-MST)	good	not as good	very good	not as good
Area-of-Interest (I-AOI)	not as good	good	good	good

Table 2: The qualities of fixation algorithms, Salvucci & Goldberg, 2000, Table 7

We categorize them based on their performance across different criteria: Velocity Threshold (I-VT) performs adequately with good accuracy, showcasing very good speed in real-time processing. However, its robustness suffers near the threshold, occasionally misinterpreting fixations due to its strict thresholding. Nevertheless, it's straightforward to implement, offering high ease compared to more complex algorithms.

Hidden Markov Model (I-HMM) exhibits high accuracy and robustness, especially in noisy environments, owing to its probabilistic nature. Though it operates efficiently, its reestimation complexity poses challenges in implementation, being less straightforward compared to simpler methods. Dispersion Threshold (I-DT) stands out with robust fixation identification, offering very good accuracy and speed in processing. However, its interdependent parameters require meticulous adjustment, impacting ease of implementation.

Minimum Spanning Tree (I-MST) provides additional means for fixation characterization but faces challenges in accurately identifying fixations without sequential information. Despite offering very good robustness, its complex global analysis and multiple traversals over the MST hinder straightforward implementation.

Area-of-Interest (I-AOI) method, although efficient to implement, compromises accuracy by including saccade points, leading to inaccuracies in fixation identification. Despite its good speed and ease of implementation, its accuracy is compromised due to the inclusion of saccade points

1.4.2 Machine Learning Techniques

Machine Learning (ML) stands as a versatile tool with diverse applications, and healthcare is among the domains where its impact is particularly significant. In healthcare, ML has been employed for various purposes, such as disease diagnosis, personalized treatment plans, and predicting patient outcomes. For instance, ML algorithms can analyze medical imaging data to detect abnormalities or assist in identifying patterns indicative of specific conditions, showcasing its potential in revolutionizing diagnostic practices.

Machine learning (ML) emerges as a viable tool in autism detection. It has been proven that people with autism has different information processing strategies compared to those without autism[14]. This understanding forms the basis of our approach, wherein we utilize machine learning to identify subtle variations in eye gaze indicative of the presence of Autism Spectrum Disorder.

In the literature, there exist two machine learning models, which achieve an acceptable level of accuracy in detecting liability to Autism Spectrum Disorder from eyetracking data. For this project, we used these 2 different Machine Learning model. In the first model [6], machine learning

was trained by analyzing eye data. The page-based (AOI) method was used to obtain eye data. During the analysis of this data, specific features were generated. For example, first look to area of interest (sec), revisit (#), fixation count (#), time viewed on element (sec). Subsequently, through the utilization of these features, Logistic Regression revealed the extent to which individuals are predisposed to autism. In the second model [14], eye data was collected in the same manner, and an ML model was trained. However, this time, the eye data was generated according to the Fixation-based method. Unlike the previous approach, web pages were divided into grids, leaving no area on the web page to be disregarded.

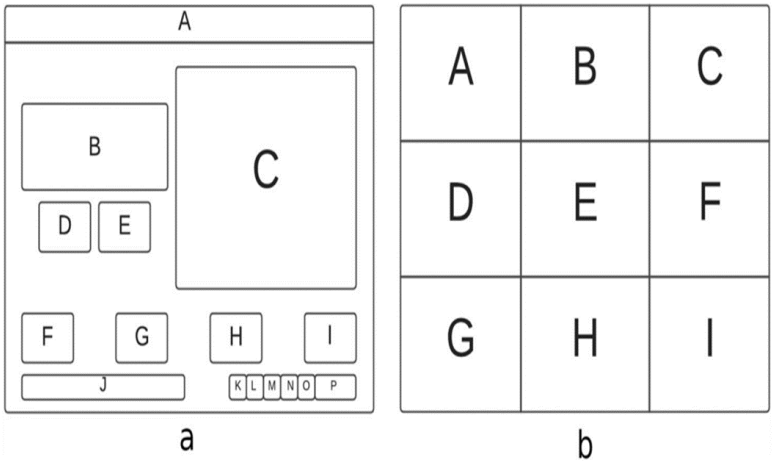


Figure 1: (Khalaji, E., Eraslan, S., Yesilada, Y., Yaneva, V., 2023)

This new approach enabled the development of new features, leading to a higher classification accuracy. Additionally, the Decision Tree algorithm was employed for ML in this model.

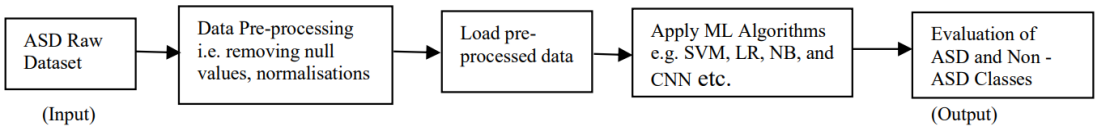


Figure 2: (Raj and Masood, 2020)

This diagram visually represents the sequential steps involved in the process, from data collection to Machine Learning analysis, leading to the determination of an autism diagnosis for the patient Furthermore, within the realm of utilizing machine learning for autism detection, it is crucial to recognize the diversity of visual stimuli employed in research endeavors. Previous studies[15] have incorporated a range of visual stimuli, encompassing not only natural images but also videos. This consideration of varied stimuli types is essential for refining the accuracy and applicability of machine learning algorithms in detecting Autism Spectrum Disorder (ASD). In our project, we

record patients' eye data through a webcam-based eye tracker and integrate it with a pre-trained model[6].

Figure 3: (Yaneva et al., 2018)

As seen in this figure 3, the test pages are divided into certain areas containing the correct answers. This AOI helps analyze and measure how patients attend to and interact with important visual elements relevant to tasks. For each AOI, we classify this data according to Time to 1st

View (sec), Revisits (#), Fixations (#), Time Viewed features. These classified data points are then processed using a logistic regression algorithm within machine learning model. This approach allows the model to return accuracy and results for autism detection.

Feature	Explanation	[6]	[14]
GRID BASED	Cells which are in form of grid segmentation		✓
AOI BASED	Segmentations based on website elements	✓	
TIME	Time based on milliseconds		✓
Media ID	The identifcation number of each web page		✓
Time To First View	The time in seconds to first fixate an AOI after the web page is displayed	✓	
Revisits	The number of times a participant revisited a previously viewed AOI	✓	
Fixations	The number of gaze fixations per participant per AOI	✓	
Time Viewed %	The total duration of all fixations per participant per AOI in seconds	✓	

Table 3: Comparison of previous models features
Key: ✓= uses this feature

According to this table, we can compare different features used in eye tracking studies and we can see in which ML model these features are utilized. Each feature represents how eye tracking data is segmented or certain metrics are measured. The table includes segmentation methods such as GRID-BASED and AOI-BASED, time measurements, descriptive features like Media ID, and various eye tracking metrics.

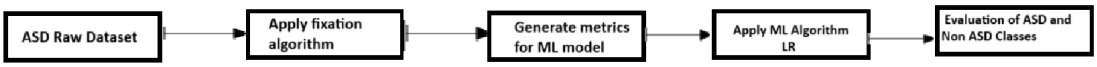


Figure 4: General sequential steps of our project

This diagram visually represents our sequential steps in the project, from data collection to Machine Learning analysis to determining the patient's autism diagnosis.

1.5 Product Overview

1.5.1 Product perspective

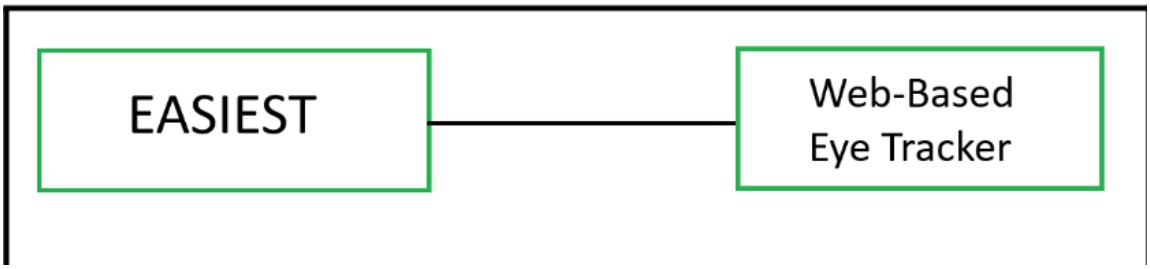


Figure 5: Block Diagram

In Figure 5, the EASIEST system is connected to only one external system, which is webGazer.

1.5.2 System Interfaces

Webcam-Based Eye Tracker Software Interface: This external system captures eye-tracking data while users interact with the web application. The functionality of this interface is essential for the application to assess eye movements and behavior, a key factor required in the machine learning predictive models for ASD screening.

1.5.3 User interfaces

Main Page: The user can register by clicking the register button on the home page. After registering, the user can log in with the login button on the home page. After logging in, users can easily manage their patients by clicking the "dashboard" button on the home page. In addition, in the dashboard section, the patient can start the autism test process for the patient registered in the system.

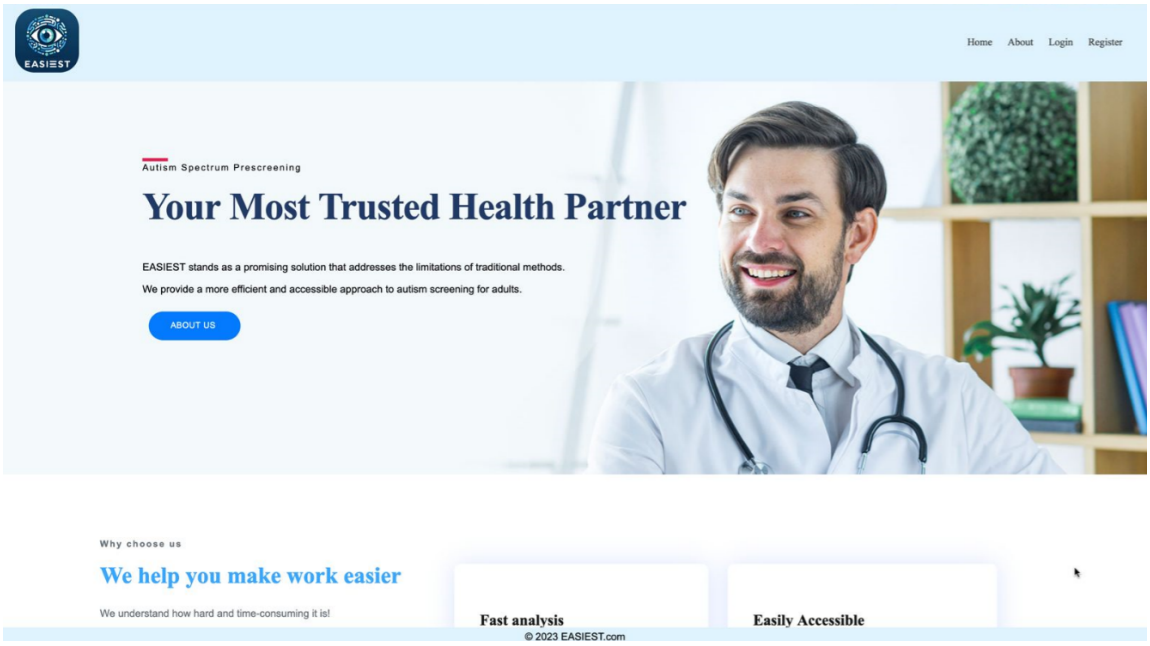


Figure 6: Main Page Of The Web site

Login Screen: To log in, users can find the link at the top right corner of the homepage. Following this link will take them to a page with input fields for Email and Password. If the credentials are entered correctly, users will be redirected to the dashboard page. If the login details are incorrect, an error message will prompt users to re-enter their information.

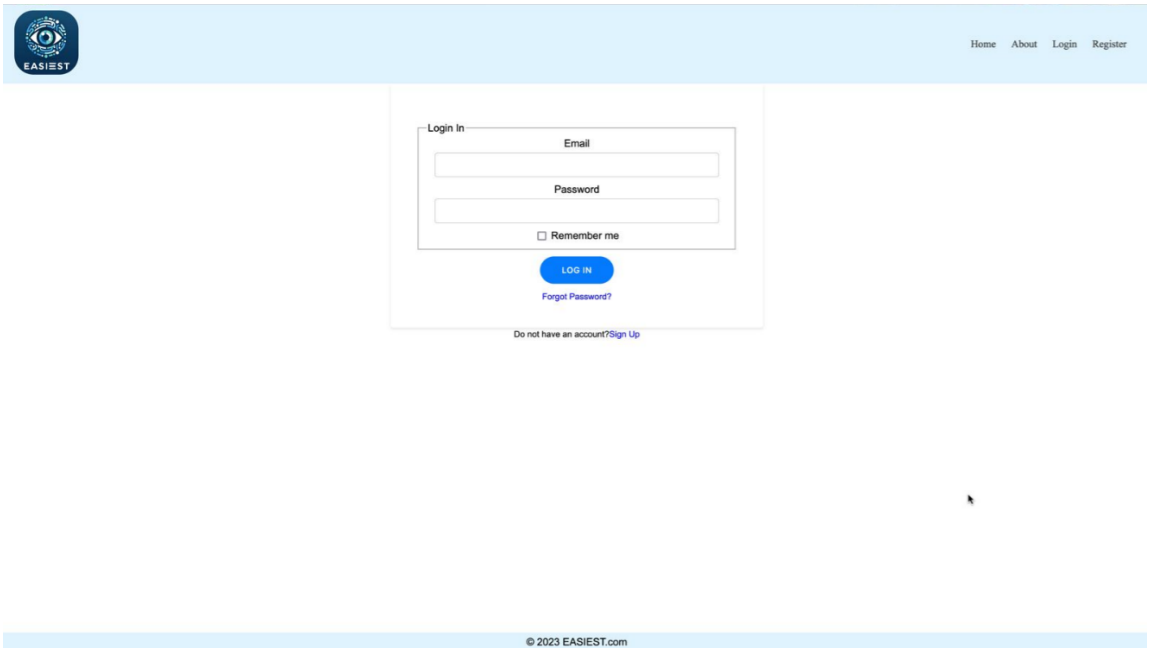


Figure 7: Login page Of The Web site

Doctor Registration: To register on the website, users should open their browser and navigate to the "Register" page, accessible via a link at the top right corner of the homepage. Once there, they will see a registration form requiring details such as Name, Surname, and Password. After completing the form, users can press the "Sign In" button to submit their information. This information is recorded in the database, and the user must wait for the website admin to validate their account. During this waiting period, users can still log in, but all functionalities will remain hidden until their account is validated.

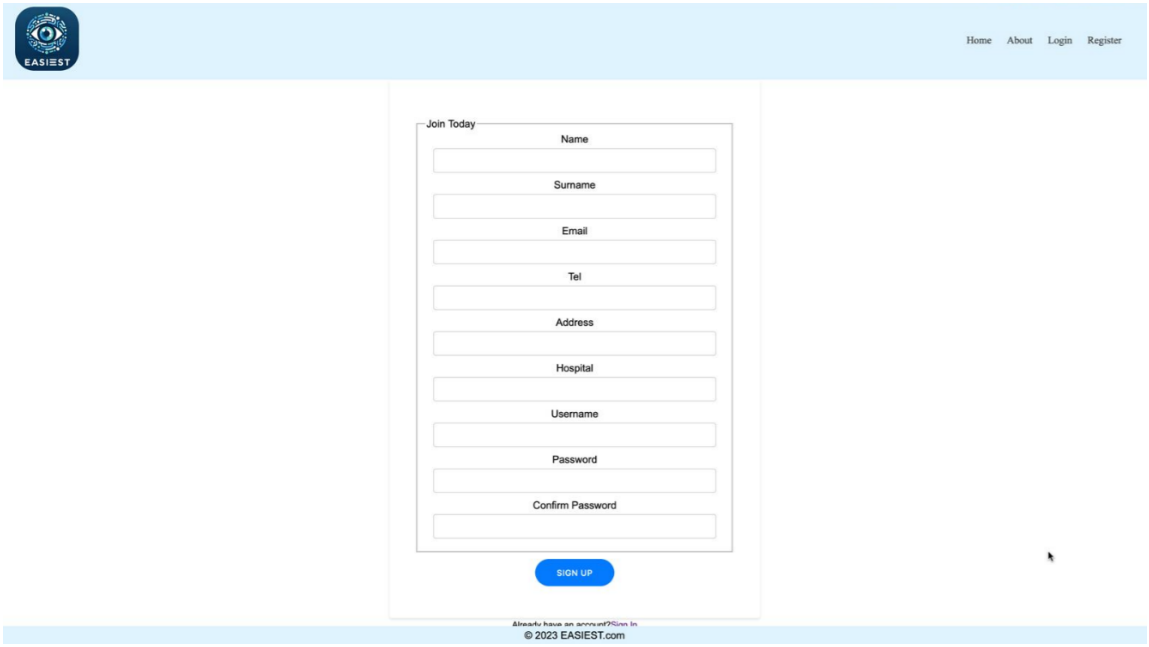


Figure 8: Register page Of The Web site

Patient Screen: On this screen, the registered user can add the patient to the system by entering the patient's details. Additionally, he/she can view the last four records and, if desired, view the patient by typing the patient's name in the search bar. These records can be deleted with the delete patient button located near the patient. He/she can see the patient's details along with the patient's test results. registered users have two options to start a test. They can select one of the available patients from the list and click on the start button next to their details. Alternatively, by entering their name, they can use the search bar to find and select the patient they want to test. In addition, the registered user can look at the history of the patient they have selected and see the results of their previous tests, for example, when the test was performed, they can also see the result for the first ML model result and the result for the second ML model along with the date.

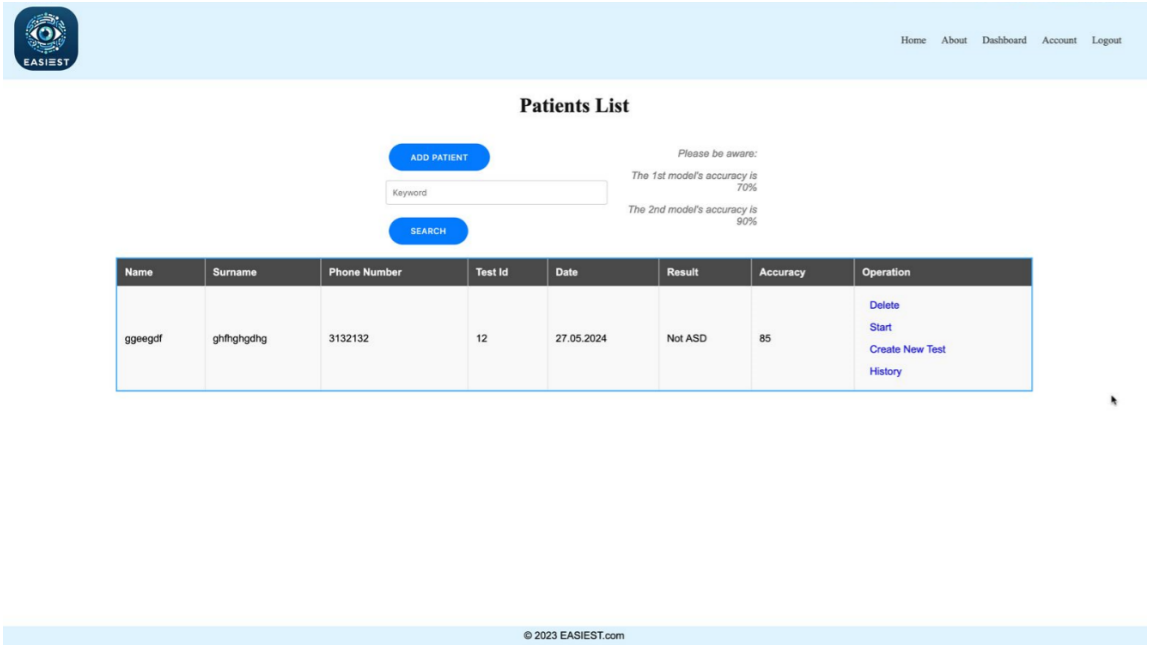


Figure 9: Patient Page Of The Web site

Autism Test Screen: On this screen, patients actively search for answers to the doctor's questions. When they find them, patients communicate their findings to the doctor, and the doctor clicks the next question button to continue.

Figure 10: Test Screen While Recording Eye Movement

In Figure 9, Patients solve the test in full screen and can move on to other tests with the "next page button" while they are being recorded.

Admin Screen Admins can log into the admin panel by entering their login credentials into the Email and Password fields. Successful login redirects the admin to the admin dashboard. An error message will appear if the credentials are incorrect, prompting the admin to re-enter the information.

When an admin logs into the system, they can view a dashboard displaying a table of currently registered doctors. The table includes all relevant information for each doctor, and the rightmost column contains 'Approve' and 'Remove Validation' buttons. The 'Approve' button validates a doctor, granting them full website functionalities. The 'Remove Validation' button revokes a doctor's access to functionalities, limiting them to account modifications only.

- **Invalidated Doctor:** A doctor who has not yet been validated by the admin can access the dashboard, but all functionalities will be hidden.
- **Validated Doctor:** Validated doctors have access to four key functions for each patient: delete, start test, create new test, and view history.
  - Delete Button: Allows the doctor to delete a patient's record.
  - Start Button: Initiates the ASD detection test for a specific patient and redirects to the calibration page.
  - Create New Test Button: Enables the doctor to start a new test for the patient.
  - History Button: Displays detailed information about a specific patient, including previous test dates, results, and calibration accuracy

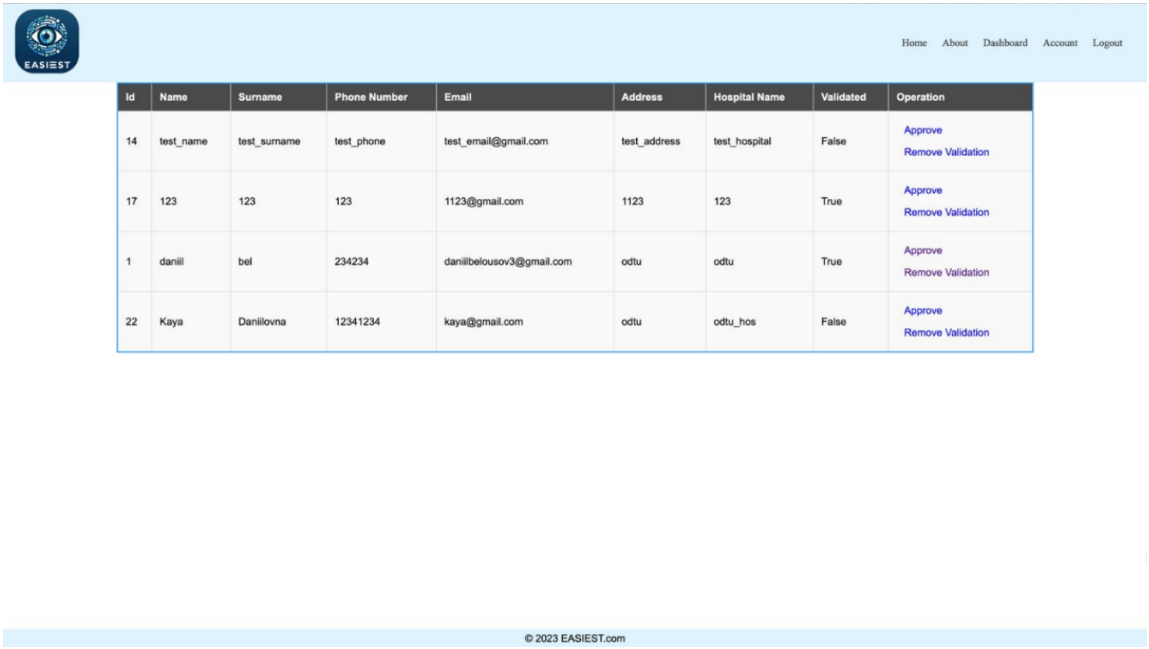


Figure 11: Admin panel of the web site

1.5.4 Product functions

- The system shall allow health professionals to search patients with keyword.
- The system shall allow health professionals to register their clients for the potential test.
- The system shall allow health professionals to register themselves into EASIEST.
- The system shall allow the patients to pass a calibration before taking the test.
- The system shall allow only the health professionals to see the results of clients' test.
- The system shall allow the patients to switch between web-pages of the test.
- The system shall allow interaction with a series of web pages by completing specific tasks while gathering the eye-tracking data.
- The system shall allow the data collected from eye-tracking to be converted to features that can be understood by the machine learning model.
- The system shall allow to analyze the data collected from eye-tracking, and predict about whether patient likely has ASD.

1.5.5 Identified stakeholders and design concerns

Patient Concerns:

- **Confidentiality:** Concerns about the handling and security of personal data, especially eye-tracking information.
- **Consent and Control:** Desire for clear information and control over how their data is used and shared within the system.
- **Comfort and Accessibility:** Ensuring the test environment is comfortable and accessible for individuals with varying abilities.
- **Clarity of Instructions:** Clear instructions and guidance throughout the testing process to minimize stress or confusion.
- **Trust in Diagnosis:** Concerns about the accuracy and reliability of the system in identifying Autism Spectrum Disorder.

Healthcare Professional Concerns:

- **Reliability of Screening:** Concerns about the accuracy and reliability of the system in identifying Autism Spectrum Disorder.
- **Clinical Validity:** Assurance that the system's diagnostic outcomes align with established clinical standards.
- **Seamless Integration:** Desire for the system to integrate smoothly into existing healthcare IT infrastructure and workflows.

- **Usability:** Ease of use and user-friendliness of the system within a clinical setting.
- **Patient Data Protection:** Ensuring patient confidentiality and compliance with healthcare data regulations.
- **Secure Data Transmission:** Confidence in the secure transmission of sensitive patient information within the system.
- **Explanation of Results:** There is a need for clear explanations of the system's output to aid clinical decision-making.

1.5.6 User characteristics

- Patients have to be over 18 as the application is dedicated to diagnose adults. In addition, they have to know at least the basics of human-computer interaction, since the application is web-based.
- Health professionals have to know more advanced the basics of human-computer interaction, as they have to register themselves as well as the patients for the test.

1.5.7 Limitations

- The system has only compatibility on the following browsers ; Google Chrome, Microsoft Edge, Mozilla Firefox, Opera, and Safari. Otherwise, the system might not work properly.
- The system should be able to recognize the face due to the light condition, distance between face and the web-cam, and human face movement.
- The system does not have its own eye-tracking system, so only a specific eye-tracking API (WebGazer) collaborates with this website.
- The individuals expected to take the autism test should not have visual impairments to the extent that screen movements cannot be perceived.
- The system has only compatibility with desktop screen sizes, not mobile phones or tablets.

1.5.8 Assumptions and dependencies

- We assumed that the web browser through which users interact with the EASIEST should be up to date to support all the functionalities.
- We assumed that the records received from the users were collected with their consent.
- We assume that there will be a stable internet connection.
- We assume that the setup of the system will in a room with appropriate light conditions.
- We assume that the doctors will be able to manage the process well, for example, ensuring that the patients do not move their head a lot, do not use a keyboard or mouse, as it is not allowed in this mode.
- We assume that patients will tell doctors immediately they complete their tasks, so that doctors can move on.

1.5.9 Definitions

ASD: Autism spectrum disorder

2 Specific requirements

2.1 External interfaces

System Interface	Functionality	Input	Output
WebGazer	Providing eye-tracking functionality	* Web-cam * Web-page * User Permission	* Eye-tracking data 1- (x, y, time) * Accuracy level 1- Percentage

2.2 Functions

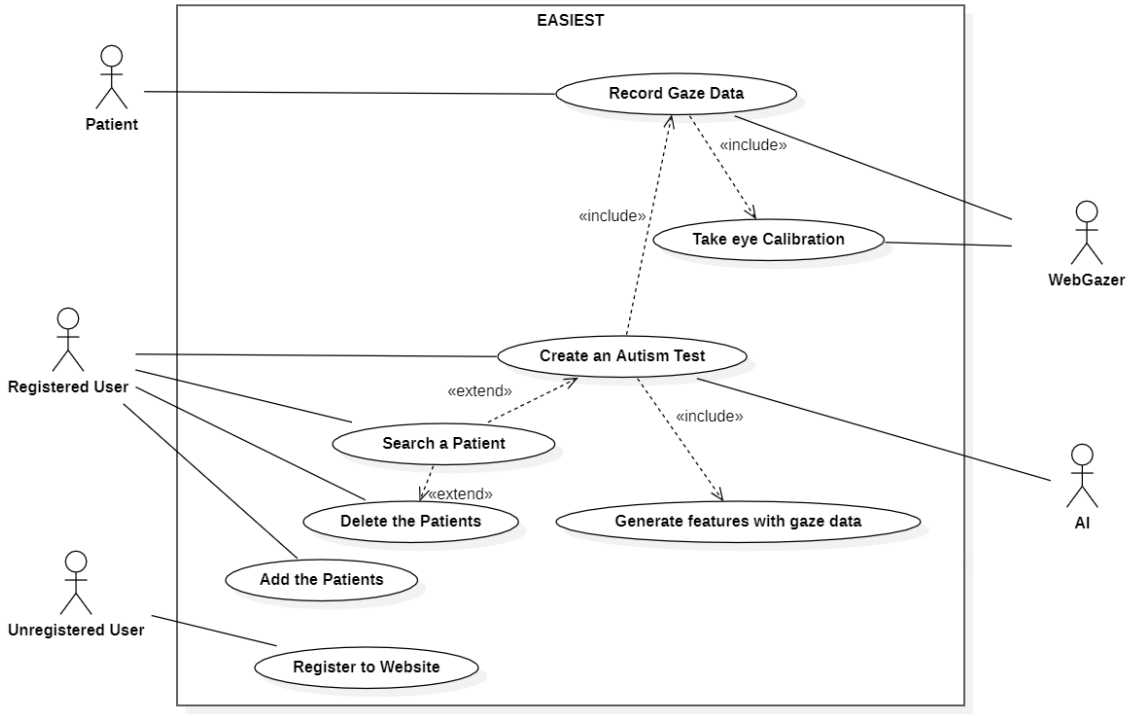


Figure 12: Use Case Diagram

Use case	Register to Website
Actors	Unregistered User
Cross references	N/A
Typical Course of Events	
Actor Intentions	**System Responsibility**
1.Unregistered user presses the "register button" on the main page of the website.	
	2. System prompts a registration form to enter personal details.
3. Unregistered user enters the personal details.	
	4. System prompts a user to enter a valid doctor certification.
5. Unregistered User uploads a valid doctor certification.	
	6. System checks the entered details .
	7. System displays information about whether registration is successful or not
	8. System adds the user into the database.
Alternative Courses	
Step 6: If an unregistered user enters an invalid personal information or invalid doctor certificate, then the system shows an error message and returns back to Step 3.	

Figure 13: Register to Website Use Case Details

Use case	Add the Patients
Actors	Registered User
Cross references	N/A
Typical Course of Events	
Actor Intentions	**System Responsibility**
1.Registered user presses the "Patients Button" on the dashboard.	
	2. System prompts all available patients for the user.
3. Registered user clicks the "Add" button above the available patients list.	
	4. System prompts a small-scale information form about the new patient.
5. Registered user completes the information about the new patient.	
	6. System inserts the new patient to the database.
	7. System displays information about whether adding patient is successful or not.
	8. System redirects the user into the dashboard again.
Alternative Courses	
Step 5: If an registered user enters an invalid personal information about patient, then the system shows an error message and returns back to Step 4.	

Figure 14: Add Patients Use Case Details

Use case	Delete the Patients
Actors	Registered User
Cross references	N/A
Typical Course of Events	
Actor Intentions	**System Responsibility**
1.Registered user clicks the "Patients Button" on the dashboard.	
	2. System prompts all available patients for the user.
3. Registered user clicks the "Delete" button for a specific patient.	
	4. System deletes the patient from the database.
	5. System displays information about whether deleting patient is successful or not.
Alternative Courses	
Step 2: Registered user can also benefit from the keyword searching. (See Search Patient)	

Figure 15: Delete Patients Use Case Details

Use case	Search a Patient
Actors	Registered User
Cross references	N/A
Typical Course of Events	
Actor Intentions	**System Responsibility**
1.Registered user clicks the "Patients Button" on the dashboard.	
	2. System prompts all available patients for the registred user.
3. Registered user clicks the search bar above the patients list and enters a keyword.	
	4. System searches the keyword in the related database.
	5. System displays the searched patient's information on the available list.
Alternative Courses	

Figure 16: Search a Patient Use Case Details

Use case	Create an autism test
Actors	Registered User, AI
Cross references	Recording Gaze Data
Typical Course of Events	
Actor Intentions	**System Responsibility**
1.Registered user presses the "autism test" button.	
	2. System prompts all available patients.
3. Registered user selects a patient from the list.	
4. Registered user clicks the "start" button and initiates the test.	
5. WebGazer starts to record fixation points.	
	6. System prompts the calibration test for the patient.
7. Patient do the calibration test.	
	8. System prompts calibration accuracy results.
	9. System redirects the patient to the test pages.
10. Patient copmletes the given tasks in the test while webgazer recording the fixation points.	
	11. System transforms the recorded data to specific features for the AI model.
12. AI predicts whether the patient has autism or not.	
	13. System generates a report for patient with specified accuracy and features.
Alternative Courses	
Step 8: If a patient couldn't pass the accuracy level 75, returns back to Step 7.	
Step 2: Registered user can also search for a patient.	

Figure 17: Create an Autism Test Use Case Details

2.3 Usability Requirements

- Web-site is easy to use and operates in a casual way, which makes autism detection more feasible.
- In the autism test, the web pages should be shown in a mode where the patients will not be distracted with other elements, such as a button to allow to go to the next page.

- The website should be designed in a way that the doctors can be access the recently added patients quickly, as they probaly need to conduct the test for these patients.
- The website should be include a documentation about how it can be used, and how the results can be interpreted, so that doctors can easily understand how to use it and how to interpret the results.

2.4 Performance requirements

- The web-site is implemented in such a way that threshold accuracy level for calibration is percentage 75, so that we can record eye-tracking data more accurate.
- The web-site must provide autism results in approximately 5-6 minutes , including the entire procedure, so that patients can have information about their results shortly.
- Efficient searching algorithms should be used which is not more than O(N$\hat{2}$), so that our system complexity is minimized as possible.
- The web-site use Dispersion Threshold (IDT), in order to manipulate the gaze points more accurately.
- Calibration time for both initial setup and any necessary recalibrations should done meticulously in order to calibrate accurately. It need around 20-30 seconds for complete it.

2.5 Logical database requirements

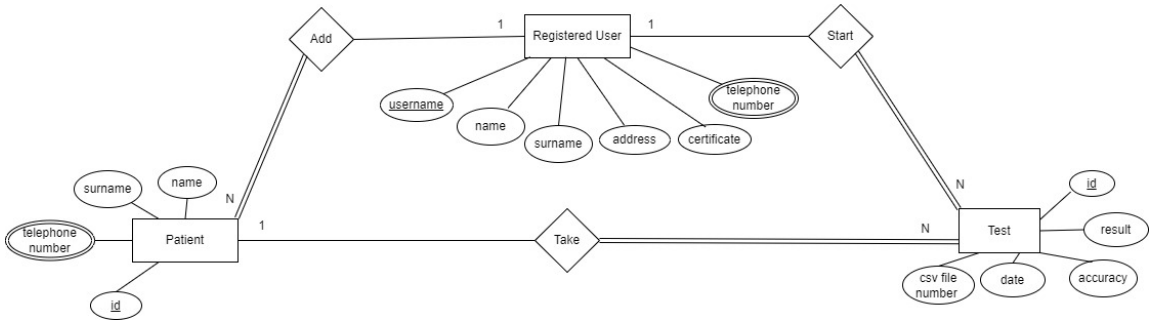


Figure 18: Logical database

2.6 Software system attributes

Reliability

- Our system's unique software architecture prevents unexpected server downtime, which allows us to recover user data more easily.
- Our website has been thoroughly tested and confirmed to work with all major browsers, which gives users peace of mind during test.

Availability

- Our servers are designed to withstand power outages, so patient data is always safe.

Security

- Authorized members, such as doctors and health professionals, are carefully granted access to patient data, ensuring that only authorized individuals can view this sensitive information.
- All sensitive data on the website is encrypted using the PBKDF2 (Password-Based Key Derivation Function 2), making it secure and difficult to decrypt.

Maintainability

- The website's modular algorithm reduces complexity and makes it easier to update and maintain individual components of the web components.
- Thorough documentation of all software code makes it easy for other developers to understand and work on.

Portability

- The website was developed using Python 3, JavaScript, HTML, CSS, and other portable programming languages. This makes it easy to port the website to other systems, allowing us to develop software products for various machines.
- The website has been tested on all major operating systems to ensure that it is platform-independent, meaning that it can be used on a wide range of devices and systems.
- Web-site must work appropriately on all platforms except "Yandex Browser" and mobile phones so it becomes more convenient to implement in hospitals.

2.7 Supporting information

PBKDF2[16]: It applies a pseudorandom function, such as hash-based message authentication code (HMAC), to the input password or passphrase along with a salt value and repeats the process many times to produce a derived key, which can then be used as a cryptographic key in subsequent operations.

3 Software Estimation

Input	Doctors info to register(L), Patient test register (L), Keyword Search (L), Autistic Test Recording(H)
Outputs	Doctors registration confirmation(L), Patients registration confirmation(L), Search result (M), Autistic Test result features (M), Eye-taking data (L), Autistic test result(L)
Inquiries	Registration of doctor (L), Registration of patient (L), Search (M), Record Storing (M)
Logical Internal Files	Autistic Eye-taking data (L)
External	Autistic Test record gaze points (M)

Figure 19: Function Point Analysis

General System Characteristics (GPC)	Degree of Influence (DI) 0: least, 5: most	
Data communications	5	
Distributed processing	0	
Performance	3	
Heavily used configuration	0	
Transaction rates	4	
Online data entry	5	
End-user efficiency	4	
Online updates	1	
Complex processing	1	
Reusability	3	
Installation ease	4	
Operational ease	3	
Multiple sites	2	
Facilitate change	3	
Total of Degree of Influence	38	
Value adjustment factor (VAF)	1,03	= TDI×0,01+0,65 = 38×0,01+0,65

Figure 20: Degree of Influence

Considering the final project we want to present, our projects functions, requirements limitations and predictions we believe that these degrees of influence depict the precise goals of our project with its general system characteristics.

Program Characteristics	Low Complexity	Medium Complexity	High Complexity	Total	
Inputs	3 * 3	0 * 4	1 * 6	15	
Outputs	4 * 4	2 * 5	0 * 7	26	
Inquiries	2 * 3	2 * 4	0 * 6	14	
Logical Internal Files	1 *7	0 * 10	0 * 15	7	
External Interface Files	0 * 5	1 * 7	0 * 10	7	
Unadjusted Total of Function Points (UTFP)				69	
Adjusted Total of Function Points (ATFP)				71,07	= UTFP * VAF = 69 * 1,03

Figure 21: Total of Function Points

Based on the projects function analysis we have calculated the UTFP and ATFP based on the functionals complexity.

Programming Language	Percentage in Project	Language Unit Size	Language Unit Size in Project	Lines of code	
SQL	0,1	13	1,30	92,39	= ATFP * LUSP = 71,07 * 1.30
Javascript	0,5	80	40	2843	= ATFP * LUSP = 71,07 * 40
Python	0,3	20	6	426	= ATFP * LUSP = 71,07 * 6
HTML	0,04	15	0,60	42,64	= ATFP * LUSP = 71,07 * 0.60
CSS	0,06	16	0,96	68,23	= ATFP * LUSP = 71,07 * 0.96
Language Unit Size			48,86		
Lines of Code			3472		
Kilo Delivered Source Instruction (KDSI)			3,47	= ATFP * LU/1000 =71,07 *48,86 / 1000	

Figure 22: Language Unit Size

To calculate the KDSI we have taken all of the programming languages we will be using n the project with their language units to get a rough estimation of the lines of code. From there we used the formula to calculate KDSI directly.

Development Type	Semi-detached	
Estimated Effort in Man-Months	12,10	= a * KDSI^b = 3 * 3,47^1,12
Estimated Development Time in Months	5,98	= 2,5 * MM^c = 2,5 *12,10^0,35
Estimated Team Size	2,02	= MM / Months = 12,10 / 5,98

Figure 23: COCOMO

In COCOMO software estimation we have decided to go with the semi-detached deployment time due to the project complexity. Considering the project to be medium in size, however quite innovative, as we are the first to utilise eye-tracking in the field of autism detection, we believe that this deployment estimation fits us best. The other critical aspect was that we don't have all the time for us, but actually some deadlines, it worked as an additional support for the selection.

Adjusted Total of Function Points (ATFP)	71,07	
Kind of Software	System	
Software Organization's Skills/ Abilities	Average	
Estimated Effort in Man-Months	11,71	= (ATFP^(3*CE)) /27 = (71,07^(3*0,45)/27
Estimated Development Time in Months	5,27	= 3 * mm^(1/3) = 3* 11,71^(1/3)
Estimated Team Size	2,00	= MM / Months = 11,71 / 5,27

Figure 24: Jones' First Order Effort Estimation

In Jones' First Order Effort Estimation we have gone with the system type estimation because the final project aims to be a system, the final consumers will use. The project itself will be providing the access to the inbuilt functionalities to assist the users in autism detection by the use of software and hardware.
The software organisational skills were considered from the experience of the team members. Having

little to no experience in javascript, HTML and CSS, however having knowledge about SQL and Python, we decided to go with the average software skills prediction.

Schedule Tables Estimate	Efficient Schedules	
Lines of Code	3472,48	< 10 000
Estimated Effort in Man-Months	24,00	
Estimated Development Time in Months	8,00	

Figure 25: Schedule Tables Estimate

In Schedule Tables Estimate we have chosen Efficient scheduling we have decided to go with the efficient scheduling for the system product because we believe it fits us best. We believe this schedule is a more realistic estimate that considers potential delays, uncertainties, and resource constraints that may arise from our skill sets.

Overall, considering the two estimation techniques (CoCoMo and Johnson's first order), we received pretty much the same estimations in development time of 6 months. Moving to the Schedule Table estimation, we received a value of 8 months for development. Considering that we are not professionals, and most of the topics are unfamiliar to us, we believe the estimations are realistic and accomplishable. However, all the estimations resulted in the need for a team of only two people, but as our group is of 4 people, we believe we can manage to finish the project earlier.

4 Architectural Views

4.1 Logical View

4.1.1 Class Diagram

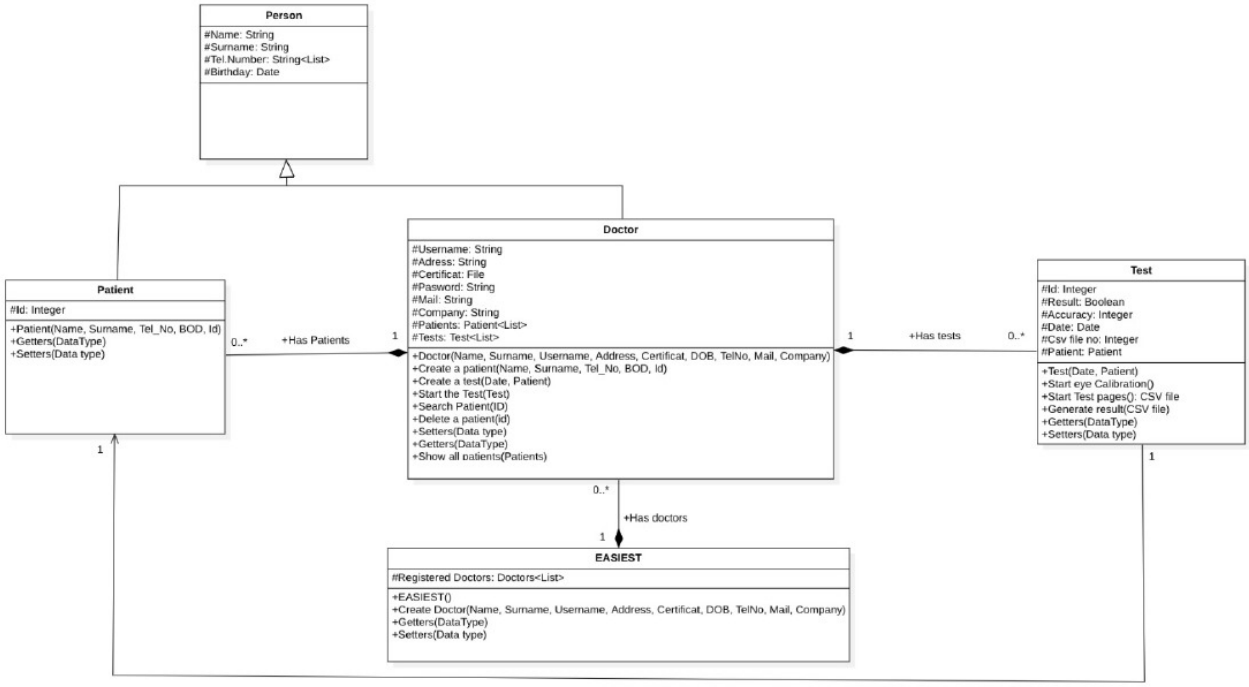


Figure 26: Class Diagram

Design rationale: The class diagram represents our classes generated from the database tables. It provides an abstract view of our system where the main functionalities could be accessed from the companies EASIEST class. This class will provide the access to all the doctors that will be stored in the system. The doctors class has all the functionalities of creating the patient and the tests, making it possible to run the autism detection test.

4.2 Process View

4.2.1 Activity Diagram

Register To Website

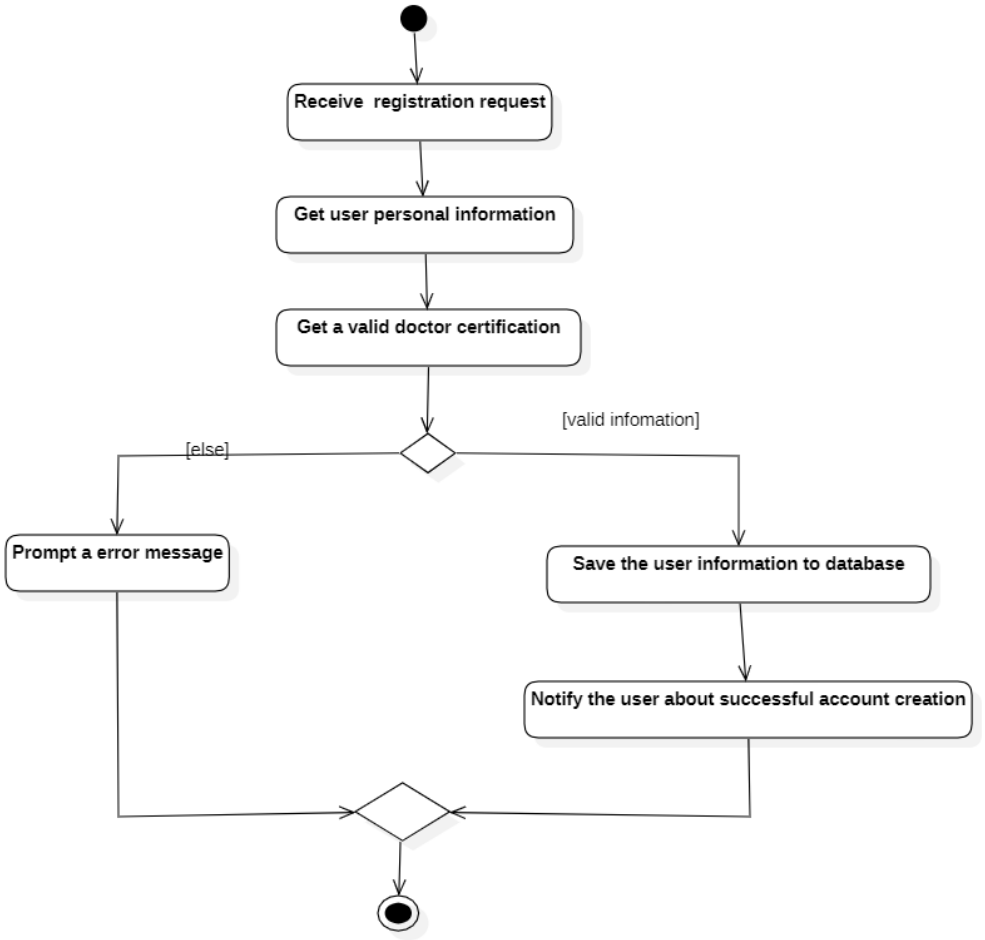


Figure 27: Activity diagram of Register to website

Design rationale: First, the system receives a registration request; Then, it gets personal information and a doctor's certificate from the user. If the account information or doctor certificate is valid, the system saves the user account into the database and notifies the user about successful account creation. The system prompts an error message if the personal information or doctor's certificate is invalid.

Add Patients

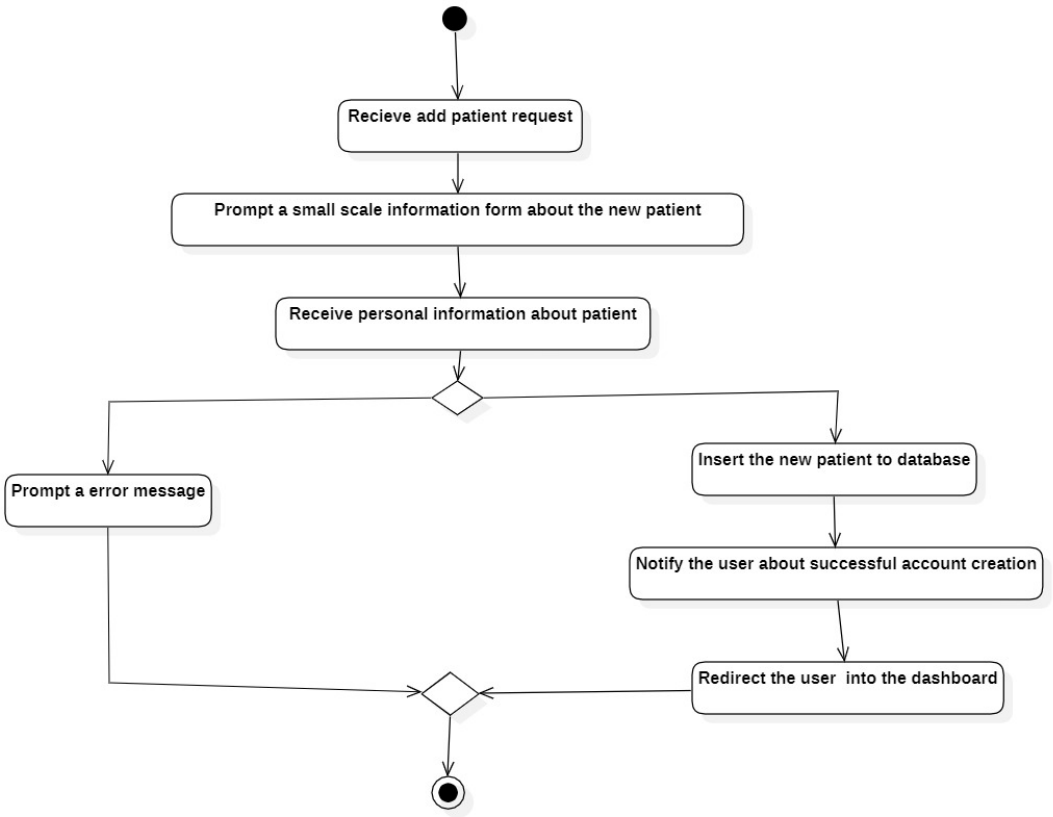


Figure 28: Activity diagram of Add Patients

Design rationale: First, When the system receives an 'add the patient' request, it initiates a concise information form where the registered user can enter the new patient's name, surname, telephone number, and birthday. The system then validates the entered information for accuracy. If the patient's details are found to be valid, the system proceeds to add the patient to the database and presents a confirmation message. Afterward, the system automatically redirects the user to the dashboard.

Search Patient

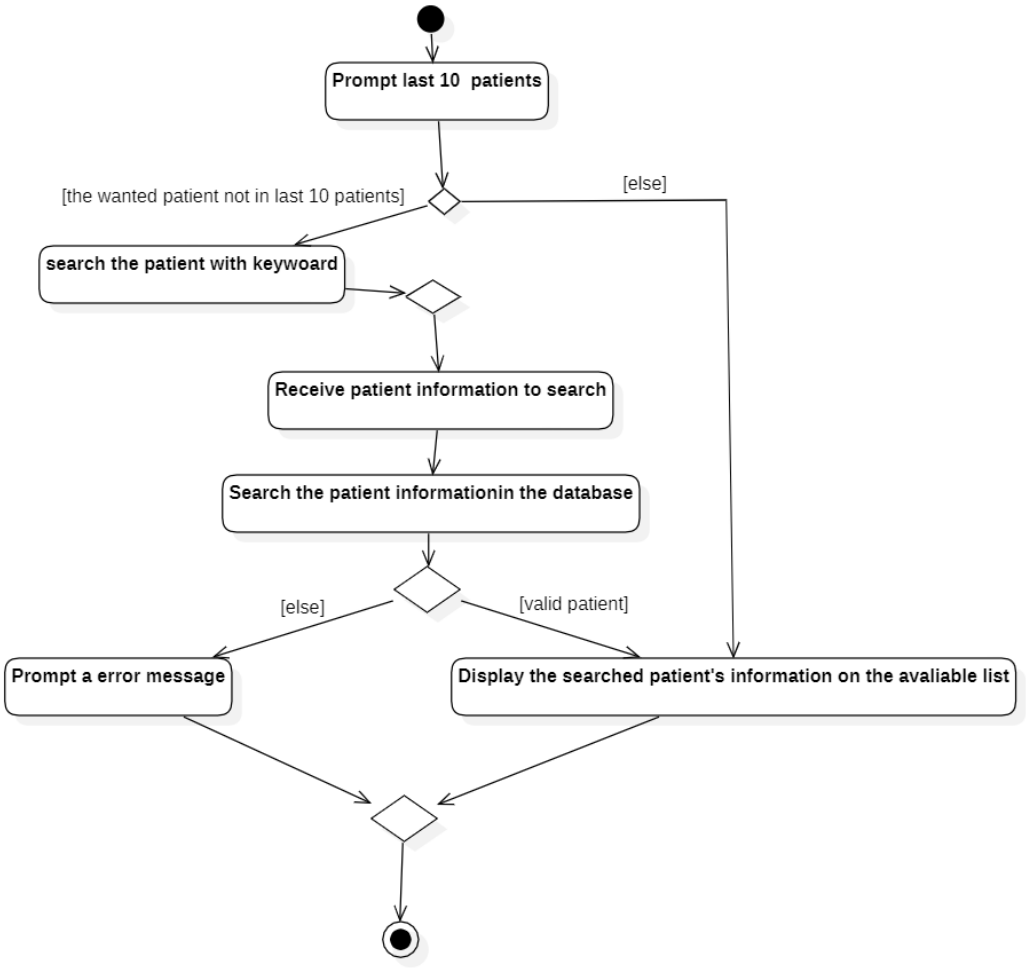


Figure 29: Activity diagram of Search Patient

Design rationale: First, when the registered user presses the "Patients Button," the system shows the last ten patients for the user. The user can select the patients from the list, or he/she can search for a patient by the keyword. If the user searches for a patient by the keyword, the system receives patient information from the user. If the patient's information is valid on the database, the system Displays the searched patient's information on the available list. The system prompts an error message if the patient information is not valid on the database.

Delete Patient

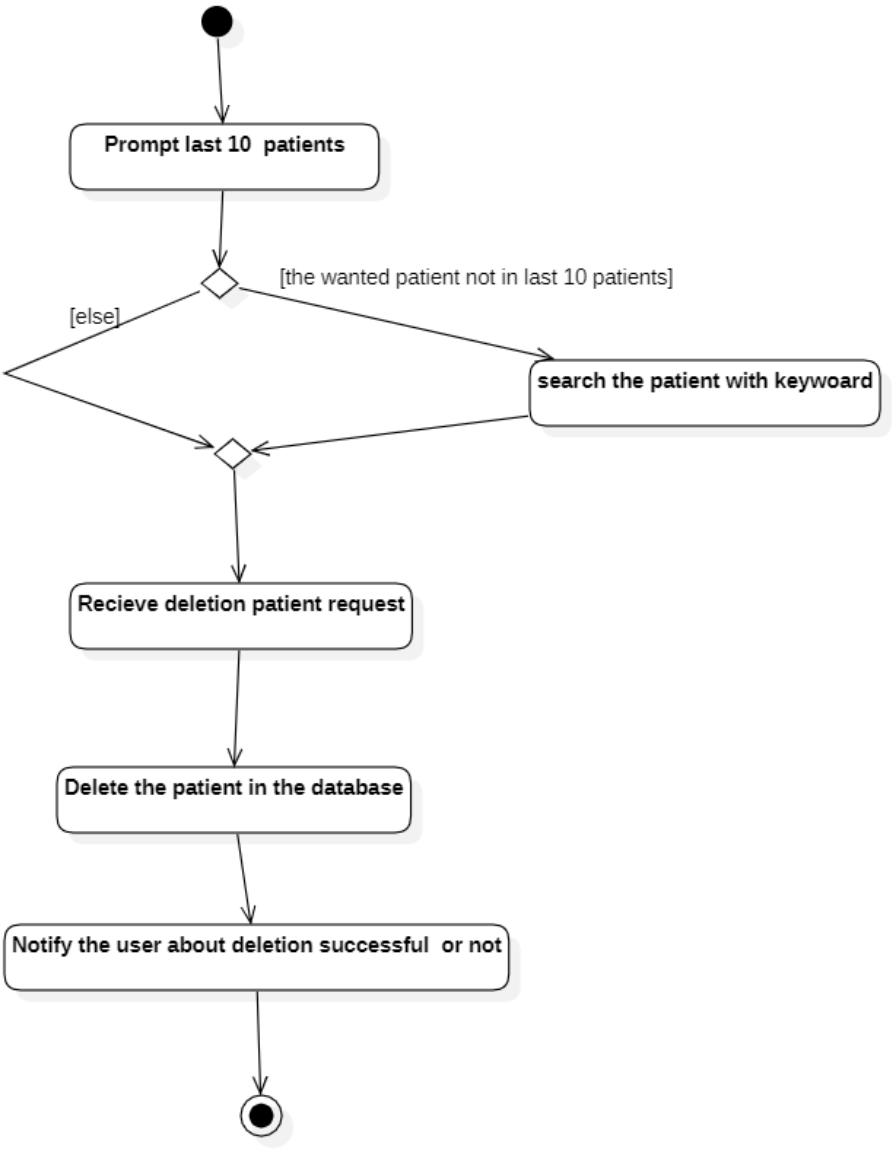


Figure 30: Activity diagram of Delete Patient

Design rationale: First, when a registered user clicks the 'Patients' button, the system initially presents the last ten patients for the user's review. If the desired patient is not found among the last 10 patients, the user can initiate a search using keywords. Then, the registered user can click

the delete button for a specific patient, and the system proceeds to remove that patient from the database. Finally, the system provides feedback on the success or failure of the patient deletion process.

Create an Autism Test

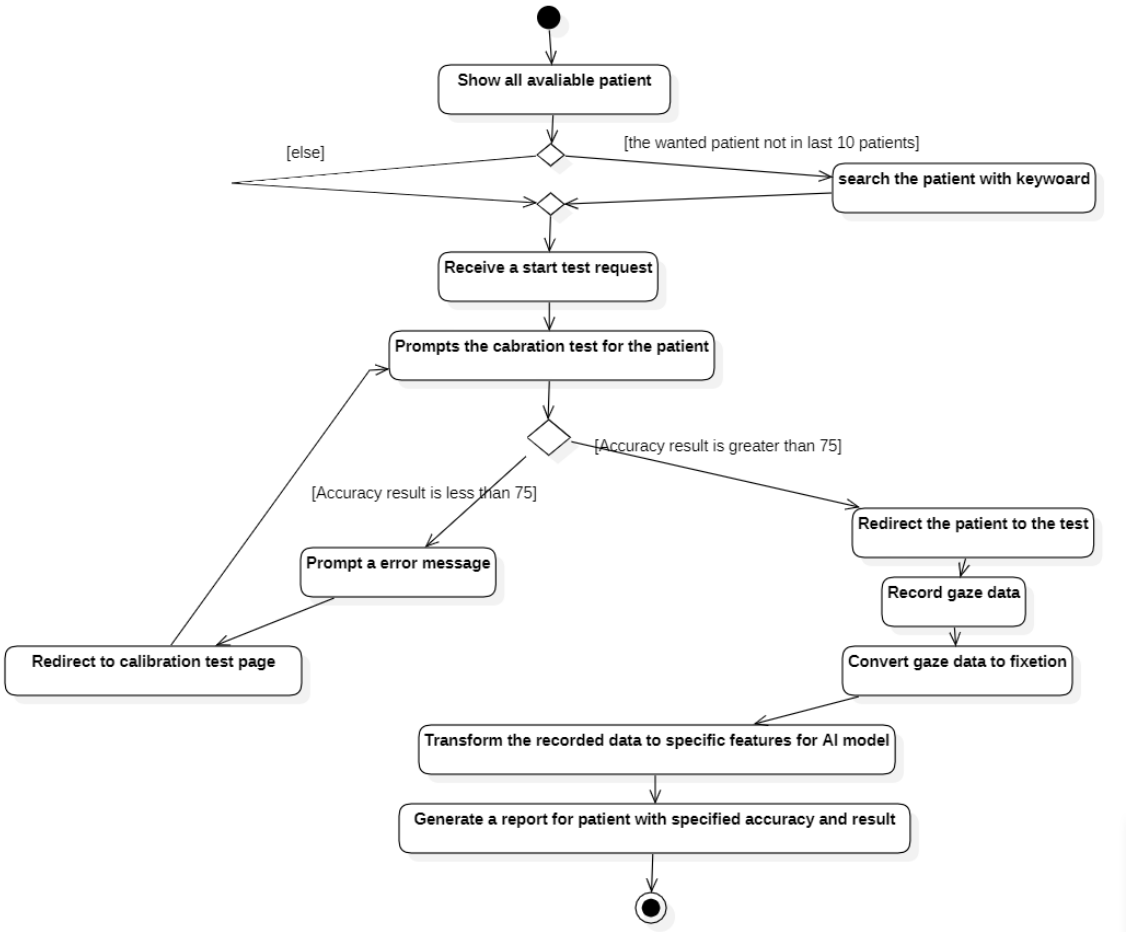


Figure 31: Activity diagram of Create an autism test

Design rationale: First, when the registered user presses the "autism test," the system initially presents the last ten patients for the user's review. If the desired patient is not found among the last 10 patients, the user can initiate a search using keywords. Then, when the registered user selects a patient, the system receives the selected patient information. The system receives a start test request and prompts the calibration test for the patient. The system checks calibration accuracy results. If the result is less than 75, the system prompts an error message and redirects to the calibration test page. If the calibration test result exceeds 75, the system redirects the patient to the test screen. While the patient is performing the test, the system records the gaze data, and

when the test is completed, the system converts the gaze data to fixation and sends it to the ML for specific features. ML generates a report for the patient with specified accuracy and features.

4.2.2 Sequence Diagrams

Register To Website

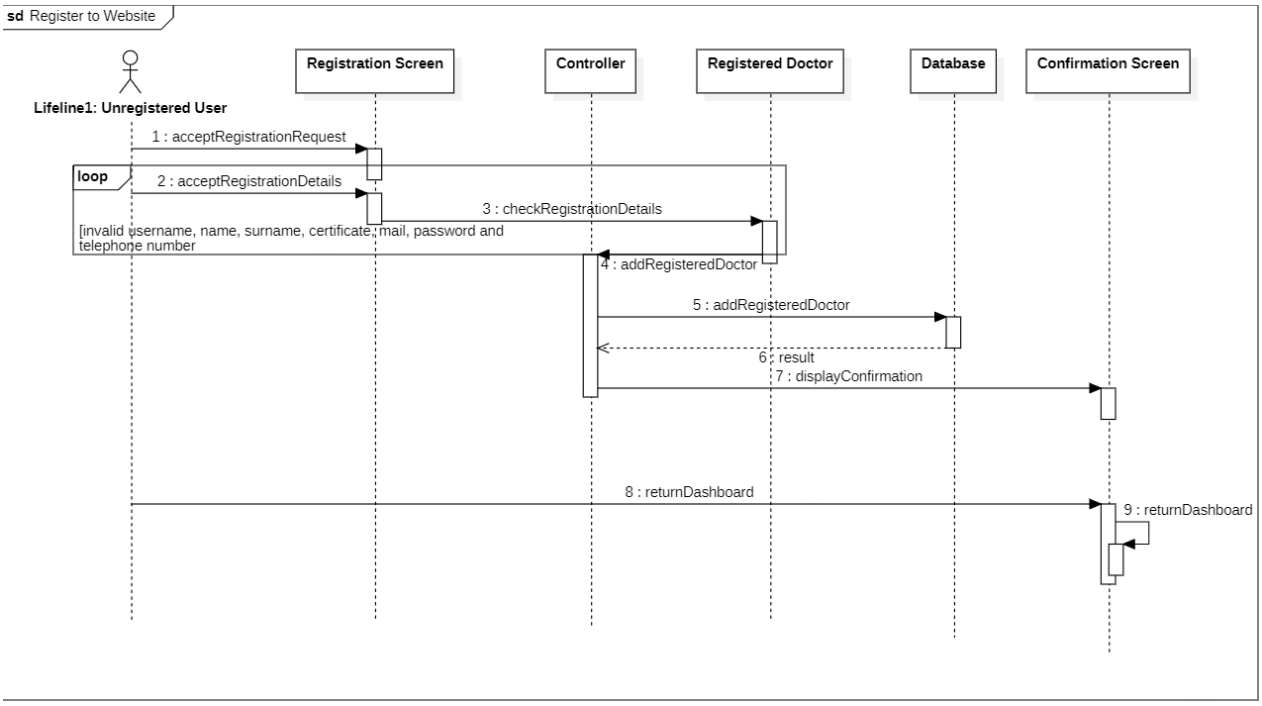


Figure 32: Sequence diagram of Register To Website

Design rationale: The system accepts the registration request first and then expects registration details from the unregistered user. If an unregistered user gives invalid registration details, the system asks for registration details again. If the details are correct, the system adds the user to the system. Finally, it gives a confirmation message to the user, and the registered user can return dashboard by clicking the return dashboard button.

Add Patient

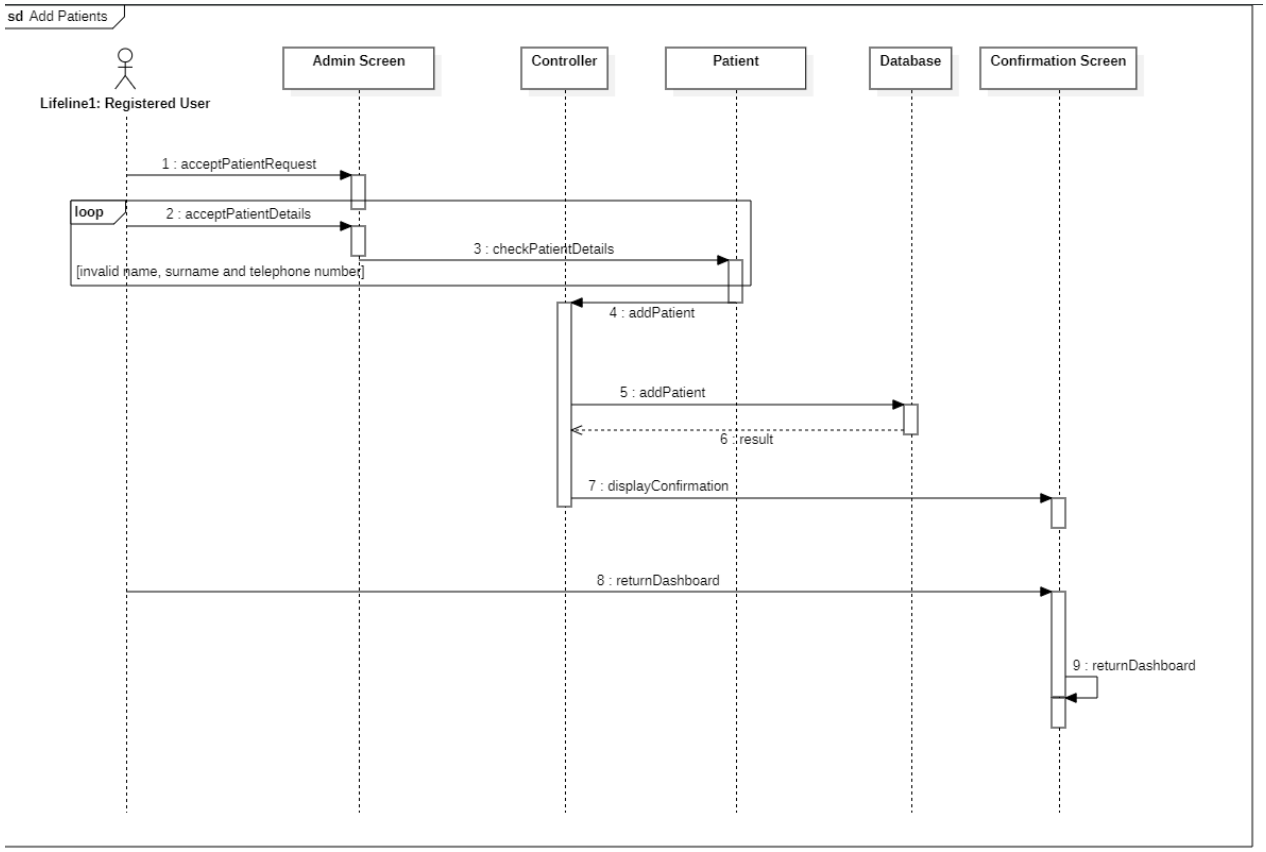


Figure 33: Sequence diagram of Add Patient

Design rationale: First, the system accepts the patient request.Then, The registered user enters the patient's details the user wants to add, and the system checks whether the patient's information has been entered correctly. If valid, it adds the patient to the database and displays a confirmation message, and the registered user can return to the dashboard by clicking the return dashboard button.

Search Patient

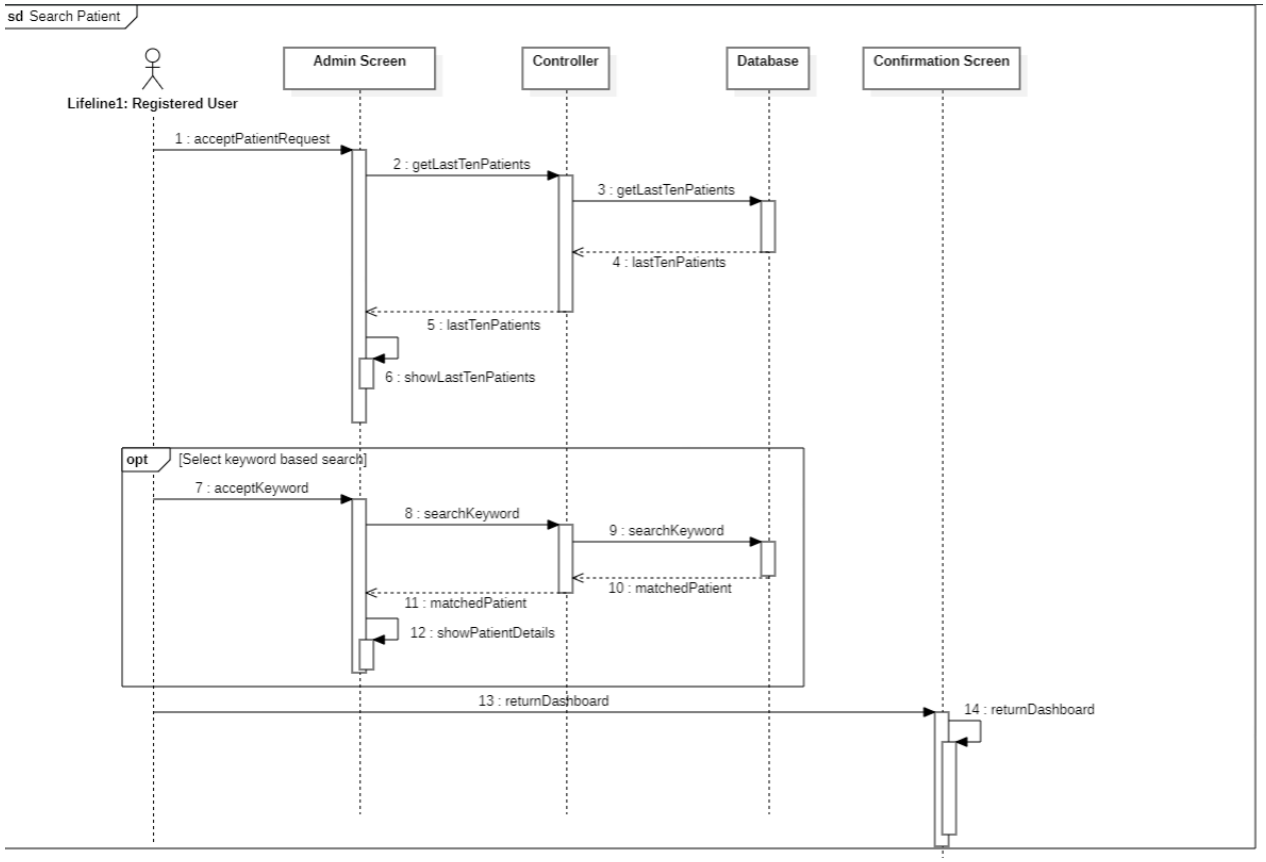


Figure 34: Sequence diagram of Search Patient

Design rationale: The system accepts the patient request and then shows the last ten patients to the registered user. Also, the registered user searches for a patient by the keyword, and the system shows the list of matching patients to the registered user. The registered user can return to the dashboard with the return dashboard button.

Delete Patient

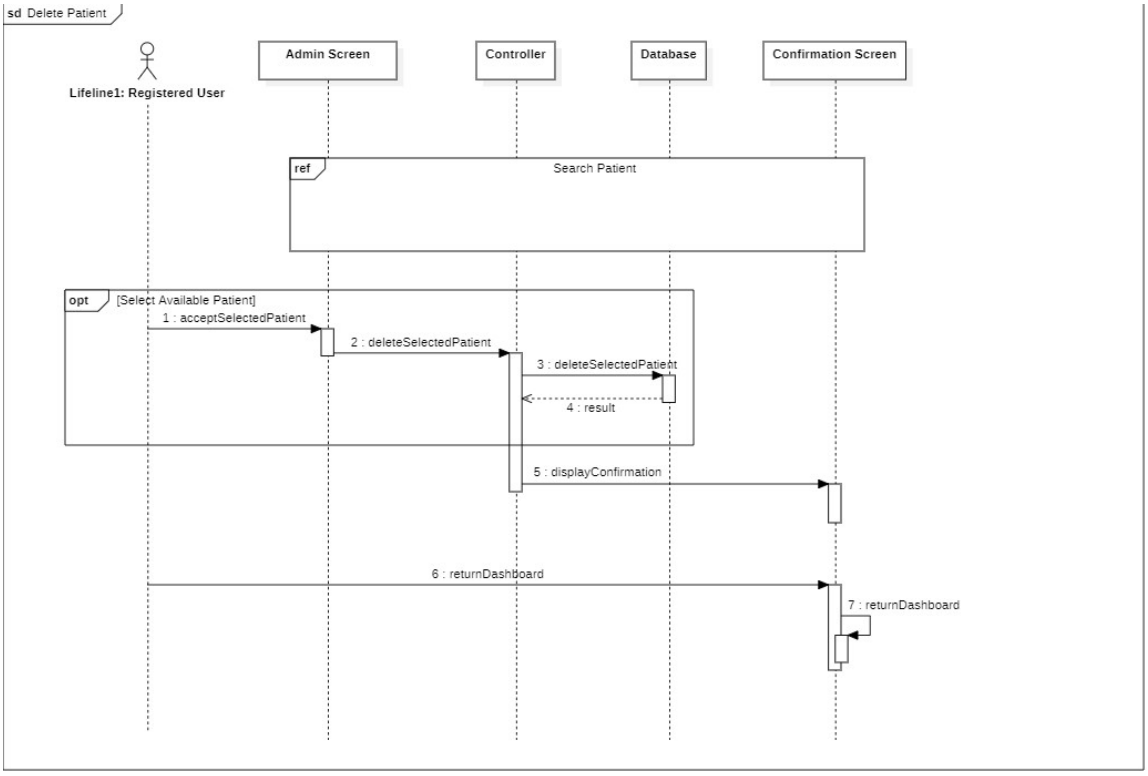


Figure 35: Sequence diagram of Delete Patient

Design rationale: First, the system accepts patient request, then it shows the latest ten patients to the registered user, and the registered user can choose one of these patients. In addition, the Registered user can search with keywords and select one of the patients if the user wishes. This process follows the order summarized in the 'Search Patient' sequence diagram, as shown in the relevant sequence diagram. After the patient is selected, the system deletes this patient from the database, then shows the registered user an information message about whether the patient has been deleted or not, and the registered user can return to the dashboard with the return dashboard button.

Create an Autism Test

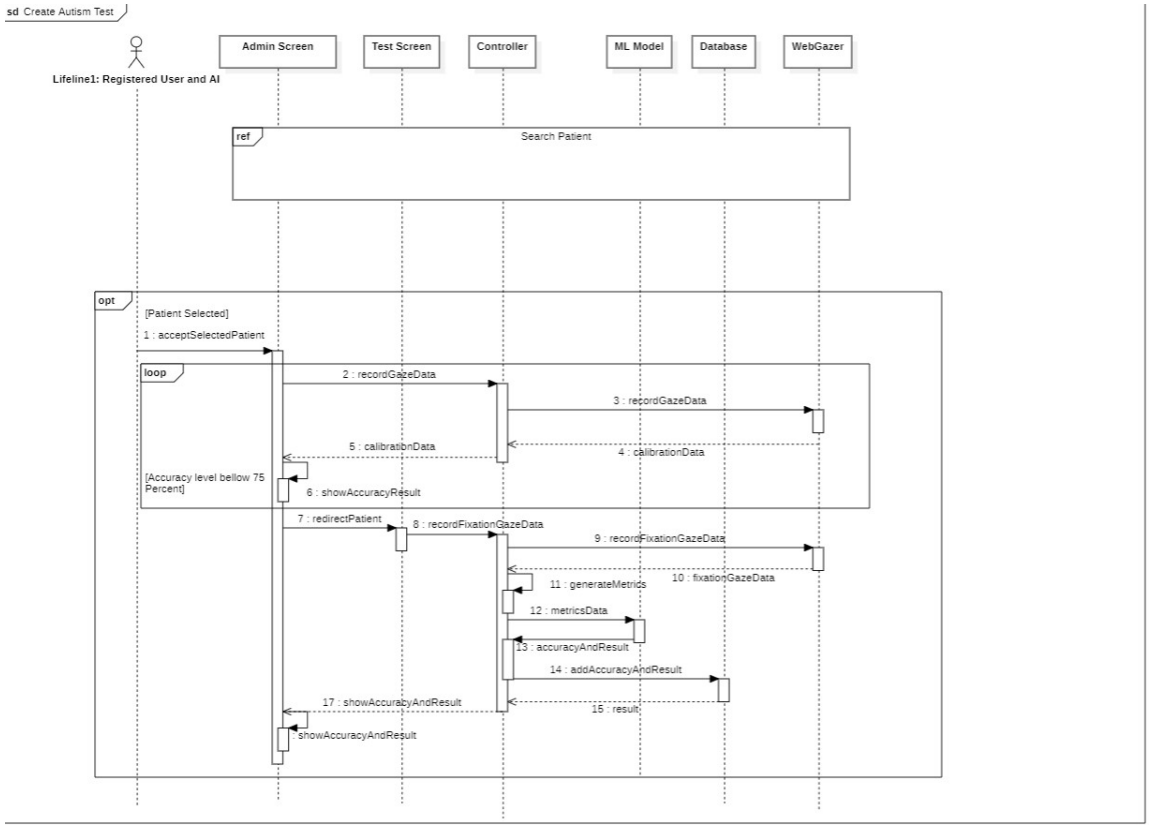


Figure 36: Sequence diagram of Create Autism Test

Design rationale: First, the system accepts patient request, then it shows the latest ten patients to the registered user, and the registered user can choose one of these patients. In addition, the Registered user can search with keywords and select one of the patients if the user wishes. This process follows the order summarized in the 'Search Patient' sequence diagram, as shown in the relevant sequence diagram. A calibration test is started once the selected patient request is accepted, and gaze data is recorded by webGazer for the patient. If the patient cannot complete the calibration test with an accuracy of 75 or above, the test does not start, and the calibration test is performed again. If the patient completes the calibration test with an accuracy of 75 or above, the patient is directed to the test screen. While the patient is performing the test, fixation gaze data are recorded, and when the test is completed, the necessary metrics are generated with these data and sent to the ML model via the controller. ML model returns accuracy and result. This accuracy and result are saved in the database and then returned to the registered user with the message accuracy and result added to the database.

4.2.3 Data Flow Diagrams

Context-level

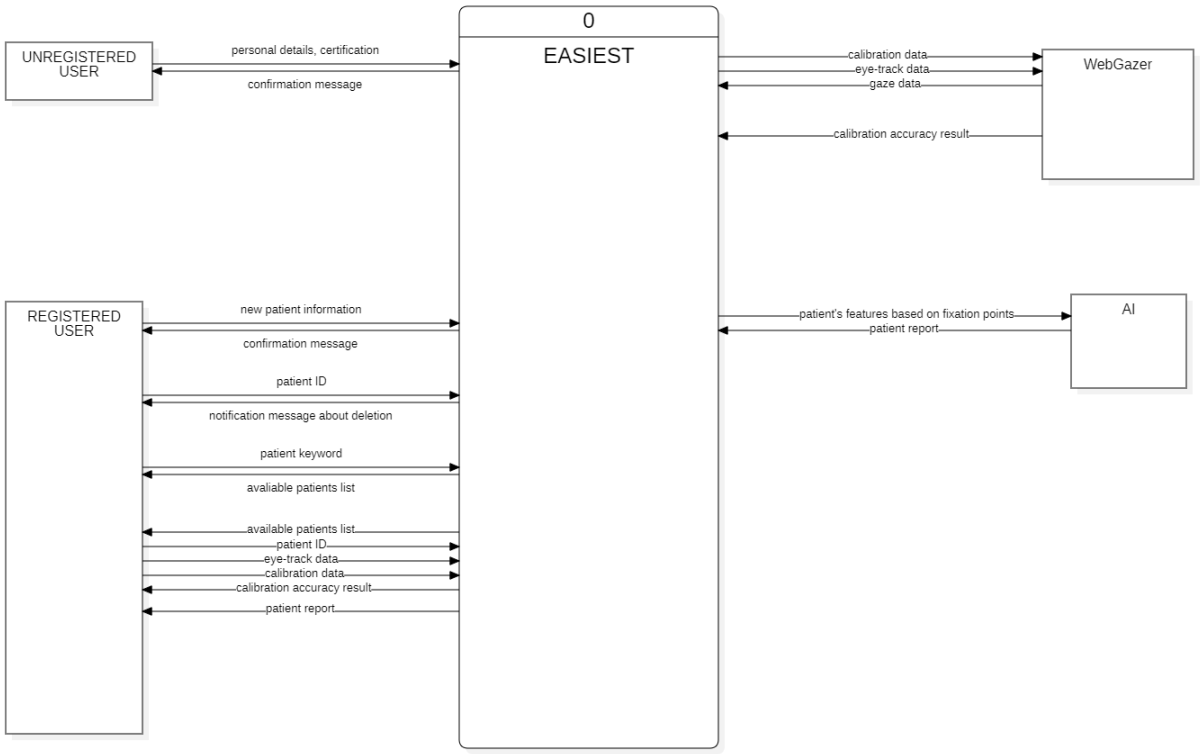


Figure 37: Data Flow diagram of Context-level

Design Rationale: This is the context-level data flow diagram of our project. We implemented our main processes and our entities here. All of the data sent to the main program, which is EASIEST is shown above. Also, we kept close the related data flows which are operated in the same process.

Level-0

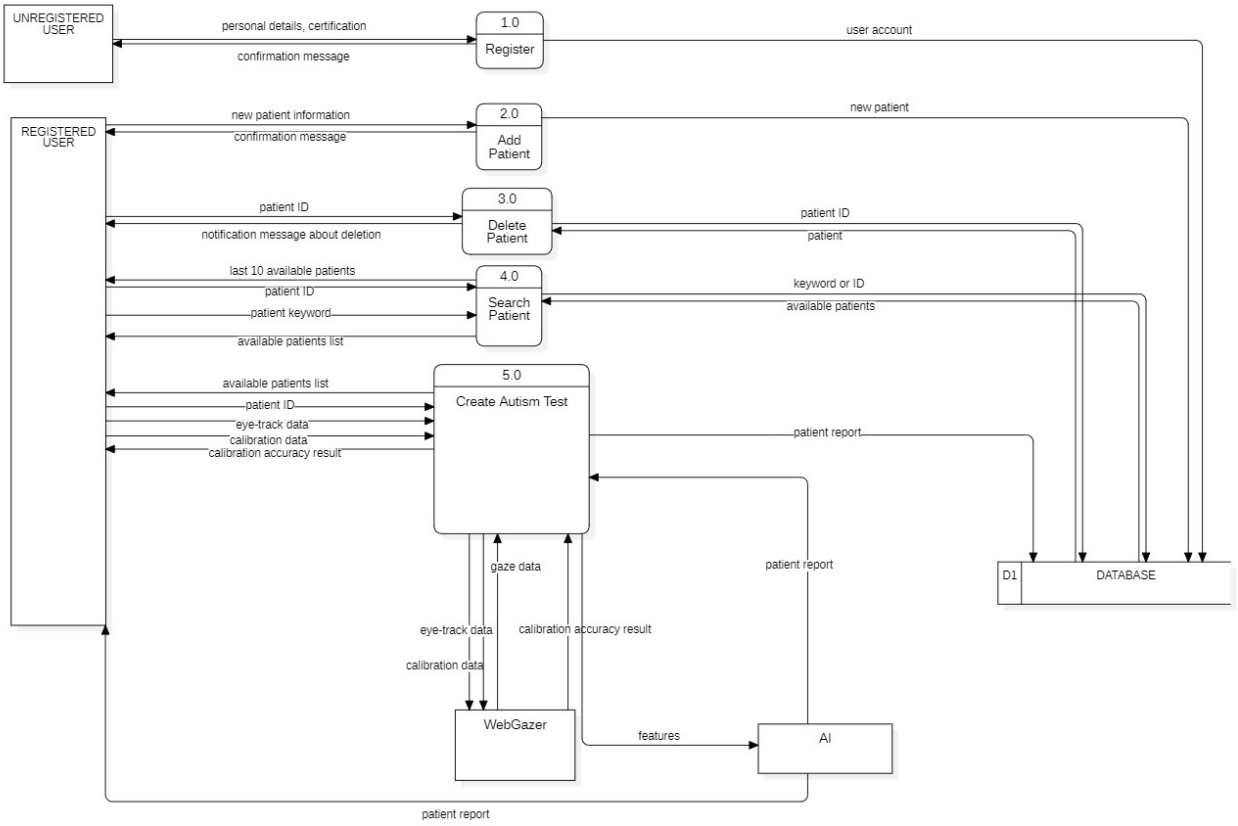


Figure 38: Data Flow diagram of Level-0

Design Rationale: This is our all-system level-0 DFD. We demonstrated all our processes (also relations between them), entities, and the website database which stores accounts, patients, and patient reports.

Level-1 Register

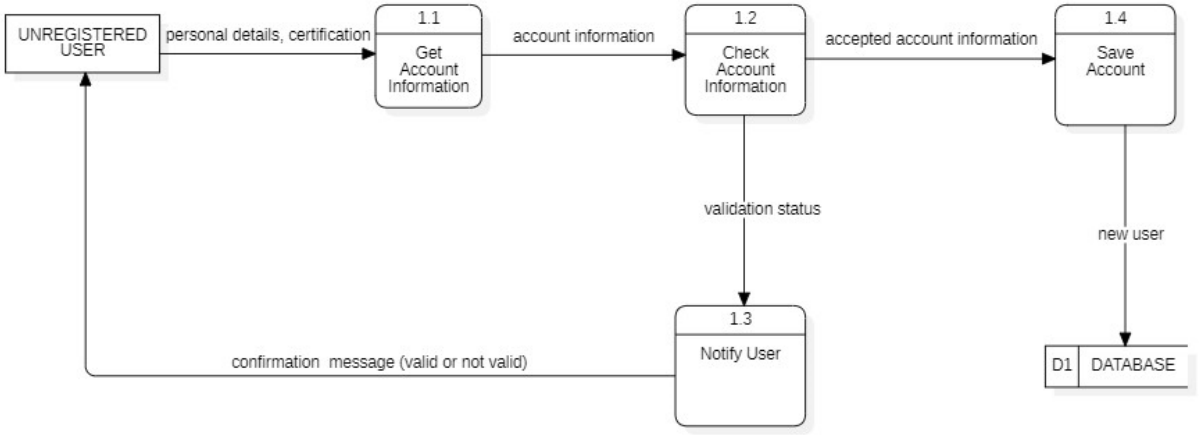


Figure 39: Data Flow diagram of Level-1 Register

Design Rationale: Unregistered users first provide details about the account and process 1.1 takes them. Then, it sends the account information to the 1.2 process which is responsible for checking the account information. Then, the validation status (valid / not valid) is sent to the process 1.3. After, process 1.3 notifies the user whether the account was successfully created or not. If the account successfully created, then 1.2 also, sends the account information to process 1.4. Process 1.4 stores the new user to the website database.

Level-1 Add Patient

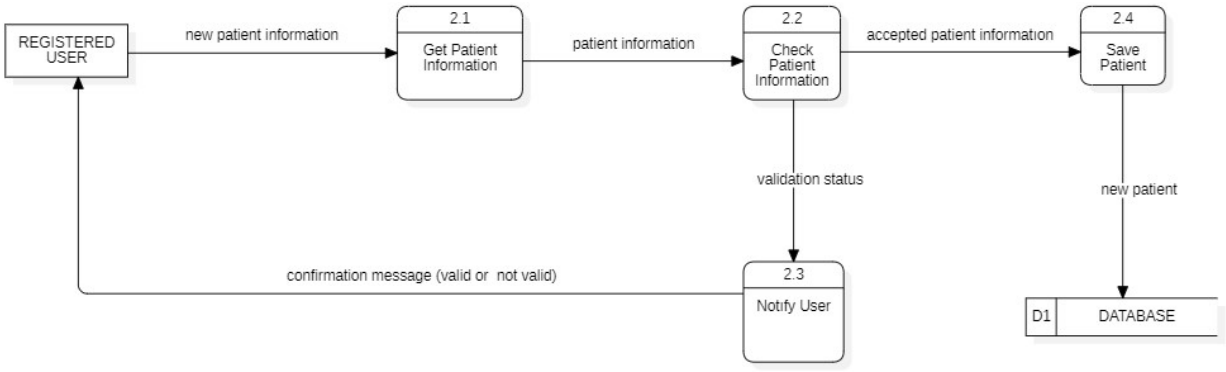


Figure 40: Data Flow diagram of Level-1 Add Patient

Design Rationale: Registered users first provide details about the patient and process 2.1

takes them. Then sends the patient information to the 2.2 process which is responsible for checking the account information. Then, the validation status (valid / not valid) is sent to the process 2.3. After, process 2.3 notifies the user whether the patient record was successfully created or not. If the new patient successfully created, then 2.2 also, sends the patient record information to process 2.4. Process 2.4 stores the new patient to the website database Delete Patient is not demonstered as a figure, since this operation is instant and short operation. It's just communication with database

Level-1 Search Patient

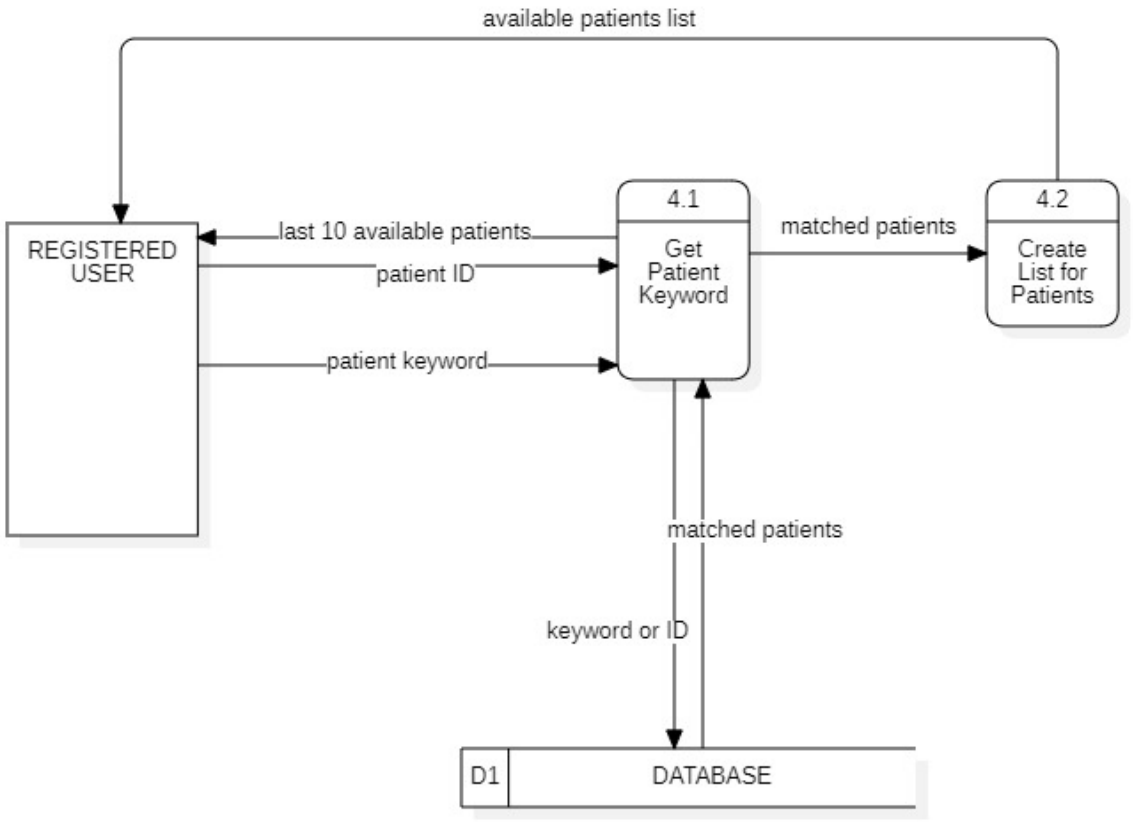


Figure 41: Data Flow diagram of Level-1 Search Patient

Design Rationale: If registered users want to search for any patient, users first need to provide a keyword in the search bar or select from the last 10 available patients. Process 4.1 gets the keyword provided by the user and it contacts with database for matching records. In case of any multiple matching, process 4.1 sends the matched patients to process 4.2 in order to make a list of them. Finally, process 4.2 returns the all-matched patients list to the registered user again. It's important to note that search operation is also plays cruical role for other processes. For example, Search Patient operation is performed before Process 3.0 and Process 5.0. (Actually, Delete Patient operation is just similar as search operation and short operation, so it is not included in the report)

Level-1 Create Autism Test

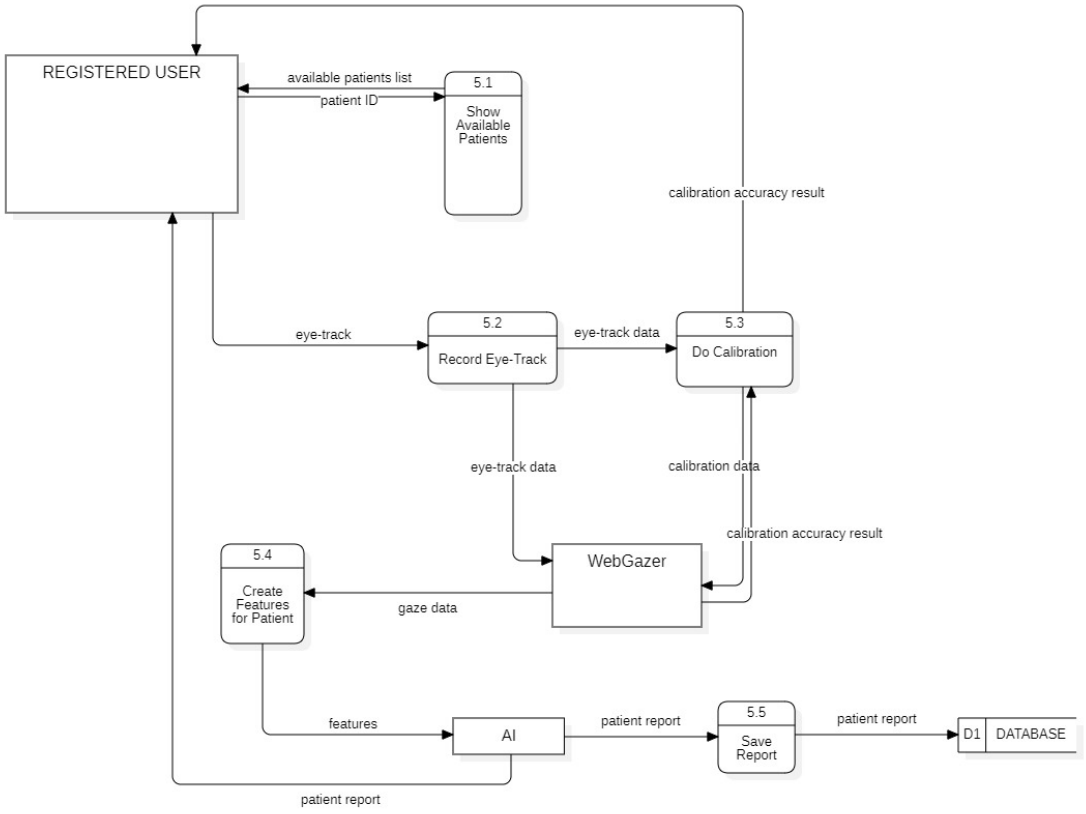


Figure 42: Data Flow diagram of Level-1 Create Autism Test

Design Rationale: In order to create an autism test, registered users first select a patient

which is provided by process 5.1. After starting, process 5.2 records all eye-tracking data and it sends it to process 5.3 for calibration. Process 5.3 returns the calibration accuracy result and displays it to the user. Also, process 5.2 flows the eye-track data to the WebGazer. WebGazer generates gaze data from eye-tracking data and calibration data and sends it to process 5.4 which is responsible for creating features for our machine learning model (shortly AI). AI takes the features and generates a patient report with the features provided. Lastly, process 5.5 takes the report and stores it in the website database.

4.3 Development View and Physical View

4.3.1 Component and Deployment Diagram

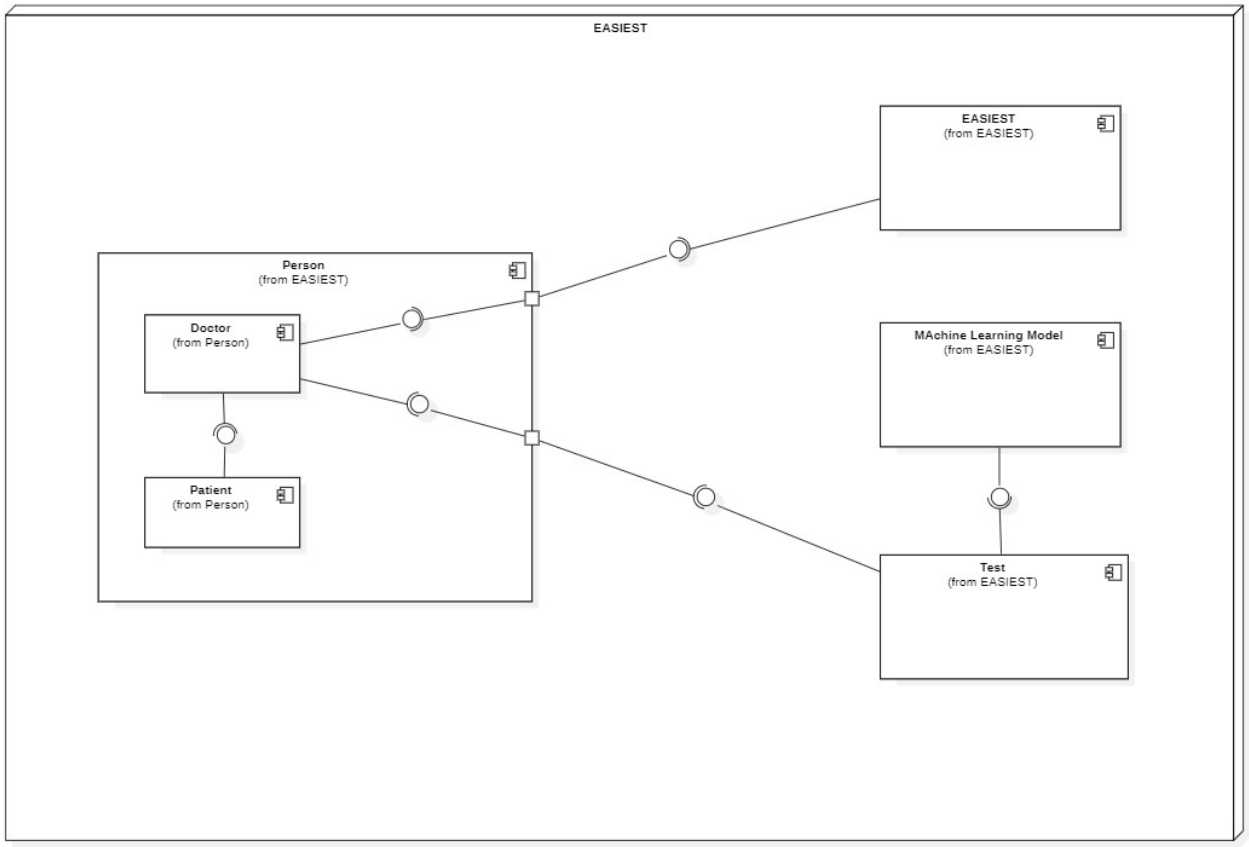


Figure 43: Component and Deployment Diagram

The component diagram below is depicting the organization and dependencies of software components in our system. We have 5 major components with one having 2 subcomponents. The

architecture of the system could be understood from the way the components are connected. As it was mainly designed from the class tables, we can see that most of the components are providing the interfaces to the doctors class, where most of the functionalities are taking place.

The deployment diagram is derived form the component diagram above however as we are not using any external systems, the deployment diagram is identical to the component diagram

5 Software Implementation

5.1 List Of Programming Languages And Frameworks

- **Python:** In our project, we used Python for the backend and used the Flask framework written in Python language. Flask is used to build web applications and APIs and is known for its minimal configuration. We used Flask to design our website.
- **JavaScript:** We used javascript for the dynamic and interactive functionality of our website. We store our JavaScript files in the "static" folder, just like CSS, and we use these scripts in our HTML files.
- **HTML5:** We used HTML5 to create the structure of our website pages. We keep the HTML in the "templates" folder in our project and use these HTML with Flask.
- **CSS:** We used CSS to determine the style and design of our website. We use these CSS' in our HTML files.

5.2 Components

- **Webgazer:**

  It is a JavaScript library that tracks users' eye movements and provides eye tracking in web applications. This library allows users to interact with web pages using their eyes. It is specifically used to improve user interfaces, improve user experience and make web pages more accessible.

  WebGazer runs on the browser and tracks users' eye movements using a webcam. It detects when users look at specific points on a web page and can determine where they focus or where they look and how long they look.

  To use WebGazer in our web application, we added webgazer's "webgazer.js" JavaScript file to the HTML page. In this way, WebGazer starts tracking users' eye movements and allows us to obtain users' eye data.

5.3 Database Management

In our project, which is built using the Flask web framework in Python, we employed PostgreSQL as our relational database management system. PostgreSQL is an open-source object-relational database system known for its reliability and robust features. It provides support for various data types, indexing techniques, and advanced querying capabilities, making it suitable for handling complex data structures and large datasets.

We chose PostgreSQL for its ACID compliance, which ensures data integrity and consistency, which is crucial for applications with transactional requirements. Additionally, PostgreSQL offers extensibility through user-defined functions, allowing us to customize database functionality to suit our project's specific needs.

To interact with PostgreSQL within our Flask application, we utilized the Flask-SQLAlchemy extension. Flask-SQLAlchemy simplifies database integration by providing an Object-Relational Mapping layer on top of SQLAlchemy, a powerful SQL toolkit, and Object-Relational Mapper for Python. This integration lets us define database models using Python classes and seamlessly perform database operations within our Flask routes and views.

By integrating PostgreSQL with Flask-SQLAlchemy into our software implementation, we established a reliable and efficient data storage solution, enabling secure management and analysis of user data for our Flask-based web application.

5.4 Libraries

- **Pandas:** It is a Python library used for data analysis and data manipulation. Provides data structures and data analysis tools. It is used to process, transform and analyze data using Pandas DataFrames. In our project, we used Pandas to manipulate and edit the data from the eye tracker.

- **NumPy:** A basic library used for scientific computing in Python. Supports multidimensional arrays and mathematical functions. NumPy is widely used to perform fast and efficient calculations such as vector and matrix operations. In our project, We used the NumPy library for the calculations of the idt and ivt algorithms.

- **SQLAlchemy**: It is a library for interacting with SQL-based relational databases in Python. Using the Object-Relational Mapping (ORM) method, it allows you to associate Python objects with database tables and perform database operations more easily and flexibly. In our project, we used SQLAlchemy to make our database easier and more structured.

- **WTForms:** It is a Python library used to create and validate web forms. It can be used with web frameworks such as Flask and Django. WTForms allows you to define form fields, validate user input, and process form data. In this way, it allows users to enter data into your web applications and process this data securely. In our application, we used the WTForms library to manage user transactions more easily.

- **Selenium:** It is a powerful tool for automating web browsers, allowing us to perform automated testing of our website. We used Selenium to simulate user interactions with our website,

such as clicking buttons, filling out forms, and navigating between pages. This ensured that our website functions as expected in various scenarios.

- **Pytest:** It is a framework that makes it easy to write simple and scalable test cases for Python code. We used Pytest to write unit tests and integration tests for our application, ensuring that all components function correctly and that new code changes do not introduce any regressions.

6 Software Testing

6.1 Unit Testing

6.1.1 Test Procedure

Our testing approach involves verifying the correctness of our method for processing eye tracker data. For example, we conducted unit tests to ensure that our fixation algorithm correctly merges fixations and that our generated features are accurately produced. To achieve this, we utilized the pytest library in Python. Kerem and Barış are responsible for creating, executing, and analyzing these tests and resolving any potential issues. To ensure the output matched our observational results, we tested feature generation scripts from eye tracker data and the fixation algorithm using pre-determined data based on our observations.

```
def test_fixation_count():
    pairs = [[1000, 1000, 100], [1010, 1050, 300], [2000, 2000, 500], [2010, 2050, 700], [2090, 3000, 900]]
    fixation_total = idt(pairs, 20, 100)

    assert len(fixation_total) == 3
```

Figure 44: IDT Algorithm fixation generation test

Our main objective with this unit test function is to make a fixation count prediction by analyzing the gaze data pairs we have manually generated. Then, we input the mock data we created into the IDT algorithm used in EASIEST to check if the predicted value matches the actual value output by the IDT. In this example, our prediction was a total of 3 fixations. We expected the first two arrays and the third and fourth arrays in the pairs array to each form a fixation. The last array was expected to form a fixation on its own without merging with any other array, and this was how we made our prediction. As you will see in the continuation of the report, the test log shows that the test_fixation_count resulted in PASSED

```
def test_element_find():
    pairs = [[579, 112, 100], [590, 112, 300], [600, 112, 500]]
    fixation_pairs = idt(pairs, 20, 100)

    media_id_dictionary = {-1: "None",
                           0: "Apple",
                           1: "AVG",
                           2: "Babylon",
                           3: "BBC",
                           4: "GoDaddy",
                           5: "Yahoo"}

    for fixation in fixation_pairs:
        element = screen_find_element((fixation[0], fixation[1]), 1, media_id_dictionary[0])

        assert element == "F"
```

Figure 45: Finding Screen element test

In this unit test function, our objective is to test if the fixation, whose coordinates we provided, correctly identifies the web page element it is within. For this, we created a triplet pair that formed a fixation point. According to our calculations, the fixation we created is within the 'F' element on the Apple page, and when we tested it, the test log showed that this test also passed successfully.

```python
def test_revisit_element():
    pairs = [[952.1624603179764, 53.13116775479648, 171.60000002384186], [958.8025864615347, 63.39863335360877, 199.5]
    fixation_pairs = idt(pairs, 20, 100)

    media_id_dictionary = {-1: "None",
                           0: "Apple",
                           1: "AVG",
                           2: "Babylon",
                           3: "BBC",
                           4: "GoDaddy",
                           5: "Yahoo"}
    data_eye_track = []
    id_counter = 1
    for fixation in fixation_pairs:
        element = screen_find_element((fixation[0], fixation[1]), 1, media_id_dictionary[0])

        row = [id_counter, fixation[0], fixation[1], fixation[2], fixation[3], fixation[4], element]

        data_eye_track.append(row)  # appending the row to my array
        id_counter += 1  # incrementing the id counter

    computed_metrics_dictionary = compute_metrics(data_eye_track, media_id_dictionary[0])
    assert computed_metrics_dictionary['C'].index(2) == 2
```

Figure 46: Revisit test for a specific segment

To obtain the sample eye track values for this unit test, we ran Webgazer ourselves, and after making a pattern between the segments using Webgazer, first "C", then "F", then "C" again, and we stopped the Webgazer test and produced x,y,time data. We tested the for the "compute_metrics" function that we wrote, and as we expected, the number of revisits for the C element was 2 .Therefore, the test passed successfully.

```python
def test_first_time_looked():
    pairs = [[949.082770891413, 111.38643759233538, 0.0], [931.8849970568288, 136.97542577007718, 98.69999998807907], [8

    fixation_pairs = idt(pairs, 20, 100)
    data_eye_track = []
    id_counter = 1

    media_id_dictionary = {-1: "None",
                           0: "Apple",
                           1: "AVG",
                           2: "Babylon",
                           3: "BBC",
                           4: "GoDaddy",
                           5: "Yahoo"}

    for fixation in fixation_pairs:
        element = screen_find_element((fixation[0], fixation[1]), 1, media_id_dictionary[0])
        row = [id_counter, fixation[0], fixation[1], fixation[2], fixation[3], fixation[4], element]
        data_eye_track.append(row)
        id_counter += 1

    computed_metrics_dictionary = compute_metrics(data_eye_track, media_id_dictionary[0])

    # First time looked test
    assert round(computed_metrics_dictionary['C'].index(1), 1) == 2
```

Figure 47: First time looked test for a specific segment

Using the method described in Figure 47, we again obtained sample data for x, y, ,time, but this time we created a different pattern. Initially, for the "C" element, we manually timed with a stopwatch and looked at the "C" element for approximately "2.34" seconds. Then, we tested whether the "compute_metrics" function we wrote correctly measured the first time looked in seconds, and as expected, it was correct (we rounded the seconds because there may be small margins of error).

```python
def test_total_time_viewed():
    pairs = [[949.082770891413, 111.38643759233538, 0.0], [931.8849970568288, 136.97542577007718, 98.69999998807907], [878.40877

    fixation_pairs = idt(pairs, 20, 100)
    data_eye_track = []
    id_counter = 1

    media_id_dictionary = {-1: "None",
                           0: "Apple",
                           1: "AVG",
                           2: "Babylon",
                           3: "BBC",
                           4: "GoDaddy",
                           5: "Yahoo"}

    for fixation in fixation_pairs:
        element = screen_find_element((fixation[0], fixation[1]), 1, media_id_dictionary[0])
        row = [id_counter, fixation[0], fixation[1], fixation[2], fixation[3], fixation[4], element]
        data_eye_track.append(row)
        id_counter += 1

    computed_metrics_dictionary = compute_metrics(data_eye_track, media_id_dictionary[0])

    total_time = 0.0
    for element_key in computed_metrics_dictionary.keys():
        duration = computed_metrics_dictionary[element_key][0]
        total_time += duration

    for element_key in computed_metrics_dictionary.keys():
        duration = round(computed_metrics_dictionary[element_key][0], 3)

    try:
        time_viewed = round((duration / total_time) * 100, 3)
        # Time viewed test
        assert (6 > time_viewed) and (time_viewed > 4)
    except ZeroDivisionError:
        time_viewed = 0
```

Figure 48: Total time viewed test for a specific segment

We used the method we explained in Figure 48 and the same x,y,time data produced. This time, we kept a stopwatch again to calculate the total time for the "C" element and recorded the time taken for "C". This time was "5.34" seconds. When we tested it, it was accurate as expected (since the margins of error may be small, it should be between the 6th and 4th seconds).

6.1.2 Test cases

ID	Description	Steps	Test Data	Pre-condition	Expected Output	Passed/Failed
1	IDT Algorithm fixation generation test	• Predict fixation count using manually generated gaze data pairs. • Test the IDT algorithm with mock data. • Verify if predicted fixation count matches actual output.	Gaze data Pairs: (x:1000, y:1000, time:100), (x:1010, y:1050, time:300), (x:2000, y:2000, time:500), (x:2010, y:2050, time:700), (x:2090, y:3000, time:900)	No precondition	Total of 3 fixations predicted	Passed
2	Finding Screen element test	• Test if fixation correctly identifies the web page element it is within. • Create a triplet pair forming a fixation point. • Check if fixation is within specified web page element.	Gaze data pair for Apple page: (x:589, y:112, time:300)	There must be preformed fixation gaze data	Fixation is within specified web page element	Passed

3	Revisit test for a specific segment	• Test 'compute_metrics' function to calculate revisits for a specific segment. • Obtain sample eye track values from Webgazer. • Check if number of revisits for segment is as expected.	Fixation and element values: [id:11, x:156.38, y:216.28, time:3.22, total:0.11, count:5, element:C] .. [id:16, x:601.54, y:184.70, time:5.18, total:0.10, count:4, F] .. [id:19, x:510.88, y:222.22, time:5.69, total:0.16, count:7, element:C]	There must be data that has been fixated and those elements are known	Passed	
4	First time looked test for a specific segment	• Test if 'compute_metrics' function correctly measures first time looked in seconds for a specific segment. • Obtain sample x, y, time data. • Check if first time looked measurement is accurate.	Fixation and element values: [id:1, x:1067.71 y:799.47, time:0.033, total:0.1, count:1, J] .. [id:9, x:419.71 y:297.47, time:2.35, total:0.17, count:5, C]	There must be data that has been fixated and those elements are known	First time looked measurement is accurate (approximately 2.34 seconds)	Passed

5	Total time viewed test for a specific segment	• Test if 'compute_metrics' function correctly calculates total time viewed in seconds for a specific segment. • Obtain sample x, y, time data. • Check if total time viewed measurement is accurate.	Fixation and element values: .. [id:9, x:419.71 y:297.47, time:2.35, total:0.17, count:5, C] .. [id:30, x:444.12 y:310.15, time:2.35, total:7.69, count:3, C]	There must be data that has been fixated and those elements are known	Total time viewed measurement is accurate (approximately 5.34 seconds)	Passed

6.1.3 Test Results

Tests that have been conducted by processing eye tracker data have passed successfully. Each test case was executed, and the expected outcomes were observed without failure. These tests, which were focused on ensuring that our fixation algorithm correctly merges fixations and that our other generated features are accurately produced, passed all assertions without issues. Therefore, we can confirm that the application's eye-tracking data process is functioning as expected.

6.1.4 Test Log

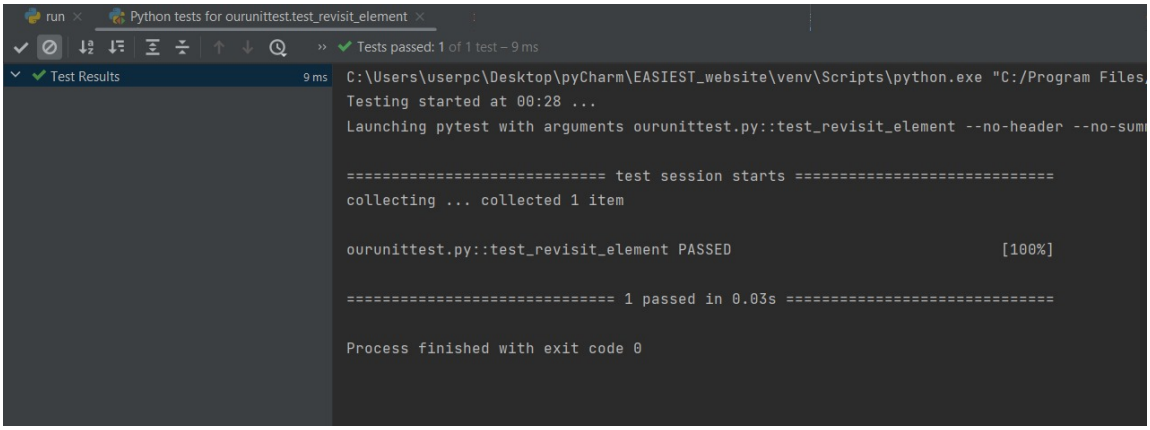


Figure 49: IDT Algorithm fixation generation test log

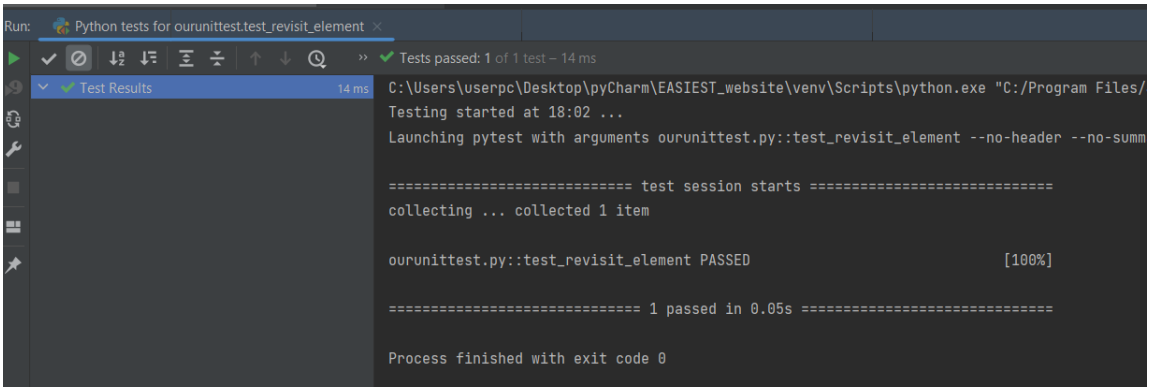


Figure 50: Revisit test for a specific segment log

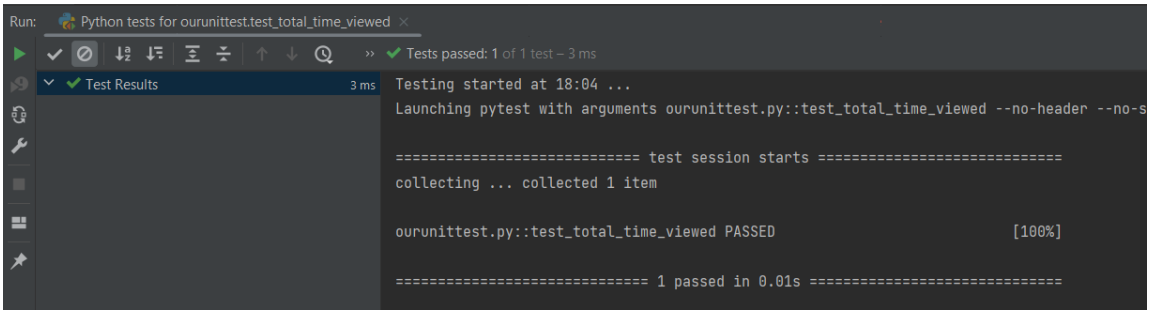


Figure 51: Total time viewed test for a specific segment log

6.2 Integration Testing

6.2.1 Test Procedure

In the integration testing, we implemented five different scenarios, whereby, using the Selenium framework, we would simulate human interactions with our website. We decided to go with the Selenium framework because it provides automated execution of the different scenarios. Simply put, it finds the HTML tags that were used during the design of the webpage, further has functionalities to fill various inputs and send different requests to the end of the application.

```python
def Registration_test():
    driver = webdriver.Chrome()
    try:
        # Open the webpage with the form
        driver.get("http://127.0.0.1:5000/register")

        # Fill in the form fields
        driver.find_element(by=By.ID, value="Name").send_keys("Kaya")
        driver.find_element(by=By.ID, value="Surname").send_keys("Daniilovna")
        driver.find_element(by=By.ID, value="Username").send_keys("KayaTheBest")
        driver.find_element(by=By.ID, value="Email").send_keys("kaya@gmail.com")
        driver.find_element(by=By.ID, value="Tel").send_keys("12341234")
        driver.find_element(by=By.ID, value="Address").send_keys("odtu")
        driver.find_element(by=By.ID, value="Hospital").send_keys("odtu_hos")
        driver.find_element(by=By.ID, value="Password").send_keys("1234")
        driver.find_element(by=By.ID, value="Confirm_password").send_keys("1234")

        # Submit the form
        driver.find_element(by=By.ID, value="Submit").submit()
        with app.app_context():
            user = User.query.filter_by(email='kaya@gmail.com').first()
            if user:
                print('\nRegistration Test Passed\n')
    except Exception as e:
        print(e)
    # Close the browser
    driver.quit()
```

Figure 52: Registration Selenium Script

With the use of this framework, we have implemented five scenarios, one of which is the Registration procedure. To implement it, we have used various functionalities of Selenium, such as finding an element of the page with its ID reference in HTML structure. Thus, we have found all the form elements to be filled out further and sent the data we wanted to test as keys to the form inputs as we can see in the figure 52, we used driver.find_element. To submit the form and register, we used the functionality that prompted the Selenium submit button. The button was also found using the HTML element search. Overall, we have conducted five integration tests with Selenium:

1. **Registration:** The code should have been able to go the registration website, fill in the form and then submit it. The test was approved if afterwards a doctor with the supplied information could be found in the database. With this code we aimed to check that the website page could be reached and the doctor could be properly created. Database connectivity was checked as well as the retrieval operation was done directly with the use of database.

2. **Login as Unvalidated Doctor:** The code had to automatically log in to the dashboard of the doctor who was not yet approved by the admin and was supposed to check if the page that was shown after the login had the right content. If the content was identical, the test was approved. With this part we wanted to check that the doctor who was not yet approved could not use the functionalities of the website.

3. **Login to the admin panel:** With this code we aimed to check that the doctor that was just created could be approved by the admin. Here the simulation had to log in into the admin panel and press the 'approve' button of the doctor that was just created. The test was said to pass if later by retrieving the doctor from the database, the status was changed to 'Validated'.

4. **Login to the Validated doctor:** For this case we aimed to check that after the doctor was validated, it was possible to login to the correct dashboard and create a patient for this doctor. The simulation had to fill the form to login in, open the panel for patient creation, further fill the form for the doctor and submit it. The test was marked as passed if firstly the correct dashboard was shown and secondly the patient whose information we supplied could be retrieved from the database with the curing doctor being the one we created earlier.

5. **Updating the address of the doctor:** In this part of testing we aimed to check that after logging in to the validated doctor, the account information could be updated. The simulation had to go to the account page, click the update button, change the value of the address and then click the button again. The test was approved if later, the address of the doctor that was retrieved from the database would be equal to the updated information we supplied with simulation.

6.2.2 Test Cases

ID	Description	Steps	Test Data	Pre-condition	Expected Output	Passed/Failed
1	Registration Check	Go to the registration page. Fill in the form. Submit the form.	Personal data of the potential doctor: Name: Kaya Surname: Danilovna Username: KayaTheBest Email: kaya@gmail.com Tel: 12341234 Address: odtu Hospital: odtu_hos Password: 1234 Confirm password: 1234	No pre-condition	A new doctor account is created	Passed
2	Login as Unvalidated Doctor Check	Log into the created account of the doctor, and check that the correct dashboard is displayed, where the doctor does not have access to the patient testing and creation.	Doctor login and password information: Email: kaya@gmail.com Password: 1234 Dashboard structure	The doctor with Email: kaya@gmail.com should not yet be validated	A successful login and proper dashboard output.	Passed
3	Login to the admin panel	Log into the admin panel from the Login page. Select the doctor that was created in this test plan before. Validate this doctor.	Admin login and password: Email: admin@gmail.com Password: 1234 Admin dashboard structure Doctor validation update	An admin with Email: admin@gmail.com should be already in the system	A previously created doctor is validated. The admin dashboard is reachable.	Passed
4	Login to the Validated doctor Check	Log into the validated doctor's dashboard. Open the panel for patient creation. Fill in the form for the patient. Submit the information of the patient to the system.	Patient information: Email: kaya@gmail.com Password: 1234 Patient Name: Kaya Patient Surname: Daniilovna Patient Tel: 12322175	The doctor with Email: kaya@gmail.com should be validated and in the system.	The doctor could log into the correct dashboard. A doctor can reach the form for creating a patient. A new patient is created.	Passed

5	Updating the address of the doctor	Log into the validated doctor's dashboard. Go to the account webpage. Update the address information.	Doctor's personal information: Email: kaya@gmail.com Password: 1234 New address: Moscow	The doctor with Email: kaya@gmail.com should be in the system	The doctor's information is updated in the database.	Passed

6.2.3 Test Results

All five integration test cases have passed, demonstrating the robustness of the website's criti- cal functionalities. The registration test effectively created a new doctor account, confirmed by the doctor's information in the database. Subsequently, the login test for unvalidated doctors demonstrated proper access control by restricting access to certain features. Moving to ad- ministrative tasks, logging in as an admin and validating a doctor proved successful, with the doctor's status updated in the database as expected. Upon validation, the test ensured that validated doctors could access the correct dashboard and create patients, providing seamless functionality. Lastly, the test for updating doctor information validated that account details could be modified, with changes accurately reflected in the database. Overall, the integration testing process has confirmed the website's reliability in handling critical user interactions.

6.2.4 Test Log

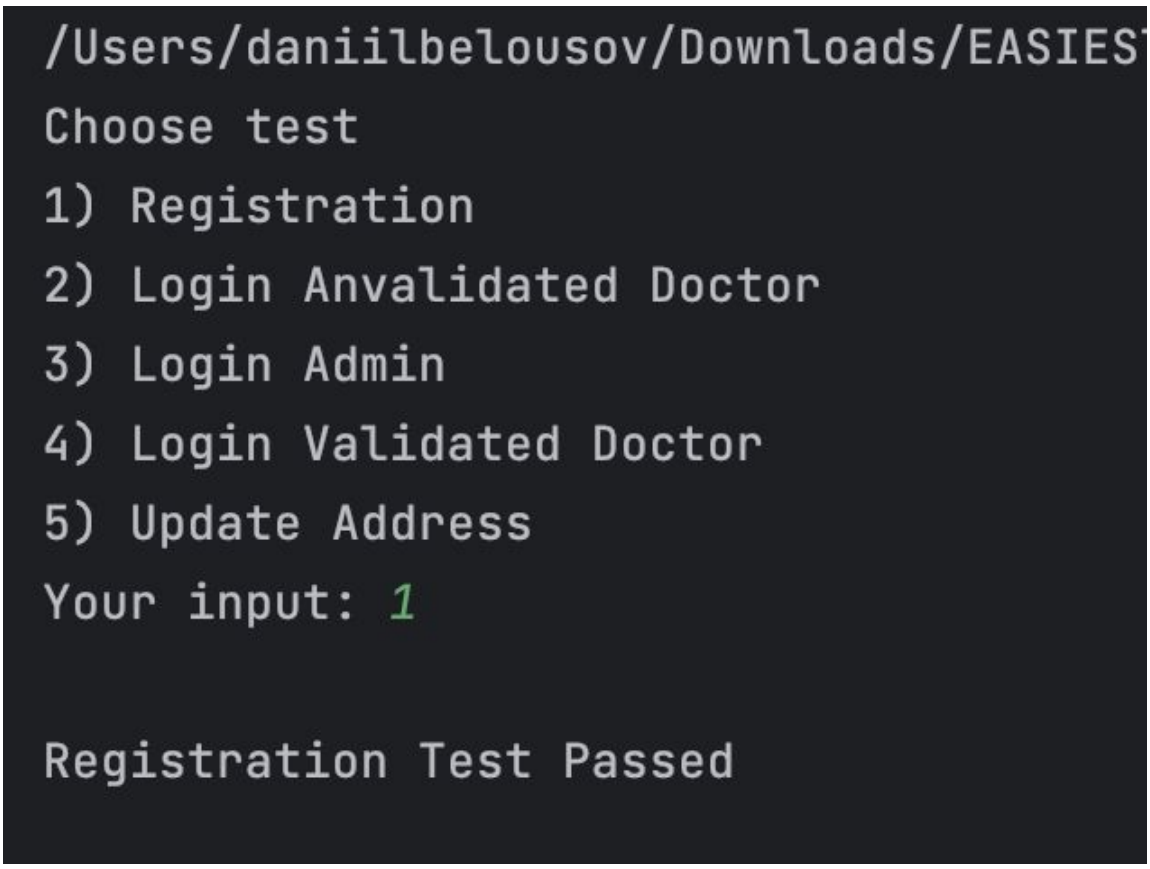


Figure 53: Selenium tests panel and registration test result

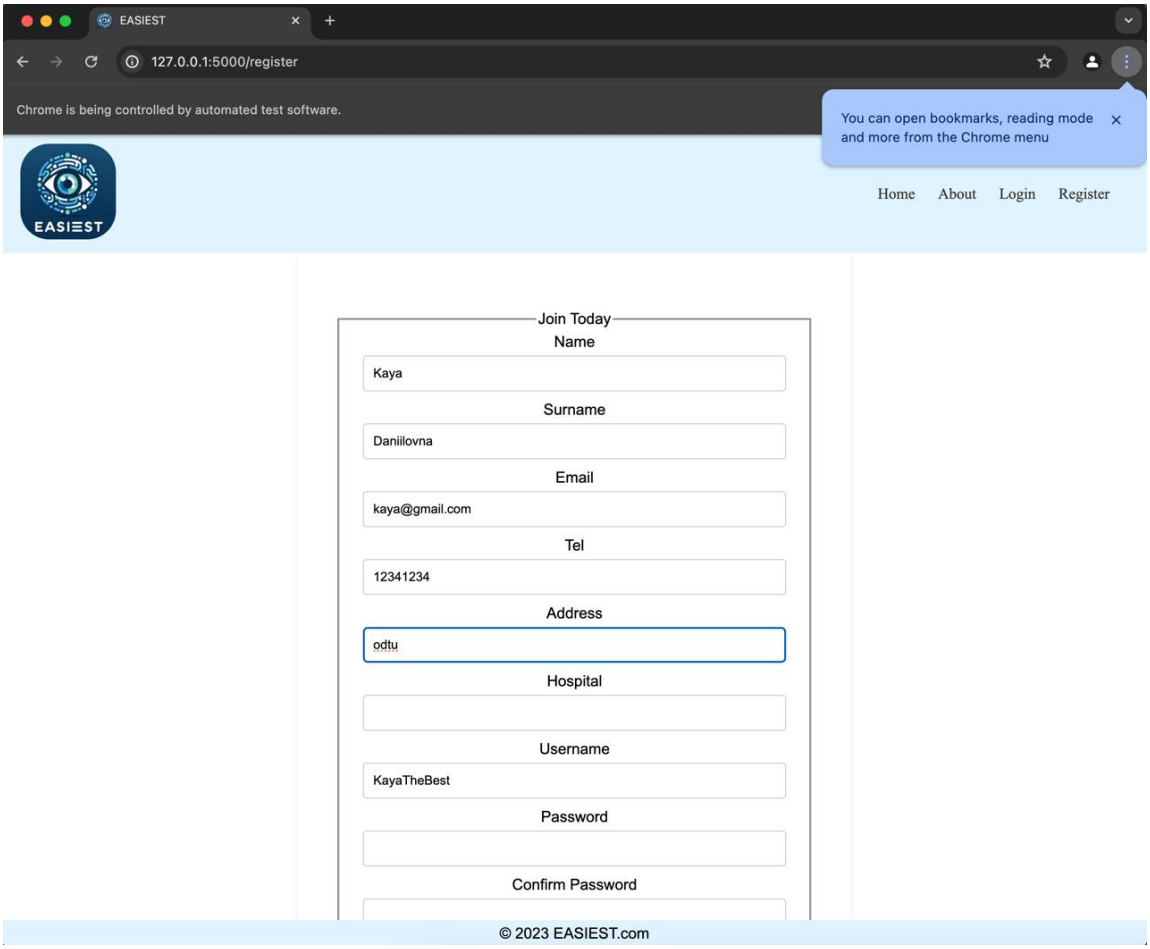


Figure 54: Selenium registration example

6.3 System Testing

6.3.1 Test Procedure

In system testing, since we were simulating a real situation, our procedure was to complete these scenarios in a role-play format. After explaining the scenarios we prepared to the individuals who would enact them, we left them in their natural environments and asked them to complete the scenarios without any intervention. Barış wrote the first scenario. Kerem played the role of the doctor, and the patient was a friend outside our team. This way, we could simulate a real situation because the doctor knew how to use the website but was patient genuinely just following the doctor's instructions. Furkan wrote the second scenario, and this time Daniil played the role of the doctor again, while the patient was another friend outside our team. For these role-play based scenarios,

we chose a quiet environment with good lighting, resembling a doctor's office.

6.3.2 Test Scenarios

In the first scenario, the doctor will register a patient in the system and perform a test on the patient. For this scenario, the doctor will register a new patient into the system and then immediately subject this patient to a test. When the patient's test is finished, the doctor will go to the dashboard, find the patient who underwent the test, and then view the results, including the accuracy and ASD/Non-ASD outcomes from two different models, in the system. This will complete the first scenario.

In the second scenario, the doctor will search for and find a patient already registered in the system in first scenario, from the dashboard. Then, the doctor will take to new test to this patient. After successfully conducting and completing the test step by step, similar to the first scenario, when the patient's test is finished, the doctor checks the test history of that patient from the dashboard. After successfully viewing all the results of both the first and second tests in the system, the second scenario will be completed.

6.3.3 Test Results

In both scenarios, these important activities were successfully completed without any problems occurring.

First Scenario	Doctor creates a new patient	✓
	Doctor tests to this patient	✓
	Doctor views the results	✓
Second Scenario	Doctor searches and finds the patient created in first scenario	✓
	Doctor creates a new test to this patient	✓
	Doctor tests to this patient	✓
	The doctor checks the test history of that patient	✓
	Doctor views all the test results of this patient in first and second scenario	✓

Table 6: Scenarios and Tasks

In this table, scenarios and actions performed in each scenario are listed. Successful steps are indicated with a checkmark

6.3.4 Test Log

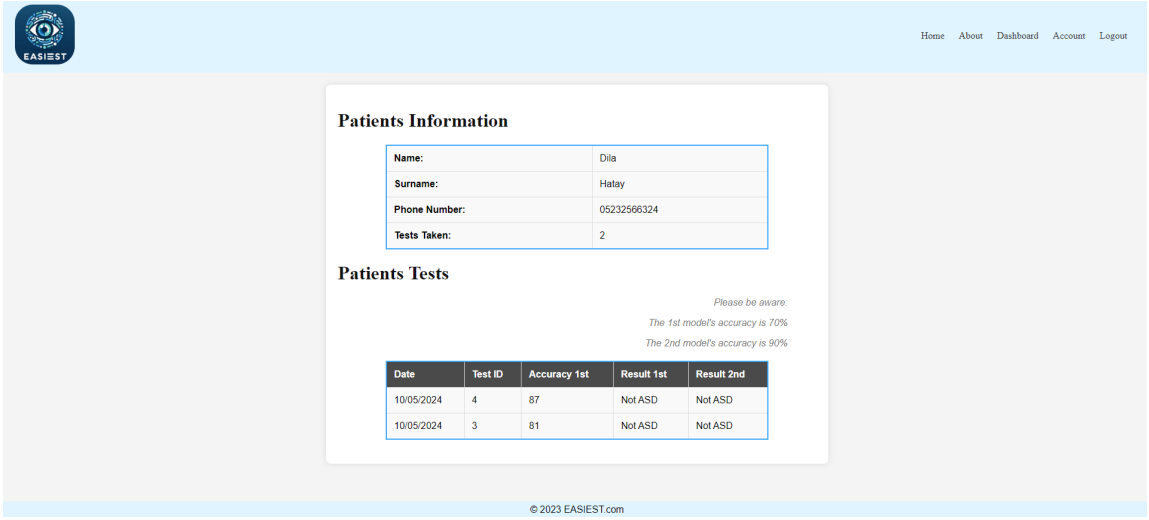


Figure 55: History of the Patient

As seen in the screenshots we obtained from our website, the results of both tests have been successfully generated for the user representing the patient role, named Dila Hatay, and displayed on the website effectively.

6.4 Hardware Testing

6.4.1 Test Procedure

Screen Size Variation Testing: Daniil manages the Screen Size Variation Testing, which aims to assess the impact of different screen sizes on eye-tracking accuracy. In this process, a controlled testing environment with consistent lighting conditions is set up. Screens of various sizes, ranging from 13 to 27 inches, are utilized, and calibration patterns are displayed on each screen to accurately calibrate the eye-tracking system. Daniil performs calibration tests, and the calibration results across different screen sizes are compared to identify any trends or optimal sizes for the system.

Impact of Lighting Conditions Assessment: Barış is responsible for the Impact of Lighting Conditions Assessment, which evaluates how different lighting conditions affect eye-tracking accuracy. Testing environments with controlled lighting conditions are created, including both natural and artificial setups. The WebGazer system is calibrated in each lighting condition to ensure accuracy. Barış performs eye-tracking tasks under each lighting condition, and collected data is analyzed to assess the impact on accuracy. Insights gained from this analysis help identify differences

or challenges in performance under varying lighting conditions.

Effect of Wearing Glasses Testing: Kerem takes charge of the Effect of Wearing Glasses Testing, which aims to determine the impact of wearing glasses on eye-tracking accuracy. Participants are invited to the testing session, and calibration tests are conducted with WebGazer for each participant, considering whether they wear glasses. Data analysis allows for an assessment of the impact of wearing glasses on accuracy. By comparing the performance between participants wearing glasses and those without glasses, valuable insights are gained into how this factor influences eye-tracking accuracy.

WebCamera Quality Assessment: Furkan oversees the WebCamera Quality Assessment, which evaluates the impact of different camera resolutions on eye-tracking accuracy. Various cameras with resolutions including 480p, 720p, and 1080p are set up for testing. Calibration tests are performed with each camera to ensure accurate measurements. Based on the calibration results, Furkan determines which cameras are compatible with WebGazer and provide optimum performance. This evaluation ensures the selection of the most suitable cameras for the system, guaranteeing optimal eye-tracking accuracy across different resolutions.

6.4.2 Test Cases

Table 7: Test Cases

ID	Description	Steps	Result
1	Calibration accuracy on various screen sizes	• Set up controlled testing environment with consistent lighting conditions. • Display calibration patterns (red dots) on each screen size (13, 15, 16, 17, and 27 inches). • Invite participants to sit in front of each screen and perform calibration tests. • Record calibration results for each screen size.	15, 16, 17 and 27 inches give better results, while 13 inches gives poor results
2	Eye-tracking accuracy under different lighting conditions	• Create testing environments with controlled lighting conditions: natural light (open curtains) and artificial light (closed curtains). • Calibrate the WebGazer system in each lighting condition. • Ask participants to perform eye-tracking tasks under each lighting condition while recording their eye movements. • Analyze collected data to assess eye-tracking accuracy.	It passes the test in good light environments but fails in bad light environments
3	Comparison of accuracy between participants wearing glasses and those without	• Invite participants to the testing session. • Perform calibration tests with WebGazer for each participant, noting whether they wear glasses or not. • Record eye-tracking accuracy data for participants wearing glasses and those without.	Passed the test with and without glasses

Table 7

ID	Description	Steps	Result
4	Calibration accuracy with different camera resolutions	• Set up web cameras with different resolutions (480p, 720p, and 1080p). • Perform calibration tests with each camera. • Record calibration results for each camera resolution.	While 720p and 1080p give better results, 480p gives poor results

6.4.3 Test Results

Calibration accuracy on various screen sizes: This test passed for screen sizes 15, 16, 17, and 27 inches but failed for the 13-inch screen. The failure for the 13-inch screen might be due to the smaller size requiring a different calibration approach. it could be an issue of the calibration patterns not being suitable for such a small display, leading to inaccuracies.

Eye-tracking accuracy under different lighting conditions: The test passed in good light environments but failed in bad light environments. The failure under bad lighting conditions could be attributed to the limitations of the eye-tracking system in capturing accurate data when visibility is reduced, affecting its ability to track eye movements effectively.

Comparison of accuracy between participants wearing glasses and those without: This test passed for both participants wearing glasses and those without. It indicates that the eye-tracking system performs consistently regardless of whether participants wear glasses or not, suggesting robustness in its calibration process.

Calibration accuracy with different camera resolutions: The test passed for camera resolutions 720p and 1080p but failed for 480p. The failure with the 480p resolution could be due to the lower image quality impacting the system's ability to detect and calibrate accurately, resulting in less reliable calibration results compared to higher resolutions.

6.4.4 Test Log

Figure 56: Calibration Screen

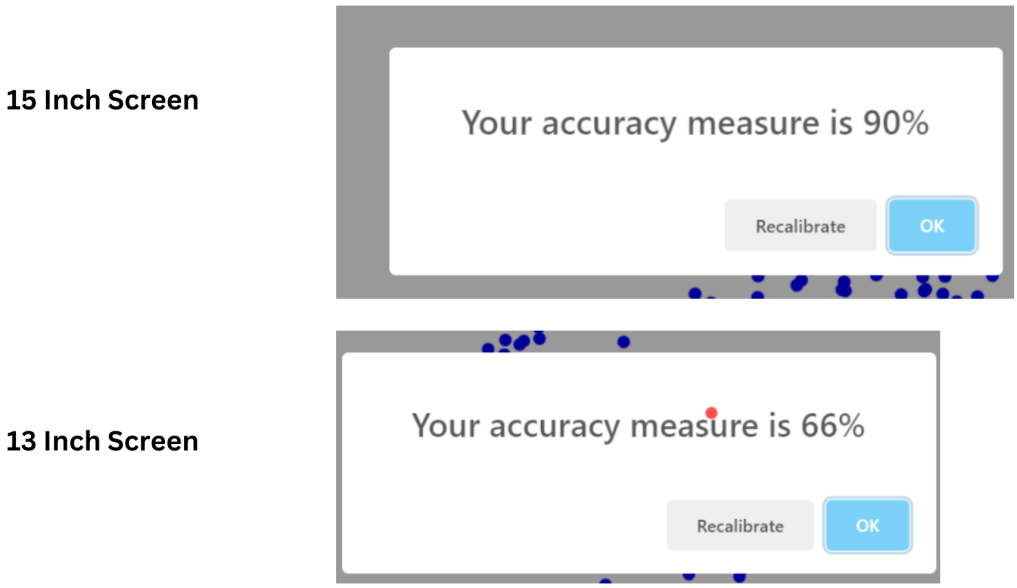


Figure 57: Comparison of 13-inch and 15-inch eye tracking accuracy

As can be seen in this screenshots above, the accuracy we obtained on 13 and 15 inch screens can be seen. We performed these tests at least five times and put the highest accuracy we received here. Accuracies for 16, 17 and 27 inches were more than 75% for each test, so only 13 inches failed this test.

6.5 Fixation Filtering Test

6.5.1 Test Procedure

The fixation filtering testing process involves several steps to ensure the accuracy and reliability of the fixation algorithm. Here's a detailed explanation of how the testing has been performed, along with the responsible individuals for each test:

1. Designing Test Screens:
    - Two different test screens were designed for the fixation filtering test.
    - The first test screen comprised a 3x3 grid categorized into 9 separate regions.
    - In the second test screen, we designed a 3x3 grid screen again, where each cell has the same size as the smallest meaningful element in our sample web pages. To determine the minimum size of a meaningful visual element, we referred to a study that focused on identifying trending elements on the same pages. For these trending elements, we calculated the individual areas of all of them and then we identified the smallest trending visual element, and we used the size of this visual element to identify the size of each cell in grid layout for the second version.
    - Test Execution:
        - Five different users were selected to participate in the test.
        - Each user underwent both tests, with a different pattern of tests prepared for each user to ensure robustness.
        - Users were instructed to look at specific cells at certain time intervals on each test screen.
    - Data Collection and Analysis with IDT Function:
        - Eye-tracking data of users were collected during the tests.
        - The data were sent to the testing program for analysis.
        - The collected raw eye-tracking data were processed using the IDT function.
        - Fixation points were generated based on the analysis.
    - Accuracy Calculation:
        - Accuracy rates were calculated initially based on users' eye movements and then recalculated using the fixation algorithms generated by the IDT function.
        - A comparison was made between the initial accuracy rates and the accuracy rates based on the fixation algorithms.

Responsibilities:

- Furkan Duman was responsible for designing the test screens, preparing different test patterns for each user, and overseeing the execution of the tests.
- **Eye Tracking Equipment:** Webgazer, a web-based eye tracker, was used to collect precise eye movement data during the tests.
- **Test Program:** Python Flask program was designed to analyze the collected data.
- **Fixation Algorithm:** The IDT function was used to create fixation algorithms for comparison.

6.5.2 Test Expectations

In this study, our expectation is to calculate a qualitative result on how much the I-DT fixation algorithm improves accuracy by applying our designed tests to users, as there is no previous qualitative data on the extent to which fixation algorithms enhance accuracy.

6.5.3 Test Results

2 different tests were applied to 5 different people and a total of 10 test results were obtained.

	Participant ID	Test Type	Row Accuracy	Idt Accuracy	Pattern
1	1	Grid	62.35	66.43	3
2	1	MFS	55.34	61.23	3
3	2	Grid	81.42	82.98	2
4	2	MFS	66.98	70.82	2
5	3	Grid	82.13	85.15	4
6	3	MFS	57.14	61.25	4
7	4	MFS	28.46	29.59	5
8	4	Grid	79.98	79.87	5
9	5	Grid	76.54	78.33	1
10	5	MFS	66.94	70.07	1

Figure 58: Fixation Filtering Test Cases

It was observed that the IDT function increased accuracy in 9 of the 10 different tests performed, and in 1 test it decreased IDT accuracy.

Eye tracking values are sensitive data, fixation algorithms sometimes cannot increase accuracy because eye tracking data may not fully reflect the user's eye movements. This can be caused by a variety of factors, such as user distraction, momentary eye flickers, or device detection errors. Additionally, some eye-tracking systems may have difficulty accurately detecting the user's gaze, especially in the case of rapid or suddenly changing eye movements. Therefore, although fixation algorithms are designed to filter out such noise and determine the correct focusing points, they may sometimes not provide the expected increase in accuracy.

In the table below, you can see how much the accuracy rate changes for each test case.

Participant ID	Row Accuracy	IDT Accuracy	Accuracy Increase Percentage
1	62.35	66.43	6.54%
1	55.34	61.23	10.64%
2	81.42	82.98	1.91%
2	66.98	70.82	5.70%
3	82.13	85.15	3.68%
3	57.14	61.25	7.19%
4	28.46	29.59	3.97%
4	79.98	79.87	-0.14%
5	76.54	78.33	2.34%
5	66.94	70.07	4.67%

Table 8: Accuracy Data

As a result, **The IDT function increased the accuracy by approximately 5.25%.**

6.5.4 Test Log

The 2 test screens used during the tests are as follows:

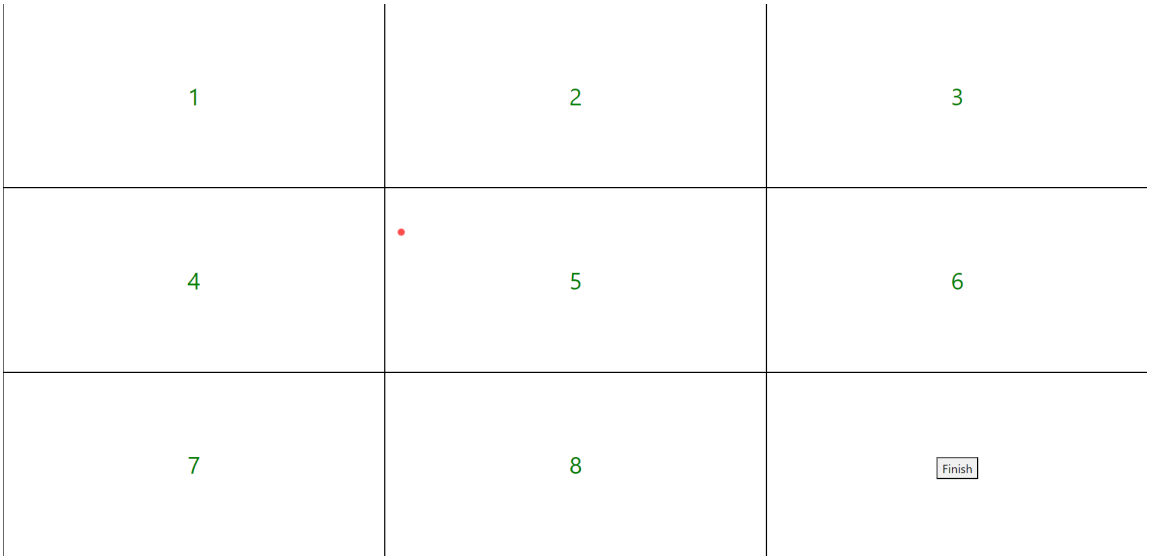


Figure 59: Test Screen - 1

In Figure 59, the first test screen used is shown.

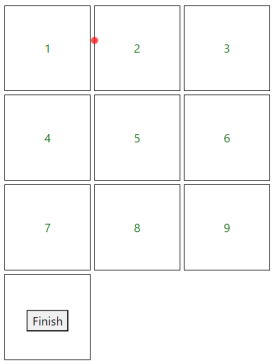


Figure 60: Test Screen - 2

In Figure 60, the second test screen used is shown.

At the end of the tests, pie charts were designed to show how much the IDT function changed the accuracy.

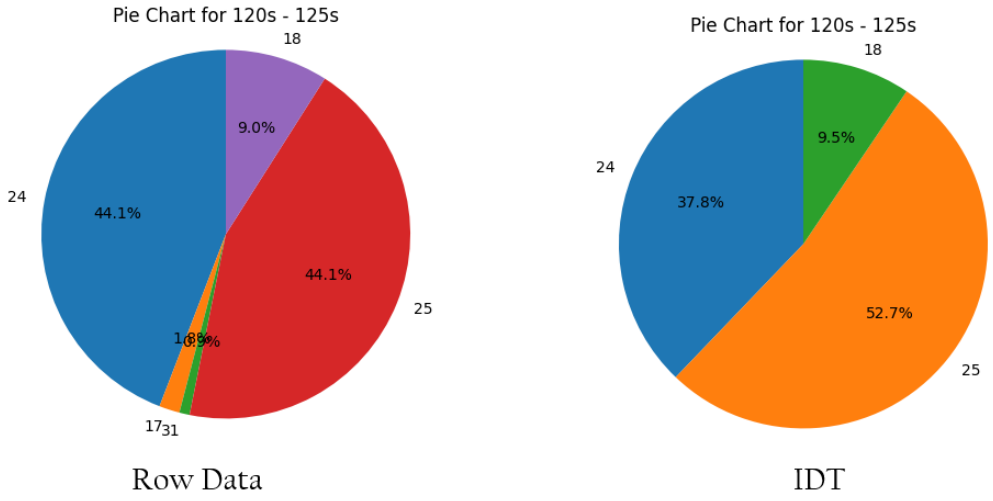


Figure 61: Row Accuracy vs IDT Accuracy-1

In the figure above, it is necessary to focus on the 25th cell between 120 seconds and 125 seconds, depending on the selected pattern. I-DT identified and eliminated cells 17 and 31 as saccade points during this time interval.

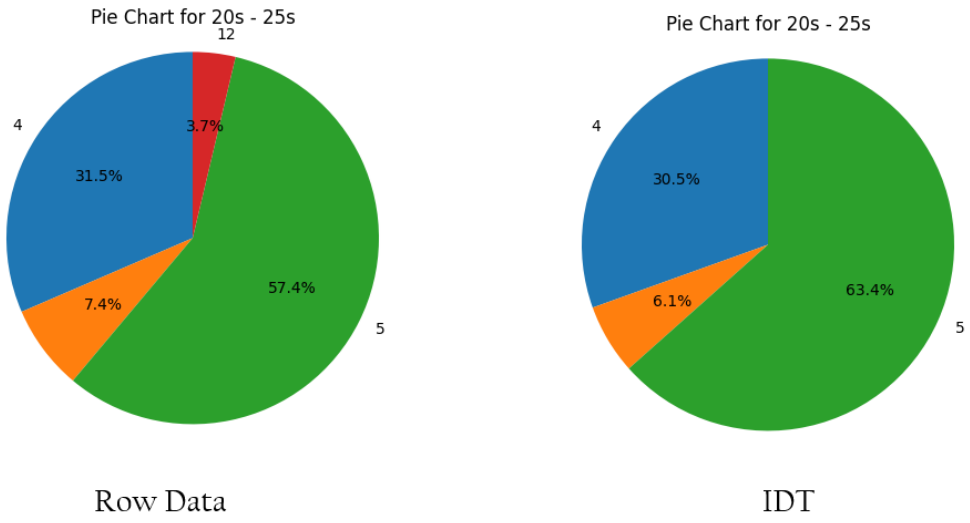


Figure 62: Row Accuracy vs IDT Accuracy-2

In the figure above, it is necessary to focus on the 5th cell between 20 seconds and 25 seconds, depending on the selected pattern. I-DT identified and eliminated cell 12 as saccade point during this time interval.

6.6 User Testing

6.6.1 Test Procedure

In this section, we present the preliminary results from our user testing of the EASIEST project. So far, we have conducted an interview with one professional healthcare provider specializing in ASD diagnosis. Kerem, Furkan and Barış made and managed this test. This report includes an overview of the interview process, key findings, and thematic analysis based on the initial data collected.

Interview Process We conducted the interview following the protocol outlined in our test plan. The process involved:

1. **Consent Form**: The healthcare professional signed a consent form to participate in the interview and agreed to audio recording of the conversation.

2. **Preliminary Questions and Explaining the Basics of the Project**: We started by asking the healthcare professional about their knowledge and experience regarding ASD diagnosis. We discussed how EASIEST works in the diagnosis of ASD and its advantages over traditional methods.

3. **Demo Video**: A demo video was shown that briefly explained the EASIEST project and demonstrated how the tests are conducted.

4. **Open-Ended Questions**: After the demo video, we asked a series of open-ended questions to gather the professional's thoughts on the project.

6.6.2 Expectations

Before conducting the user testing of the EASIEST project, we expected that the healthcare professional would validate the potential of eye-tracking technology as a valuable supplement to traditional ASD diagnostic methods. We expected detailed feedback highlighting both benefits and limitations, including how eye-tracking could enhance diagnostic accuracy and efficiency while addressing any challenges. A thorough comparison with traditional methods, such as the Autism Diagnostic Observation, was predicted, along with an evaluation of the practicality and feasibility of implementing eye-tracking in clinical environments, considering patient observation and integration with current workflows. Additionally, we hoped for formative suggestions for improvement and discussions on the broader application of eye-tracking technology beyond initial ASD screening, including ongoing monitoring, therapy, or research. Overall, we aimed to gain comprehensive insights to refine and enhance the EASIEST project for effective real-world clinical use.

6.6.3 Results

Below are the key insights gathered from the interview with the healthcare professional:

1. **Question 1: Do you think eye-tracking based ASD diagnosis is a viable supportive technology?**

   Response Summary: The healthcare professional expressed doubts about the sufficiency of eye-tracking alone for ASD diagnosis. They emphasized that ASD diagnosis usually involves multiple factors, including social interaction, non-verbal communication, and repetitive behaviors. While eye-tracking could provide valuable data, especially for screening purposes, it cannot replace the comprehensive clinical evaluation required for an ASD diagnosis. They also noted potential issues with false positives and negatives due to other factors affecting eye movement. However, they agreed that eye-tracking could be helpful for screening or as an additional clinical tool for detailed examination of eye-related findings.

2. **Question 2: What are your thoughts about the potential benefits of eye-tracking dialogue compared to traditional methods?**

   Response Summary: The healthcare professional highlighted that eye-tracking provides quantitative data, which could be valuable. Traditional methods involve a holistic assessment by a clinical team based on interaction observations, such as the ADOS (Autism Diagnostic Observation Schedule). Eye-tracking could offer useful data for these assessments, especially for adult populations.

3. **Question 3: What are the potential disadvantages of eye-tracking based diagnosis compared to traditional methods?**

   Response Summary: The potential disadvantages include difficulty in getting individuals, particularly those with low-functioning ASD, to comply with the test. Eye-tracking could

also yield false positives due to similar symptoms in other medical conditions. The healthcare professional does not believe that eye-tracking alone will speed up the diagnostic process but sees it as an additional data point that could complement clinical assessments.

4. **Question 4: Do you have any additional comments or suggestions for improvement?**

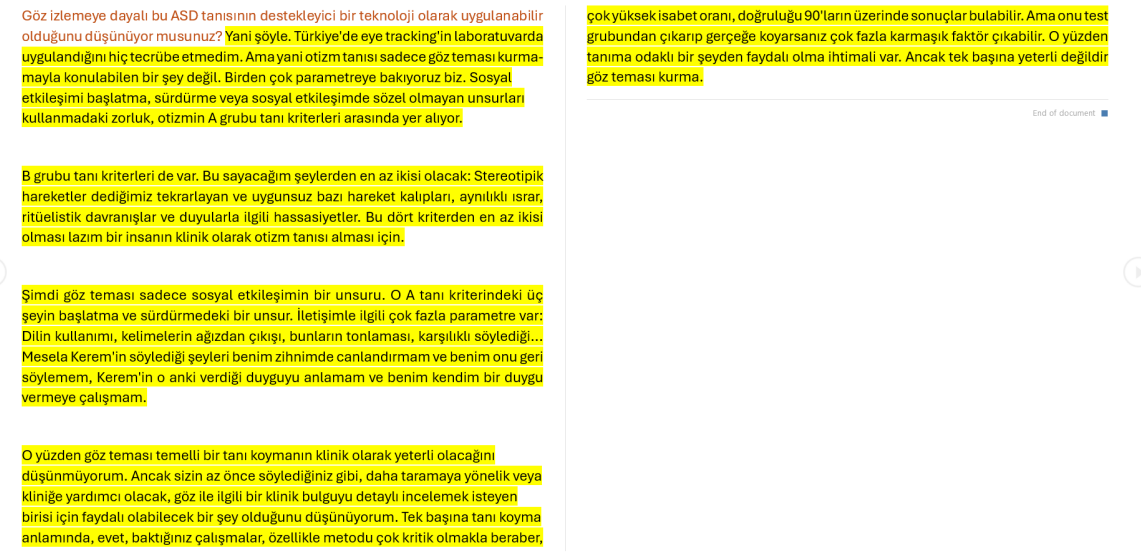
   Response Summary: The professional suggested focusing on younger children rather than adults, as late diagnosis in adults is relatively rare. They proposed developing a more cost-effective eye-tracking solution, such as a wearable device, to gather data while children interact with objects. This could provide valuable insights into their focus and interaction patterns. Moreover, incorporating other indicators such as motor movements and verbal interactions could enhance the diagnostic process.

5. **Question 5: How practical do you find this eye-tracking based supportive technology in extending or improving the diagnosis of ASD?**

   Response Summary: The healthcare professional did not think that eye-tracking would significantly speed up the diagnosis process. They suggested that it could serve as a bonus, providing additional data to complement clinical evaluations. He emphasized that it would not replace traditional methods but could be a valuable supplementary tool.

Unfortunately, we could not do the thematic analysis part as we have only been able to contact 1 doctor so far. We will complete the thamtic analysis part when we contact another doctor in the future.

6.6.4 Test Log


Göz izlemeye dayalı bu ASD tanısının destekleyici bir teknoloji olarak uygulanabilir olduğunu düşünüyor musunuz? Yani şöyle. Türkiye'de eye tracking'in laboratuvarda uygulandığını hiç tecrübe etmedim. Ama yani otizm tanısı sadece göz teması kurmamayla konulabilen bir şey değil. Birden çok parametreye bakıyoruz biz. Sosyal etkileşimi başlatma, sürdürme veya sosyal etkileşimde sözel olmayan unsurları kullanmadaki zorluk, otizmin A grubu tanı kriterleri arasında yer alıyor.

B grubu tanı kriterleri de var. Bu sayacağım şeylerden en az ikisi olacak: Stereotipik hareketler dediğimiz tekrarlayan ve uygunsuz bazı hareket kalıpları, aynılıklı ısrar, ritüelistik davranışlar ve duyularla ilgili hassasiyetler. Bu dört kriterden en az ikisi olması lazım bir insanın klinik olarak otizm tanısı alması için.

Şimdi göz teması sadece sosyal etkileşimin bir unsuru. O A tanı kriterindeki üç şeyin başlatma ve sürdürmedeki bir unsur. İletişimle ilgili çok fazla parametre var: Dilin kullanımı, kelimelerin ağızdan çıkışı, bunların tonlaması, karşılıklı söylediği... Mesela Kerem'in söylediği şeyleri benim zihnimde canlandırmam ve benim onu geri söylemem, Kerem'in o anki verdiği duyguyu anlamam ve benim kendim bir duygu vermeye çalışmam.

O yüzden göz teması temelli bir tanı koymanın klinik olarak yeterli olacağını düşünmüyorum. Ancak sizin az önce söylediğiniz gibi, daha taramaya yönelik veya kliniğe yardımcı olacak, göz ile ilgili bir klinik bulguyu detaylı incelemek isteyen birisi için faydalı olabilecek bir şey olduğunu düşünüyorum. Tek başına tanı koyma anlamında, evet, baktığınız çalışmalar, özellikle metodu çok kritik olmakla beraber, çok yüksek isabet oranı, doğruluğu 90'ların üzerinde sonuçlar bulabilir. Ama onu test grubundan çıkarıp gerçeğe koyarsanız çok fazla karmaşık faktör çıkabilir. O yüzden tanıma odaklı bir şeyden faydalı olma ihtimali var. Ancak tek başına yeterli değildir göz teması kurma.

End of document

Figure 63: An example image of the interview audio recording transcribed into text

Those written in red texts belong to ours, and those marked in yellow belong to the doctor.

7 Project Scheduling

7.1 Milestones and Tasks

Milestones:

- Implementation of web-gazer (M1)
- Implementation of 1st ML model (M2)
- Implementation of database (M3)
- Implementation of 2nd ML model (M4)
- Perform evaluations on project (M5)

ID	Task	Duration	Dependency	Members
T1	Research and revise all related academic articles	2 weeks		Barış, Furkan
T2	Implement the web-gazer	5 days	M1	All members
T3	Design calibration	1 week	T2	Kerem, Barış
T4	Construct back-end and deploy website	2 weeks	T2	Daniil, Kerem
T5	Implement the web-pages to the website	4 days		All members
T6	Generate user CSV file after test	2 weeks		All members
T7	Design interactive front-end	3 days		Furkan, Daniil
T8	Design database structure using PostgreSQL	1 week		Kerem, Daniil
T9	Integrate the database	1 week	T8, M3	All members
T10	Integrate the ML model	2 weeks	M2	All members
T10	Add History and Search features to website	1 week		Daniil
T11	Prepare grid-based inputs for the second ML model	2 weeks		Kerem, Barış
T12	Integrate the second ML model	4 week	T11, M4	All members
T13	Design and implement admin panel	3 weeks		All members
T14	Conduct several testings on our project	4 weeks	M5	All members

Figure 64: Milestones and Tasks

7.2 Gantt Chart

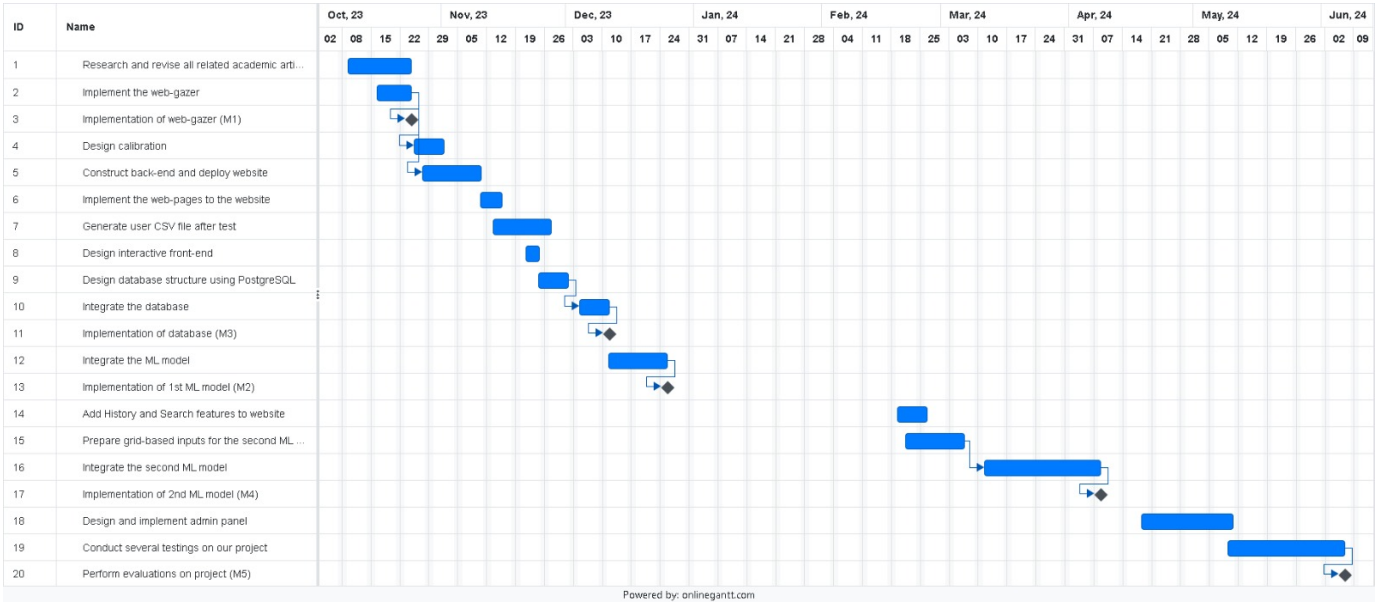


Figure 65: Gantt Chart

8 Conclusion

Our project EASIEST has addressed an innovative approach to the traditional autism diagnosis methods for adults. Traditional approach often being time-consuming and resource-intensive, involving detailed interviews, assessments, and expensive techniques like MRI scans, could be not reachable for all of the people, that is why we are trying to ease the access to the detection. By integrating gaze-tracking technology and machine learning, EASIEST offers a novel solution for autism detection in adults, using a basic computer camera to collect and analyze eye movement and behavioural data. This approach significantly reduces time and costs associated with initial screenings, making the process more accessible and user-friendly. The project successfully demonstrates how leveraging advanced technology can overcome the limitations of traditional diagnostic methods, promising a more efficient and accessible means for early autism detection, ultimately aiming to enhance the quality of life for individuals with autism and their families. After the implementation of the project was completed, we completed the tests of our project by applying tests such as Unit Test, Hardware Test, Fixation Filtering Tests. For Unit Testing, our goal was to find 5 doctors for User Testing but we could only find 1 doctor. Since we are working as doctors this term, we could not complete our thematic test, but we will show our project to other doctors in the future. In addition, in the 2nd semester, we implemented a new machine learning algorithm into our project and this machine learning accuracy is significantly higher than the first one. Currently, we can use two different machine learning algorithms. In term 2, we also introduced a more user-friendly interface regarding general access for customers, where people can update their accounts, run multiple tests for the same people, and view patients' history. A team of four people worked together on this project, Barış Aydınay, Furkan Duman, Kerem Keptiğ, and Daniil Belousov. We would like to express our endless gratitude to Mr. Şükrü Eraslana and Ms. Yeliz Yeşilada, who supported us with their unique ideas and always supported us in this project.

9 References

References

[1] Ami Klin, Celine Saulnier, Katherine Tsatsanis, and Fred R Volkmar. Clinical evaluation in autism spectrum disorders: Psychological assessment within a transdisciplinary framework. *Handbook of autism and pervasive developmental disorders*, 2:772–798, 2005.

[2] Parisa Moridian, Navid Ghassemi, Mahboobeh Jafari, Salam Salloum-Asfar, Delaram Sadeghi, Marjane Khodatars, Afshin Shoeibi, Abbas Khosravi, Sai Ho Ling, Abdulhamit Subasi, Roohallah Alizadehsani, Juan M. Gorriz, Sara A. Abdulla, and U. Rajendra Acharya. Automatic autism spectrum disorder detection using artificial intelligence methods with mri neuroimaging: A review. *Frontiers in Molecular Neuroscience*, 15, 2022.

[3] Automatic autism spectrum disorder detection using artificial intelligence methods with MRI neuroimaging: A review — doi.org. `https://doi.org/10.3389/fnmol.2022.999605`. [Accessed 22-11-2023].

[4] Webcam-Based Eye Tracking vs. an Eye Tracker [Pros & Cons] - iMotions — imotions.com. `https://imotions.com/blog/learning/best-practice/webcam-eye-tracking-vs-an-eye-tracker/`. [Accessed 22-11-2023].

[5] Henrik Skovsgaard, Javier San Agustin, Sune Alstrup Johansen, John Paulin Hansen, and Martin Tall. Evaluation of a remote webcam-based eye tracker. In *Proceedings of the 1st Conference on Novel Gaze-Controlled Applications*, NGCA '11, New York, NY, USA, 2011. Association for Computing Machinery.

[6] Victoria Yaneva, Le Ha, Sukru Eraslan, and Yeliz Yesilada. Detecting high-functioning autism in adults using eye tracking and machine learning. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, PP:1–1, 04 2020.

[7] Sahin Bodur and A.Şebnem Soysal Acar. Otizmin erken tanýsý ve Önemi. 01 2004.

[8] B. Pugazhenthi, G.S. Senapathy, and M. Pavithra. Identification of autism in mr brain images using deep learning networks. In *2019 International Conference on Smart Structures and Systems (ICSSS)*, pages 1–7, 2019.

[9] WebGazer.js: Democratizing Webcam Eye Tracking on the Browser — webgazer.cs.brown.edu. `https://webgazer.cs.brown.edu/`. [Accessed 30-11-2023].

[10] TurkerGaze — turkergaze.cs.princeton.edu. `https://turkergaze.cs.princeton.edu/`. [Accessed 30-11-2023].

[11] Online Eye Tracking — Webcam eye-tracking software — gazerecorder.com. `https://gazerecorder.com/`. [Accessed 30-11-2023].

[12] Online Research Platform with Webcam Eye-Tracking — RealEye.io — realeye.io. `https://www.realeye.io/`. [Accessed 30-11-2023].

[13] PyGaze — Open source eye-tracking software and more. — pygaze.org. https://www.pygaze.org/. [Accessed 30-11-2023].

[14] Erfan Khalaji, Sukru Eraslan, Yeliz Yesilada, and Victoria Yaneva. Effects of data preprocessing on detecting autism in adults using web-based eye-tracking data. *Behaviour Information Technology*, 42:1–9, 09 2022.

[15] Qiuhong Wei, Huiling Cao, Yuan Shi, Ximing Xu, and Tingyu Li. Machine learning based on eye-tracking data to identify autism spectrum disorder: A systematic review and meta-analysis. *Journal of Biomedical Informatics*, 137:104254, 2023.

[16] PBKDF2 - Wikipedia — en.wikipedia.org. https://en.wikipedia.org/wiki/PBKDF2#:~:text=In%20cryptography%2C%20PBKDF1%20and%20PBKDF2,%2C%20specifically%20PKCS%20%235%20v2. [Accessed 30-11-2023].

10 Appendices

10.1 Acronyms and abbreviations

ASD : Autism spectrum disorder

10.2 Glossary

Eye Data Recording: Captures fixation gaze data during patient engagement.

Calibration and Main Tests: The calibration test is the test used to calibrate the eye-tracker used before showing the main test to the patient. If the user passes the calibration test, it is transferred to the main test. The main test is where eye data is collected by asking questions to patients.

Metrics Generation: Key metrics derived from recorded data, utilized by the machine learning (ML) model.

ML Model Outcome: Provides accuracy and results stored in the database and relayed to registered users.

Stakeholder: Any person or organization affected by or interested in the system, such as Healthcare Professionals and Patients.