

A Time Management Platform Based on Real-Time Eye Tracking



New New New Seagull

201924480 Junhyung Park

201924431 Sangyu Kim

201924530 Sangjun Lee

Advisor: Sangryul Jeon

Date: October 15, 2024

Pusan National University

Contents

1	Introduction	1
1.1	Research Background and Existing Problems	1
1.2	Research Objectives	1
2	Research Background	2
2.1	EyeTracker	2
2.1.1	Webcam-Based Eye Tracker	2
2.1.2	Limitations of the Eye Tracker	2
2.2	GPT API	3
3	Research Content	4
3.1	Comparison of Eye Trackers	4
3.2	WebGazer.js	4
3.3	Implementation of WebGazer	5
3.3.1	Reason for Implementation	5
3.3.2	Training Process	5
3.3.3	Operating Principle	6
3.3.4	Noise Filtering	7
3.4	JWT	9
3.5	User Authentication and Authorization Using Spring	9
3.5.1	User Login Process	9
3.5.2	Operating Principle of the Spring Security Filter	10
3.6	Deployment Automation	10
3.7	Docker	12
3.8	Deployment of a MySQL Database Server Using MS Azure	12
3.8.1	Reasons for Adopting MS Azure	13
3.8.2	Introduction to Azure Database for MySQL	13

3.8.3	DB Deployment Process	13
3.9	Project Features	14
3.9.1	Follow and Follower Features	14
3.9.2	Top 10 Ranking	14
3.9.3	Todo	14
3.9.4	Screen Time and Categories	15
3.9.5	Study Time Management System	16
4	Analysis and Evaluation Results	18
4.1	Development Schedule	18
4.2	Division of Responsibilities	19
4.3	API Specification	19
4.4	ERD	20
4.5	GitHub	20
5	Conclusion and Future Work	22
5.1	Conclusion and Expected Benefits	22
5.2	Future Research Directions	22

Chapter 1 Introduction

1.1 Research Background and Existing Problems

Recently, students who need to concentrate on their studies have increasingly experienced difficulty focusing because they are easily distracted by digital media such as YouTube and TikTok. Existing study timers have the problem of being ineffective due to a lack of enforcement. For example, “Passion-Filled Timer,” released by Pallo Co., Ltd. on November 11, 2018, allows users to measure and share their study time with others. By observing people who are studying, users can be motivated to study as well. However, the system does not determine whether users are actually concentrating on their studies during the recorded time. Therefore, users can leave the timer running while sleeping or playing, resulting in an unreasonably inflated study time.

To address this issue, this project proposes the use of eye-tracking technology in computer vision to identify the user’s gaze and determine whether the user is currently concentrating on studying. By allowing the timer to run based on the user’s level of concentration, the proposed system aims to encourage efficient study habits and promote user motivation.

1.2 Research Objectives

The objective of this study is to develop a real-time eye-tracking technology, EyeTracker, using a webcam and accurately identify the points at which users focus on the screen. First, the performance of the eye-tracking system is improved so that the movement of the user’s gaze can be analyzed precisely. Next, a model is developed to automatically determine whether the content the user is viewing on the screen is related to learning by utilizing gaze data. Finally, the ultimate goal is to implement a web service that integrates these functions so that users can easily access and use it. The system aims to analyze each user’s learning habits and achievements and provide personalized learning feedback.

Chapter 2 Research Background

2.1 EyeTracker

2.1.1 Webcam-Based Eye Tracker

Hardware-based eye trackers use dedicated cameras and sensors designed to capture eye movements under various lighting conditions. These devices utilize infrared diodes to capture corneal reflections and can precisely measure head movements and various physiological changes in the eye region. Although they provide high accuracy and reliability, they require expensive hardware and may be subject to environmental restrictions.

In contrast, webcam-based eye tracking is a video-based technology that generally uses a standard camera capable of detecting light in the visible spectrum. It tracks the user's gaze by analyzing facial and pupil movements without using infrared diodes, obtaining gaze data through image-processing techniques rather than corneal reflection.

In this study, webcam-based eye tracking was selected in consideration of accessibility and cost efficiency. Since webcams are built into most computers, no additional equipment is required, and the system does not depend on specialized hardware. This makes it possible to implement an eye-tracking system that anyone can easily use and significantly improves user accessibility.

2.1.2 Limitations of the Eye Tracker

1. Insufficient number of pixels in the eye region due to webcam resolution

Although most modern webcams support high resolutions, a balance must be maintained between latency and frame rate. A low resolution can ensure real-time performance, but it results in a small number of pixels representing the eye region. When the number of pixels is insufficient, the information required for eye tracking is limited, which leads to reduced accuracy and reliability.

2. Accuracy issues under dark lighting conditions

Unlike dedicated hardware devices, webcam-based eye tracking performs visual tracking using only data from the visible-light spectrum rather than infrared light. Therefore, the data may change depending on the surrounding lighting conditions. In general, when contrast is low, the system may have difficulty detecting eye movements against the facial background, resulting in reduced accuracy.

3. Accuracy issues caused by external conditions such as glasses, facial angle, and movement

The user's glasses or other worn accessories, the camera angle, and head movements during eye tracking can be major factors that reduce the performance of the model used to extract the eye region from the facial region. A limitation exists in that the model can make accurate predictions primarily under ideal conditions with few such external factors.

2.2 GPT API

Overview of the GPT API

1. The GPT (Generative Pre-trained Transformer) API is an artificial-intelligence-based text generation and processing service provided by OpenAI. Through this API, developers can use GPT models to perform various natural language processing tasks. GPT models are pre-trained on large-scale language data and possess the ability to understand and generate natural language. By providing these models through an API, users can easily access and utilize them.
2. **Text generation:** Generates highly relevant and natural text based on the text entered by the user. It can be used for article writing, storytelling, and automatic response generation.
3. **Language understanding:** Provides answers to questions and performs tasks such as text summarization and sentiment analysis.
4. **Translation:** Supports text translation between various languages.
5. **Code generation and modification:** Helps generate code snippets or modify existing code based on its understanding of programming languages.

Chapter 3 Research Content

3.1 Comparison of Eye Trackers

Table 3.1: Comparison of Eye Trackers

Product	Language	Performance	Price	Main Features
GazePointer	C#	Medium (basic eye tracking, slight delay)	Free	Webcam-based, desktop application
WebGazer.js	JavaScript	High (real-time, minimal delay)	Free (open source)	Web-browser compatible, API support
OpenGaze	C++	Medium (real-time, system-dependent)	Free (open source)	Computer vision algorithms, desktop application
Cambridge Face-Tracker	C++	High (real-time, low latency)	Free	High-performance face tracking, desktop application
iTracker	Python	Variable (system-dependent)	Free (open source)	Deep-learning-based, separate web integration required

3.2 WebGazer.js

Overview of WebGazer

WebGazer is an eye-tracking library that uses a standard webcam to infer the gaze location of web visitors on a page in real time. The eye-tracking model included in WebGazer observes how web visitors interact with web pages, calibrates itself, and trains a mapping

between eye features and screen positions. As a result, tracking accuracy can automatically improve over time without requiring a separate and complicated calibration process.

WebGazer is written entirely in JavaScript and can be integrated into any website. This ease of integration simplifies the development process and saves time. Since WebGazer.js runs entirely in the client-side browser, video data does not need to be transmitted to a server.

3.3 Implementation of WebGazer

3.3.1 Reason for Implementation

Inspired by the approach of WebGazer.js, this project implemented its own web-based eye-tracking system. While referring to WebGazer's core concepts, including interaction-based self-calibration and pure JavaScript implementation, an optimized solution was developed to meet the specific requirements of this project.

Specifically, the following core components were implemented directly:

- A module for collecting and analyzing the user's click and mouse-movement data
- An algorithm for predicting gaze positions based on the collected data
- A self-calibration system that improves accuracy in real time

Through this direct implementation approach, it became possible to retain the advantages of WebGazer while providing a customized solution that better meets the specific requirements of the project. Compared with simply using an existing library, this approach made it possible to build a much more flexible and extensible system and establish a foundation for continuous improvement and optimization in the future.

3.3.2 Training Process

To map the pupil position and eye features to screen coordinates, a two-dimensional vector representing the pupil position and a 120-dimensional vector representing eye features must each be mapped to gaze coordinates on the screen. Since these values vary depending on the three-dimensional position and rotation of the user's head relative to the camera and screen, careful calibration and extensive computation are required for accurate processing.

Instead, WebGazer generally relies on continuous self-calibration through common user interactions. According to Gaze and Cursor Alignment in Web Search[1], the distance between the gaze and the cursor at the moment of a click is, on average, 74 pixels. Although this distance may vary depending on the task, this study simplifies the analysis by assuming that the gaze and cursor are perfectly aligned when a click occurs. This assumption is generally sufficiently comprehensive to be applied across various environments and tasks. Although this project focuses on clicks and cursor movements, the approach can be extended to other types of interaction.

3.3.3 Operating Principle

Acquisition of Facial Landmark Coordinates Using the TFJS Face-Landmark Model

The first step is to extract the coordinates of major landmarks on the user's face using TensorFlow.js's face-landmark model. The Face-Landmark model is a deep-learning-based facial recognition model that detects facial features in real time from image frames captured by a webcam. This model provides essential information for extracting the eye positions.

The TFJS¹ model was selected because of its high usability in JavaScript environments.

Eye Image Extraction

Once the landmark coordinates are obtained, the region corresponding to the eyes is cropped from those coordinates to obtain an image. The eye-region image is used as input for eye tracking. The bounding box of the eye region is obtained using the landmark coordinates that make up the eyes, and the image is then extracted.

Eye Image Preprocessing

The extracted eye image is resized to a resolution of 10×6 to be used as input for linear regression. The resized image undergoes grayscale conversion and histogram equalization to remove noise, reduce the influence of environmental conditions, and improve robustness. The image is then converted into a one-dimensional vector and used as input for the linear regression model.

¹<https://github.com/tensorflow/tfjs-models/tree/master/face-landmarks-detection#keypoints>

Linear Regression Training

The eye images collected through the webcam are used as the variable X , while the user's click positions are used as the variable Y . The relationship between X and Y is then learned using a linear regression algorithm.

Linear regression is an algorithm that identifies a linear relationship between X and Y . In this project, it predicts the position the user is actually looking at on the monitor from the eye-image vector. During this process, regression coefficients are learned based on the user's gaze data, and these coefficients are later used for real-time eye tracking.

Reasons for Using Linear Regression Instead of Deep Learning

1. Real-time data processing

Linear regression can immediately reflect real-time data because it requires relatively little computation. Since the system continuously relies on self-calibration through user interactions, it must be able to update the model quickly whenever new data are added. Deep learning may require substantial computational resources, which can reduce real-time performance and lead to a negative user experience.

2. Small amount of data

Linear regression uses a simple mathematical model and is suitable for estimating the relationship between eye-image vectors and gaze coordinates. It does not require complex data preprocessing or large-scale training. Therefore, it can achieve satisfactory performance with a relatively small amount of data.

3.3.4 Noise Filtering

Window Slicing

In this study, the Window Slicing technique was applied to further improve the prediction accuracy of the eye-tracking data while considering uncertainty. This technique focuses on deriving the final prediction value from recent prediction results in order to provide a consistent gaze-tracking path.

Window Slicing operates by selecting the four most recent prediction values at a given point in time and calculating their average. This reduces the variability of individual predictions and generates more stable gaze coordinates. This approach provides the following benefits:

- **Reduction of variability:**

Using the average of recent prediction values minimizes the effect of sudden fluctuations caused by temporary noise or external factors on the final prediction. This contributes to improving the reliability of the eye-tracking results.

- **Maintenance of temporal continuity:**

Since Window Slicing uses a set of recent prediction values, it can naturally reflect the temporal continuity of gaze movements. This helps represent the flow of the user's attention across the screen more smoothly.

Kalman Filter

In this study, a Kalman Filter was applied to improve the accuracy of the webcam-based eye-tracking system. Eye-tracking technology analyzes in real time where a user is looking on the screen. During this process, small amounts of noise or momentary fluctuations can create visually unstable tracking paths. To solve this problem, a Kalman Filter was applied to connect gaze coordinates smoothly and stably.

The Kalman Filter operates by predicting the gaze coordinates at the current point in time and updating the prediction by comparing it with the actually measured gaze data. It compensates for potential noise and errors while naturally tracking the gaze movement path.

The filter consists of two main stages:

- **Prediction stage**

Based on the gaze coordinates from the previous point in time, the system predicts the position at which the user will look next on the screen. This stage helps maintain the continuity of gaze movement and prevents sudden changes.

- **Update stage**

After comparing the actually measured gaze coordinates with the predicted values, the current gaze coordinates are adjusted based on the difference between the two values. Noise is taken into account to improve the accuracy of the tracking path.

As a result, the gaze-tracking path is displayed smoothly and naturally on the screen, and irregularities in real-time data can be effectively reduced. In particular, the tracking results can be displayed stably even when the user moves their gaze quickly or the lighting

conditions change. The Kalman Filter played an important role in stabilizing the eye-tracking technology, ultimately enabling the system to provide visually consistent and reliable tracking data.

3.4 JWT

SeagullsRoom is expected to establish itself as a comprehensive learning management platform that goes beyond simply measuring study time by utilizing the GPT API to comprehensively improve and optimize users' learning habits. Through continuous improvement and expansion of its functions based on user feedback, the platform will be able to provide a more effective and personalized learning experience in the future.

Reasons for Adoption

- **Statelessness:** The system provides excellent scalability because the server does not need to maintain a separate session storage system.
- **Security:** Encrypted signatures ensure data integrity, while minimizing the exposure of sensitive information even if a token is stolen.
- **Efficiency:** Since authentication information is included in the token, the number of database queries can be reduced.
- **Flexibility:** A consistent authentication system can be maintained across various environments, including web and mobile platforms.
- **Cross-domain authentication:** Token-based authentication is possible across different domains.

3.5 User Authentication and Authorization Using Spring

This project adopted the Spring Security framework to manage user authentication and authorization securely and efficiently. Spring Security is a powerful authentication and access-control framework responsible for securing Spring-based applications.

3.5.1 User Login Process

1. A login request is received from the user.

2. The `UsernamePasswordAuthenticationFilter` extracts the ID and password and passes them to the Authentication Manager.
3. The Authentication Manager retrieves the user's information from the database and verifies it.
4. If the verification is successful, `successfulAuth` generates a JWT through `JWTUtil` and sends it to the user.

3.5.2 Operating Principle of the Spring Security Filter

Spring Security intercepts client requests at an intermediate filter while they pass through multiple filters toward the `DispatcherServlet` (Controller), and then performs validation, including authentication and authorization.

The user's request proceeds through the following process:

Client Request → Servlet Container → Servlet Filter → Servlet (Controller)

3.6 Deployment Automation

This project implemented a deployment automation pipeline using GitHub Actions and Docker to establish an efficient and stable deployment process. This ensured a consistent deployment process from development through production and minimized errors caused by manual operations.

Overview of Deployment Automation

Deployment automation is a technology that automates the build, testing, and deployment processes of software, thereby improving development productivity and deployment stability. In this project, GitHub Actions was selected as the CI/CD (continuous integration/continuous deployment) tool, while Docker was selected as the containerization platform to construct the automation pipeline.

Use of GitHub Actions

GitHub Actions is a workflow automation tool provided by GitHub and performs the following tasks:

- Code-change detection: Automatically executes the workflow when a push or pull-request event occurs on the main branch
- Automated build and testing: Builds the project and runs unit tests
- Docker image creation: Automatically creates a Docker image for the application
- Image push: Pushes the created image to Docker Hub or a private registry

Docker Containerization

By containerizing the application using Docker, the following benefits were obtained:

- Environmental consistency: Maintains consistency between development, testing, and production environments
- Ease of deployment: Enables simple deployment and rollback using container images
- Resource efficiency: Optimizes resource utilization through lightweight and fast container execution
- Scalability: Makes it easy to scale container instances as needed

Automation Pipeline Flow

1. Code push: A developer pushes code to the main branch of the GitHub repository.
2. Workflow trigger: The GitHub Actions workflow is executed automatically.
3. Build and testing: The project is built and automated tests are performed.
4. Docker image creation: A Docker image is created after a successful build.
5. Image push: The generated image is pushed to a registry.
6. Deployment: The application is automatically updated on the target server using the new image.

By establishing this deployment automation pipeline, the project created a more efficient and stable development and operations process. This significantly improved the service quality, scalability, and maintainability of the project.

1. When a Pull Request is merged into the main branch, the GitHub Actions script is executed.
2. GitHub Actions creates a Docker image based on Docker.
3. After logging in to Docker Hub, the generated Docker image is pushed.
4. The GitHub Actions script accesses the EC2 instance, pulls the pushed Docker image, and runs it.
5. Deployment is completed.

3.7 Docker

This project adopted Docker to maximize the efficiency, consistency, and flexibility of the deployment and production environments. Docker is an open-source platform for containerizing, developing, deploying, and running applications, and it is a core technology in modern software development and operations.

The introduction of Docker modernized the development and operations processes of this project and significantly improved the stability and scalability of the application. As a result, it led to improved service quality and reduced maintenance costs.

```
FROM openjdk:17-jdk
ARG JAR_FILE=SeagullsRoom/build/libs/*.jar
COPY ${JAR_FILE} app.jar
ENTRYPOINT ["java", "-Dspring.profiles.active=docker", "-jar", "app.jar"]
```

Figure 3.1: Dockerfile Created

3.8 Deployment of a MySQL Database Server Using MS Azure

In this project, a MySQL database server was deployed using MS Azure to enable efficient database management and smooth development in a cloud environment. Cloud technology has become a core infrastructure component of modern software development. Reducing the burden of server management while securing scalability and availability is an important factor in the success of a project. In addition, practical skills were strengthened through the experience of deploying the system in an actual cloud environment.

3.8.1 Reasons for Adopting MS Azure

1. **Experience with a cloud environment:** To gain practical experience by deploying the system in a cloud environment.
2. **Student support benefits:** A credit of \$100.00 was provided through the MS Azure student subscription.
3. **Convenient management features:** Azure Database for MySQL is provided as a managed service, so users do not need to install and manage the server themselves.
4. **High scalability and availability:** MS Azure provides automatic scaling through its global cloud infrastructure and can flexibly adjust performance according to database traffic, ensuring high scalability and availability.
5. **Security and data protection:** Azure provides strong security features. Important data can be protected through data encryption, access control, and network firewall configuration.
6. **Localized data-center support:** Although Azure is a global service, it minimizes network latency and provides faster response times by offering services through data centers located close to users. For this project, the Korea Central server was used for deployment.

3.8.2 Introduction to Azure Database for MySQL

Azure Database for MySQL is a fully managed MySQL database service provided by MS Azure. Users can utilize a stable and scalable MySQL environment in the cloud without having to manage the MySQL server directly. This service ensures performance and reliability through features such as automatic backups, high availability, and automatic scaling.

3.8.3 DB Deployment Process

1. **Accessing the Azure Portal:** Log in with a Microsoft Azure account and access the Azure portal.
2. **Creating a Resource:** Select “Create a resource,” search for “Azure Database for MySQL,” and create a database server.

3. **Configuration:** Configure the database server settings, including the region, pricing tier, and version, and then set the administrator username and password.
4. **Network Configuration:** Configure firewall rules for database access and allow the required IP addresses.
5. **Deployment Completion and Connection:** After the server has been deployed, the database can be accessed through a MySQL client using the provided connection information.
6. **Database Creation and Management:** Create databases and tables and manage data using tools such as MySQL Workbench.

3.9 Project Features

3.9.1 Follow and Follower Features

Follow and follower functionality was implemented so that users can follow one another. The relationship between users and friends is managed through the Friend table, and the friend-addition function was implemented based on a one-to-many relationship between a User and Friends. Through this feature, users can add other users as friends. Validation is included to prevent duplicate additions and to prevent users from adding themselves as friends.

3.9.2 Top 10 Ranking

This feature displays the top 10 users based on the total study time of all users on the current day.

To implement this feature, the study data for the current date is retrieved, grouped by user ID, and multiple study records are aggregated. The results are then sorted in descending order according to the total study time, and the top 10 users are retrieved using Pageable.

3.9.3 Todo

The Todo feature is a system that allows users to manage their personal to-do lists. Users can create new Todo items containing a title and description, which are stored in the

database. Each Todo item can be marked as completed, allowing users to track their progress.

Users can view, modify, and delete their Todo items. The view function lists all of the user's Todo items, while the edit function allows the user to change the title, description, and completion status. The delete function removes Todo items that are no longer needed.

The system protects personal information through user authentication, and each user can manage only their own Todo items. In addition, input validation prevents inappropriate data from being entered. Through this functionality, users can systematically manage and track their study plans.

3.9.4 Screen Time and Categories

The Screen Time & Category feature is a system that allows users to analyze and manage their learning activities in greater detail. This feature tracks screen activity based on user-defined categories and presents the results visually.

Users can define and add their own unique categories through the web interface. The added category information is stored in the browser's `localStorage` and continuously managed on the client side. When a study session begins, the stored category information is automatically included in the `ChatGPTRequestDto`.

Specifically, whenever a screen capture is taken, the list of user-defined categories is retrieved from `localStorage` and included in the request sent to the server. During this process, the default categories, such as "GAME," "SNS," and "OTHER," are sent to ChatGPT together with the user-defined categories.

ChatGPT analyzes screen activity based on this expanded list of categories. By including user-defined categories, the AI can more accurately understand and classify each user's unique learning environment and activity patterns. For example, if a user adds a category called "Online Lecture," ChatGPT can recognize it and classify related screen activity under that category.

Through this customized approach, the system can provide a more accurate and personalized study-concentration monitoring service that takes each user's individual learning habits and environment into account. Users can experience more accurate classification and tracking of their learning activities, which can ultimately lead to more effective learning management and improvement.

Screen activity is captured at configured intervals and analyzed by ChatGPT, and the

results are aggregated by category. This data is stored and accumulated on the server, allowing users to visually check their screen-time data through the web interface.

On the frontend, a doughnut chart is used to intuitively display the proportion of time spent in each category. This allows users to understand their learning patterns and time usage at a glance.

In addition, this feature can be used to monitor users' learning behavior and provide feedback. For example, the system can issue a warning if a user spends an excessive amount of time in a particular category, or provide advice when a pattern of insufficient concentration is detected.

This detailed data tracking and analysis helps users improve their learning habits and manage their time more efficiently. Consequently, this feature can contribute to improving users' learning efficiency and achieving better learning outcomes.

3.9.5 Study Time Management System

1. Study Start Function

The study start function is activated when a user begins a learning session. This function is implemented through the following process:

- (a) The user clicks the "Start Study" button on the web interface.
- (b) The system checks whether the user has defined at least three categories.
- (c) The system requests and obtains permission to share the screen.
- (d) The eye-tracking system is initialized and started.
- (e) The timer starts measuring the study time.
- (f) The screen is captured periodically, every five seconds, and sent to the server for AI analysis.

2. Study Stop (Pause) Function

The study stop function is used when the user temporarily interrupts a learning session.

- (a) The function is activated when the user clicks the "Pause" button or when the AI automatically detects non-learning activity. In particular, when non-learning activity is detected, a notification event is sent using the Web API.
- (b) The timer is stopped, and the study time accumulated up to that point is saved.

- (c) The screen-capture and analysis processes are paused.
- (d) When the user clicks the “Restart” button, the session resumes from the point at which it was interrupted.

3. Study Save (End) Function

The study save function is used to completely end a learning session and save the data.

- (a) The user clicks the “End” button.
- (b) The timer is stopped, and the final study time is calculated.
- (c) The eye-tracking system and screen sharing are terminated.
- (d) The study-time data is sent to the server and saved.
- (e) The screen-time data for each category is sent to the server and saved.
- (f) Temporary data stored in `localStorage` is cleared.
- (g) The user interface is reset to its initial state.

These functions are processed on the server side through `StudyService` and `StudyController`, while they are implemented on the client side using JavaScript. This system focuses on accurately tracking users’ learning patterns, supporting effective time management, and improving learning efficiency through analysis based on the ChatGPT API.

Chapter 4 Analysis and Evaluation Results

4.1 Development Schedule

The development schedule shown in Table 4.1 presents the overall progress of the project on a monthly basis. All major development stages, from the planning phase to final deployment and revisions, are visually represented. This makes it possible to clearly identify the starting and ending points of tasks in each development category, as well as periods during which multiple tasks were conducted simultaneously.

In particular, the table effectively illustrates how key development stages, including AI model development, database design, REST API design, and application development, were carried out in parallel. This is highly important from a project-management perspective, as it helps each team member clearly understand their role, responsibilities, and position within the overall project.

Table 4.1: Development Schedule

Development Category	Details	~5	~6	~7	~8	~9	10
Planning	Topic selection and refinement	○					
	Preliminary research	○					
AI model development	Eye-tracking implementation	○	○				
	GPT API integration		○	○			
	GPT fine-tuning		○	○			
DB design	DB design and construction		○	○			
REST API design	Integration with the DB			○			
	Feature development			○	○		
Application development	REST API integration		○	○	○	○	
	Feature deployment		○	○	○	○	
Deployment and revision	Revision and deployment					○	○

4.2 Division of Responsibilities

The division of responsibilities among the team members is shown in Table 4.2.

Table 4.2: Division of Responsibilities

Student ID	Name	Responsibilities
201924480	Junhyung Park	User registration and login using JWT GPT-4o API logic implementation Top-10 study-time ranking feature Following feature Chatbot frontend UI Deployment automation using Docker and AWS
201924431	Sangyu Kim	Login page Study timer GPT Function Calling Eye-tracker implementation My Page implementation Web page design
201924530	Sangjun Lee	Study domain Friend domain Todo domain Notification feature API Deployment and integration of the Azure Database for MySQL server Screen time API specification document

4.3 API Specification

An API specification is a document that describes the structure and usage of a web service's APIs in detail. It is an important tool that helps developers understand and correctly use APIs. Swagger is an open-source set of tools for API design, documentation, and testing. It enables RESTful APIs to be visually represented and managed.

Swagger was adopted in this project because it provides automated documentation generation, API testing through an intuitive user interface, and improved efficiency during the development process. By using Swagger, API documentation can be kept up to date, and smooth collaboration between frontend and backend developers can be promoted.

- All endpoints are assumed to be accessible only to authenticated users when they

include a `Principal` parameter.

- The detailed structure of request and response bodies is defined in the corresponding DTO classes.
- Error responses are not specified separately but are returned with appropriate HTTP status codes.
- The `/api/v1/users` endpoint redirects the user to the login page after registering a new account.

4.4 ERD

The ERD shown in Figure 4.1 visually represents the database structure of our system. It shows the major entities and the relationships between them and clearly defines the system's data model.

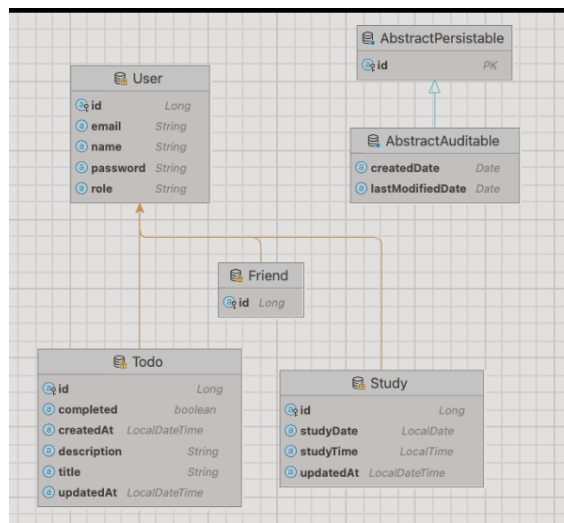


Figure 4.1: Database ERD

4.5 GitHub

The complete source code and related documents for this project can be found in the GitHub repository. The repository contains all components of the project and is used as a platform for continuous development and collaboration through active pull requests and issues.

Repository URL: <https://github.com/new3seagull/SeagullsRoom>

Through this GitHub repository, we aim to share the technical implementation details of our project and promote continuous improvement and development through collaboration with the community.

Chapter 5 Conclusion and Future Work

5.1 Conclusion and Expected Benefits

Through this project, we successfully developed SeagullsRoom, a time-management platform based on real-time eye tracking. This platform, which utilizes an eye tracker and the GPT API, is expected to provide users with an experience that enables them to manage their time efficiently while engaging in various activities, including studying. The main expected benefits are as follows.

SeagullsRoom is expected to establish itself as a comprehensive learning management platform that goes beyond simply measuring study time by utilizing the GPT API to comprehensively improve and optimize users' learning habits. Through continuous improvement and expansion of its functions based on user feedback, the platform will be able to provide a more effective and personalized learning experience in the future.

5.2 Future Research Directions

Based on the implementation of the SeagullsRoom project, the following future research directions are proposed. These research directions aim to expand the platform's functionality and further improve the user experience.

Through these developments, SeagullsRoom will provide users with a richer and more effective learning experience. Rather than remaining a simple tool for managing study time, it will evolve into a comprehensive educational platform that supports and optimizes the user's overall learning journey. Ultimately, it has the potential to improve individual learning outcomes and bring positive changes to the education system as a whole.

As these research directions are realized, the platform will continue to be improved by consistently incorporating user feedback and the requirements of educational environments. Through this process, SeagullsRoom is expected to become the most effective and

innovative learning tool for users in an ever-changing learning environment.

Bibliography

- [1] Jeff Huang, Ryen White, and Georg Buscher. User see, user point: gaze and cursor alignment in web search. In *Proceedings of the sigCHI conference on human factors in computing systems*, pages 1341–1350, 2012.
- [2] Broadcom Inc. Spring security 6.5.0. Online Document. Checked on 2024-10-08.
- [3] Docker Inc. Docker documentation. Online Document. Checked on 2024-10-08.
- [4] Ole Baunbæk Jensen. Webcam eye tracking vs. an eye tracker [pros & cons]. Online Document, 2022. Checked on 2024-10-08.
- [5] Okta. Introduction to JSON web tokens. Online Document. Checked on 2024-10-08.
- [6] Alexandra Papoutsaki, James Laskey, and Jeff Huang. Searchgazer: Webcam eye tracking for remote studies of web search. In *Proceedings of the 2017 conference on conference human information interaction and retrieval*, pages 17–26, 2017.
- [7] Alexandra Papoutsaki, Patsorn Sangkloy, James Laskey, Nediya Daskalova, Jeff Huang, and James Hays. WebGazer: Scalable webcam eye tracking using user interactions. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 3839—3845, 2016.
- [8] Brown WebGazer Team. WebGazer GitHub repository. Online Document, 2016. Checked on 2024-10-08.