**COMP4910 Senior Design Project 1, Fall 2024**

**Advisor: Behçet Melih Sarıbatır**

**AimMaxHit**

**Requirements Specifications Document**

**16.01.2025**

**Revision 2.0**

**By:**

**Efe Demirtaş, 20070001053**

**Kaan Mert Kozalı, 21070001064**

**Umut Özgür İpin, 20070001037**

**Volkan Ege Kılınç, 20070001058**

# Revision History

|  |  |  |
| --- | --- | --- |
| **Revision** | **Date** | **Explanation** |
| 1.0 | 16.11.2024 | Initial requirements, textual form |
| 2.0 | 16.01.2025 | Requirements Model, mostly in UML notation |

Table Of Contents

**1. Introduction**

**2. Requirements List**

**3. Actors and Use Case Diagram**

 3.1. Actors in AimMaxHit

 3.2. Class Diagram

 3.3. Use Case Diagram

 3.4. Activity Diagram

**4. Main User Interface and Functions**

 4.1. Word Meaning and Description

 4.2. Login – Sign Up Page

 4.2.1. Function 1: User Authentication

 4.2.2. Function 2: Real-Time Data Capture

 4.2.3. Function 3: Performance Analysis

 4.2.4. Function 4: Forgot Password

 4.2.5. Function 5: View Gameplay History

 4.2.6. Function 6: Export and Share Data

 4.2.7. Function 7: Notifications and Alerts

 4.2.8. Function 8: Display Leaderboard

 4.2.9. Function 9: In-Game Feedback

 4.2.10. Function 10: Interactive Tutorial Mode

 4.2.11. Function 11: User Settings and Customization

 4.2.12. Function 12: Save Progress and Analytics

 4.3. Main Page

 4.4. Dashboard Screen

**5. Testing: AimMaxHit Application**

 5.1. User Authentication Testing

 5.2. Real-Time Data Testing

 5.3. Performance Analytics Testing

 5.4. Leaderboard and History Testing

**6. Development Environment**

 6.1. Frontend Development

 6.1.1. IDE (Integrated Development Environment)

 6.1.2. Unity UI Toolkit

 6.1.3. Styling and Animation

 6.1.4. Package Manager

 6.2. Backend Development

 6.2.1. Backend Framework (.NET Scripts)

 6.2.2. Database Management System ( Firebase)

 6.2.3. Server Runtime

 6.2.4. API Integration

 6.3. Database Setup

 6.3.1. Database Configuration

 6.3.2. Real-Time Sync and Updates

 6.4. Security Measures

 6.4.1. HTTPS Implementation

 6.4.2. Authentication and Authorization

 6.4.3. Role-Based Access Control

**7. References**

#

|  |  |
| --- | --- |
| Word | Meaning and Description |
| Unity | Unity is a real-time development platform used to create interactive 2D and 3D applications, especially for games. It offers tools for building engaging UI and immersive environments. |
| Firebase | Firebase is a cloud-hosted platform providing tools like real-time databases, authentication, and hosting, ideal for building scalable and secure applications. |
| .NET C# | .NET Core is an open-source framework used for backend operations, providing a robust and scalable environment for developing APIs and handling server-side logic. |
| Python | Python is a versatile programming language used in AimMaxHit for data processing and analysis, enabling the generation of insights and trends from gameplay data. |
| Firebase | Firebase is a NoSQL cloud database within Firebase, used for storing and syncing data in real time, ensuring consistent updates across all devices and users. |
| API | An Application Programming Interface (API) allows communication between the Unity front end and the backend, facilitating functionalities like data retrieval and user interactions. |
| HTTPS | HTTPS (Hypertext Transfer Protocol Secure) ensures encrypted and secure communication between the client and server, protecting sensitive user data in transit. |
| JSON | JSON (JavaScript Object Notation) is a lightweight data-interchange format used to export user performance data in AimMaxHit for external analysis. |
|  |  |

# 1. Introduction

AimMaxHit is an innovative gaming performance analysis platform designed to evaluate, track, and enhance users' gaming abilities. The project focuses on creating a comprehensive system for analyzing real-time gameplay data and providing actionable insights for improvement. By leveraging cutting-edge technologies such as Unity, Firebase, and .NET Core, AimMaxHit delivers a seamless and interactive experience for users.

 **Objective:** The primary objective of AimMaxHit is to empower players by giving them access to detailed performance metrics, including reaction times, accuracy rates, and other critical indicators of gaming skill. These insights help players identify their strengths and weaknesses, allowing them to improve their overall performance.

### **System Overview:**

1. **Data Capture and Storage:**
	1. The system captures real-time gameplay data such as hit accuracy, reaction speed, and movement patterns.
	2. All data is securely stored in Firebase, ensuring fast and reliable access for analysis.
2. **Performance Analysis:**
	1. Using Python-based algorithms, the system processes gameplay data to identify trends, track progress, and recommend areas for improvement.
3. **User Engagement:**
	1. Players can access their detailed game history, view leaderboards, and share their achievements through social media integration.
	2. Notifications and real-time feedback during gameplay keep users engaged and motivated.

### **Gameplay Integration:**

* AimMaxHit integrates seamlessly with Unity-based games, enabling players to track their performance in a variety of gaming scenarios.
* The platform supports both single-player and multiplayer modes, ensuring flexibility and scalability for different types of users.

### **Strategy and Vision:**

AimMaxHit is designed to be an accessible and user-friendly tool for gamers of all skill levels. Whether users are casual players seeking to understand their performance or competitive gamers looking for an edge, the platform caters to their needs through advanced analytics and intuitive design. The project also envisions expanding into more complex gaming metrics and genres, providing even greater value to its users in the future.

#

# 2. Requirements List

|  |  |  |
| --- | --- | --- |
| Use Case Number | Use Case Name | Short Explanation |
| 1 | User Authentication | Authenticate users with email and password using Firebase. |
| 2 | Real-Time Data Capture | Record and sync live gameplay data such as accuracy and reaction times. |
| 3 | Performance Analysis | Analyze user gaming data and provide actionable insights. |
| 4 | Password Recovery | Allow users to recover their password through a forgot password functionality. |
| 5 | Gameplay History | Allow users to review past game sessions with detailed performance metrics. |
| 6 | Export and Share Data | Allow users to download performance data and share achievements. |
| 7 | Notifications | Send real-time alerts and updates to keep users engaged. |
| 8 | Display Leaderboard | Showcase rankings of top-performing players globally and locally. |
| 9 | In-Game FeedBack | Provide interactive visual and audio feedback during gameplay. |
| 10 | Tutorial Mode | Include a tutorial mode to help users understand how to use the system. |
| 11 | User Settings and Customization | Allow users to modify preferences such as themes or notifications. |
| 12 | Save Progress and Analytics | Store user data and analytics securely in the database. |

|  |  |  |
| --- | --- | --- |
| No | Requirements | Use Case(s) |
| 1 | Process user sign-in with credentials verification. | User Authentication |
| 2 | Capture and sync real-time game performance metrics. | Real-Time Data Capture |
| 3 | Analyze gameplay data for trends and improvement suggestions. | Performance Analytics |
| 4 | Provide a secure method for users to reset forgotten passwords. | Password Recovery |
| 5 | Enable access to a log of previous game sessions and performance metrics. | Gameplay History |
| 6 | Allow users to export performance data as CSV or JSON. | Export and Share Data |
| 7 | Notify users about game milestones, updates, and achievements. | Notifications |
| 8 | Display top players' rankings with filtering options. | Display Leaderboard |
| 9 | Provide live feedback to users during gameplay. | In-Game Feedback |
| 10 | Create an interactive guide for new users to learn the system. | Tutorial Mode |
| 11 | Allow users to customize their settings and preferences. | User Settings and Customization |
| 12 | Store user progress and analytics securely in Firebase. | Save Progress and Analytics |

# 3. Actors and Use Case Diagram

# 3.1. Actors in AimMaxHit

|  |  |
| --- | --- |
| Actor | Description |
| Registered Player | A user who has created an account and can access the full suite of application features. |
| Administrator | An individual responsible for managing user accounts, monitoring system performance, and maintaining data integrity. |
| Data Analyst | A user tasked with evaluating game data for trends and recommending areas for improvement. |
| Firebase | The backend system responsible for user authentication, data storage, and notifications. |

Class Diagram :

# Use Case Diagram :

# metin, diyagram, çizgi, yazı tipi içeren bir resim  Açıklama otomatik olarak oluşturuldu

# Activity Diagram :



**Login – Sign Up Page :**

Objective: This interface aims to provide users with a secure and user-friendly authentication process. It also includes a user authentication process to facilitate user registration and login to your web application. Additionally, it allows users to play as guests without the need for registration. Features:

-User Registration: Allows users to register with email, password, username, and date of birth.

-Login Functionality: Enables existing users to log in.

-Change Password Functionality: Provides an option for existing users to change their passwords.

-Play as Guest: Allows users to play the game without the need for registration.

Main Page :

**Objective:**

AimMaxHit provides users with a seamless gaming performance tracking experience. The primary goal is to engage users with detailed analytics while maintaining a simple and intuitive design for accessibility. The platform encourages strategic improvement by offering easy access to key performance metrics and game data.

**Features:**

1. **Ease of Use:**
	* Users can quickly access game analytics and performance metrics with a straightforward and understandable interface.
	* A user-friendly design ensures fast navigation and functionality.
2. **Multiplayer Support:**
	* Allows users to connect with others for dynamic gaming experiences.
	* Options for head-to-head matches with friends or random players globally.
3. **Advanced Data Tracking:**
	* Tracks gameplay performance and history, highlighting areas for improvement.
	* Provides an option for users to analyze data trends and strategies.
4. **Clean Design:**
	* Relaxing and modern color palette ensures a visually appealing experience.
	* Fully optimized for smooth operation on both mobile and desktop devices.
5. **Game Statistics:**
	* Offers insights into performance metrics such as wins, losses, and accuracy.
	* Summarizes key statistics at the end of each game session.

**4.2.1. Function 1: User Authentication**

Authenticate users with their email and password or allow new users to register.

**Inputs:**

* **Username:** [User enters their username]
* **Password:** [User enters their password]

**Buttons:**

* **Sign In**
* **Register**

**4.2.2. Function 2: Real-Time Data Capture**

Capture and sync user gameplay data such as accuracy and reaction times.

**Features:**

* Track gameplay metrics in real-time.
* Securely store data in Firebase for analysis.

**4.2.3. Function 3: Performance Analysis**

Analyze gameplay metrics and provide actionable insights.

**Features:**

* Advanced algorithms to process data.
* Generate user-specific improvement suggestions.

**4.2.4. Function 4: Forgot Password**

Enable users to reset their passwords securely.

**Features:**

* Email-based password recovery system.
* Link sent to reset password.

**Inputs:**

* Email: [User enters registered email].

**4.2.5. Function 5: View Gameplay History**

Display users' past gameplay sessions and performance details.

**Features:**

* Show session dates, scores, and trends over time.

**4.2.6. Function 6: Export and Share Data**

Allow users to download and share their performance data.

**Features:**

* Export options for CSV or JSON formats.
* Social media sharing for achievements.

**4.2.7. Function 7: Notifications and Alerts**

Keep users informed with real-time notifications.

**Features:**

* Achievement milestones.
* System updates and reminders.

**4.2.8. Function 8: Display Leaderboard**

Show top players’ rankings with detailed statistics.

**Features:**

* Global and local leaderboards.
* Filters for regions and game modes.

**4.2.9. Function 9: In-Game Feedback**

Provide real-time feedback during gameplay to guide improvement.

**Features:**

* Visual and audio cues for user actions.
* Progress tracking during the game.

**4.2.10. Function 10: Interactive Tutorial Mode**

Introduce new users to the system with an interactive guide.

**Features:**

* Step-by-step tutorials for gameplay and features.

**4.2.11. Function 11: User Settings and Customization**

Enable users to modify their preferences for a personalized experience.

**Features:**

* Change themes, notification settings, and display preferences.

**4.2.12. Function 12: Save Progress and Analytics**

Store user progress and analytics for long-term tracking.

**Features:**

* Automatically sync data with Firebase.
* Allow multi-device access to user history.

### **4.3. Main Page**

**Objective:**

Provide a centralized interface for accessing all core features, such as starting a game, viewing leaderboards, or checking progress.

**Features:**

* Simple navigation for all functionalities.
* Highlighted sections for important updates or achievements.

### **4.4. Dashboard Screen**

**Objective:**

Give users a quick overview of their performance and system updates.

**Features:**

* Summarize key statistics (accuracy, reaction time, win/loss ratios).
* Show recent notifications and upcoming challenges.

**5. Testing: AimMaxHit Application**

The testing phase is essential to ensure the functionality, reliability, and user-friendliness of the AimMaxHit platform. Below are the key testing scenarios:

5.1. User Authentication Testing

* **Scenario 1: Sign-In Validation**
	+ **Steps:**
		1. Enter a valid username and password.
		2. Click the "Sign In" button.
	+ **Expected Result:**
		1. Successful login, granting access to the system.
* **Scenario 2: Registration**
	+ **Steps:**
		1. Click the "Register" button.
		2. Fill in the required registration details.
		3. Click the "Register" button.
	+ **Expected Result:**
		1. Successful registration, creating a new user account.
* **Scenario 3: Incorrect Credentials Handling**
	+ **Steps:**
		1. Enter an invalid username or password.
		2. Click the "Sign In" button.
	+ **Expected Result:**
		1. Display an error message and prompt the user to enter correct credentials.

5.2. Game Functionality Testing

* **Scenario 1: Solo Game Initialization**
	+ **Steps:**
		1. Click the "Start Game" button.
	+ **Expected Result:**
		1. A solo game session starts, recording performance metrics in real-time.
* **Scenario 2: Exporting Gameplay Data**
	+ **Steps:**
		1. Access the "Export Data" section
		2. Choose a file format (CSV or JSON) and click "Export."

**Expected Result:**

* + 1. A downloadable file containing performance data is generated.
* **Scenario 4: View Leaderboard**
	+ **Steps:**
		1. Navigate to the "Leaderboard" section.
	+ **Expected Result:**
		1. Display a leaderboard showing top player rankings and scores.

5.3. Gameplay Testing

* **Scenario 1: Data Capture Analysis**
	+ **Steps:**
		1. Play a game session.
		2. Allow the system to capture performance metrics like accuracy and reaction times.
	+ **Expected Result:**
		1. Real-time data capture and storage in Firebase for further analysis.
* **Scenario 2: Notifications**
	+ **Steps:**
		1. Achieve a milestone or receive a system update.
	+ **Expected Result:**
		1. Display a real-time notification for the event.
* **Scenario 3: Viewing Game History**
	+ **Steps:**
		1. Access the "View Game History" section.
	+ **Expected Result:**
		1. Display a log of previous games and relevant performance metrics.
* **Scenario 4: Tutorial Mode**
	+ **Steps:**
		1. Access the "Tutorial mode” section.
	+ **Expected Result:**
		1. An interactive tutorial explains the system and gameplay features.

6.1. Development Environment

* **IDE (Integrated Development Environment):**
	+ Unity Editor for interface design and development.
* **Version Control:**
	+ Github for version control and team collaboration.
* **UI Framework:**
	+ Unity UI Toolkit for building dynamic user interfaces.
* **Styling:**
	+ Use Unity’s styling options and animations for enhanced user experience.
* **Package Manager:**
	+ Unity Package Manager for managing dependencies and third-party libraries.

6.1.2. Backend Development

* **Backend Framework:**
	+ .NET Core for robust and scalable server-side operations.
* **Database Management System:**
	+ Firebase for storing user data, performance metrics, and analytics.
* **Server Runtime:**
	+ Firebase for handling real-time communication (e.g., WebSocket-based multiplayer).
* **API Integration:**
	+ Develop and integrate RESTful APIs for seamless communication between the Unity frontend and Firebase backend.

6.1.3. Database Setup

* **Database Configuration:**
	+ Configure Firebase settings, including secure access rules and indexing for fast queries.
* **Database Migrations:**
	+ Use Firebase’s data migration tools for schema updates and data management.

6.1.4. Security Measures

* **HTTPS Implementation:**
	+ Ensure all communication between the client and server is encrypted using HTTPS.
* **Authentication and Authorization:**
	+ Use Firebase Authentication for secure user login and access control.

**7. References**

During the development of the AimMaxHit gaming analytics platform, the following references were consulted to ensure best practices, security measures, and optimal implementation of features:

1. **Unity Documentation**: Unity Documentation for developing interactive 3D interfaces and implementing game mechanics.
2. **Firebase Documentation**: Firebase Documentation for authentication, Firebase database, and real-time data handling.
3. **.NET Core Documentation**: [.NET Core Documentation](https://learn.microsoft.com/en-us/dotnet/core/) for backend development and API integrations.
4. **Python Documentation**: [Python Documentation](https://docs.python.org/3/) for data analysis and processing algorithms.
5. **GitHub Documentation**: [GitHub Documentation](https://docs.github.com/) for best practices and workflows in version control and collaboration.
6. **Node.js Documentation**: Node.js Documentation for implementing real-time communication and server-side operations.
7. **REST API Guidelines**: [REST API Design Guidelines](https://restfulapi.net/) for creating secure and efficient endpoints.