

**YAŞAR UNIVERSITY**

**FACULTY OF ENGINEERING**

**DEPARTMENT OF COMPUTER ENGINEERING**

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

**Advisor: Dr. Behçet Melih SARIBATIR**

**AimMaxHit: Game Performance Analyzer**

**Final Report**

**20.01.2025**

**By:**

**Efe DEMİRTAŞ, Student ID: 20070001053**

**Kaan Mert KOZALI, Student ID: 21070001064**

**Umut Özgür İPİN, Student ID: 20070001037**

**Volkan Ege KILINÇ, Student ID: 20070001058**

# PLAGIARISM STATEMENT

This report was written by the group members and in our own words, except for quotations from published and unpublished sources which are clearly indicated and acknowledged as such. We are conscious that the incorporation of material from other works or a paraphrase of such material without acknowledgement will be treated as plagiarism according to the University Regulations. The source of any picture, graph, map or other illustration is also indicated, as is the source, published or unpublished, of any material not resulting from our own experimentation, observation or specimen collecting.

**Project Group Members:**

|  |  |  |  |
| --- | --- | --- | --- |
| Name, Lastname | Student Number | Signature | Date |
| Efe DEMİRTAŞ | 20070001053 |  | 20.01.2025 |
| Kaan Mert KOZALI | 21070001064 |  | 20.01.2025 |
| Umut Özgür İPİN | 20070001037 |  | 20.01.2025 |
| Volkan Ege KILINÇ | 20070001058 |  | 20.01.2025 |

**Project Advisors:**

|  |  |  |  |
| --- | --- | --- | --- |
| Name, Lastname | Department | Signature | Date |
| Behçet Melih Sarıbatır | Computer Engineering |  | 20.01.2025 |
|  |  |  |  |

# ACKNOWLEDGEMENTS

First and foremost, we would like to thank our supervisor, Behçet Melih Sarıbatır, for their invaluable guidance, insightful feedback, and constant encouragement throughout the duration of this project. Their expertise and support played a vital role in shaping our ideas and ensuring the project's progress.  
  
Finally, we extend our appreciation to our team members, Efe Demirtaş, Kaan Mert Kozalı, Umut Özgür İpin, and Volkan Ege Kılınç, for their hard work, dedication, and exceptional collaboration. Each member’s unique skills and commitment made this project possible, and it was a pleasure to work together as a team.

Thank you to everyone who believed in us and contributed to the success of this endeavor.

# KEYWORDS

Game Performance Analysis,

Unity Game Development,

Firebase Integration,

Real-time Analytics,

eSports Training,

Performance Metrics,

Gaming Analytics,

User Authentication,

Data Visualization,

Performance Tracking

# ABSTRACT

AimMaxHit is a cutting-edge performance analytics platform designed to help gamers, eSports professionals, and gaming enthusiasts elevate their gameplay to the next level. The platform offers a comprehensive set of tools for tracking, analyzing, and improving gaming performance by leveraging advanced data analysis techniques and real-time feedback mechanisms. Built using Unity for the frontend, Firebase for backend services, and incorporating various performance metrics, AimMaxHit provides users with detailed insights into their gaming performance, including accuracy rates, reaction times, and overall progress tracking. The system features user authentication, real-time data synchronization, and an intuitive interface for visualizing performance metrics. This project demonstrates the successful integration of modern gaming technology with performance analytics, creating a valuable tool for the gaming community.

# ÖZET

AimMaxHit, oyuncuların, eSports profesyonellerinin ve oyun tutkunlarının oyun performanslarını bir üst seviyeye çıkarmalarına yardımcı olmak için tasarlanmış modern bir performans analiz platformudur. Platform, gelişmiş veri analizi teknikleri ve gerçek zamanlı geri bildirim mekanizmalarını kullanarak oyun performansını takip etmek, analiz etmek ve geliştirmek için kapsamlı bir araç seti sunmaktadır. Frontend için Unity, backend servisleri için Firebase kullanılarak geliştirilen ve çeşitli performans metriklerini içeren AimMaxHit, kullanıcılara isabet oranları, tepki süreleri ve genel ilerleme takibi dahil olmak üzere oyun performansları hakkında detaylı bilgiler sağlamaktadır. Sistem, kullanıcı kimlik doğrulama, gerçek zamanlı veri senkronizasyonu ve performans metriklerini görselleştirmek için sezgisel bir arayüz sunmaktadır. Bu proje, modern oyun teknolojisinin performans analitiği ile başarılı bir şekilde entegrasyonunu göstererek, oyun topluluğu için değerli bir araç oluşturmaktadır.

# TABLE OF CONTENTS

[PLAGIARISM STATEMENT ii](#_Toc153213499)

[ACKNOWLEDGEMENTS iii](#_Toc153213500)

[KEYWORDS iv](#_Toc153213501)

[ABSTRACT v](#_Toc153213502)

[ÖZET vi](#_Toc153213503)

[TABLE OF CONTENTS vii](#_Toc153213504)

[LIST OF FIGURES viii](#_Toc153213505)

[LIST OF TABLES ix](#_Toc153213506)

[LIST OF ACRONYMS/ABBREVIATIONS x](#_Toc153213507)

[1. INTRODUCTION 1](#_Toc153213508)

[1.1. Description of the Problem 1](#_Toc153213509)

[1.2. Project Goal(s) 1](#_Toc153213510)

[1.3. Project Outputs/Deliverables 2](#_Toc153213511)

[2. SURVEY OF RELATED WORK 2](#_Toc153213512)

[3. REQUIREMENTS 2](#_Toc153213513)

[4. DESIGN 3](#_Toc153213514)

[4.1. High Level Design 3](#_Toc153213515)

[4.2. Detailed Design 3](#_Toc153213516)

[4.3. Realistic Restrictions and Conditions in the Design 3](#_Toc153213517)

[5. IMPLEMENTATION AND TESTS 3](#_Toc153213518)

[5.1. Implementation of the Product 3](#_Toc153213519)

[5.2. Tests and Results of Tests 3](#_Toc153213520)

[6. PROJECT MANAGEMENT 4](#_Toc153213521)

[6.1. Project Plan 4](#_Toc153213522)

[6.2. Project Effort/Manpower 4](#_Toc153213523)

[6.3. Project Cost Analysis 4](#_Toc153213524)

[7. CONCLUSIONS 4](#_Toc153213525)

[7.1. Summary 4](#_Toc153213526)

[7.2. Benefits of the Project 4](#_Toc153213527)

[7.3. Future Work 4](#_Toc153213528)

[REFERENCES 5](#_Toc153213529)

[APPENDICES 6](#_Toc153213530)

[APPENDIX A: REQUIREMENTS SPECIFICATION DOCUMENT 7](#_Toc153213531)

[APPENDIX B: DESIGN SPECIFICATION DOCUMENT 18](#_Toc153213532)

[APPENDIX C: PROJECT MANAGEMENT DOCUMENTS 29](#_Toc153213533)

[APPENDIX C1: PROJECT PLAN 29](#_Toc153213534)

[APPENDIX C2: PROJECT EFFORT LOG- CONSOLIDATED 31](#_Toc153213535)

[APPENDIX C3: PROJECT EFFORT LOGS FOR EACH TEAM MEMBER- 33](#_Toc153213536)

# LIST OF ACRONYMS/ABBREVIATIONS

API - Application Programming Interface

CSS - Cascading Style Sheets

FPS - First Person Shooter

HTML - HyperText Markup Language

HTTP - Hypertext Transfer Protocol

HTTPS - Hypertext Transfer Protocol Secure

IDE - Integrated Development Environment

JSON - JavaScript Object Notation

SDK - Software Development Kit

SQL - Structured Query Language

UI - User Interface

UML - Unified Modeling Language

UX - User Experience

XML - Extensible Markup Language

# 1. INTRODUCTION

# 1.1. Description of the Problem

The competitive gaming industry has experienced significant growth in recent years, with eSports gaining mainstream popularity. With this growth, there has been an increasing need for players to analyze and improve their performance. However, the current market lacks comprehensive, data-driven tools that provide detailed analysis and user-friendly feedback for gamers.

Key problems include:

1. Performance Measurement Deficiency:

- Players lack access to crucial metrics like aim precision, reaction time, and overall performance metrics

- Existing tools are often limited to specific games or platforms

- Real-time performance tracking and analysis is inadequate

2. Data Analysis and Visualization:

- Insufficient transformation of performance data into meaningful insights

- Limited user-friendly data visualization tools

- Inadequate progress tracking and comparative analysis capabilities

3. Accessibility and Usability:

- Existing solutions are often complex and difficult to use

- Limited platform-independent solutions

- Lack of real-time feedback mechanisms

4. Data Security and Management:

- Secure storage and management of user data

- Reliable synchronization of performance data

- Protection of user privacy

# 1.2. Project Goal(s)

The main objectives of AimMaxHit are:

1. To develop a comprehensive gaming performance analytics platform that provides real-time feedback and detailed performance metrics

2. To create an intuitive and user-friendly interface for tracking and analyzing gaming performance

3. To implement secure user authentication and data storage using Firebase

4. To provide actionable insights that help players improve their gaming skills

5. To create a scalable platform that can support multiple users and maintain performance data over time

# 1.3. Project Outputs/Deliverables

The key deliverables of the project include:

1. A fully functional Unity-based game interface with performance tracking capabilities

2. Firebase integration for user authentication and real-time data storage

3. Performance analytics dashboard showing key metrics such as accuracy, reaction time, and progress

4. User profile management system

5. Real-time leaderboard system

6. Comprehensive documentation including technical specifications and user guides

# 2. SURVEY OF RELATED WORK

In recent years, several gaming analytics platforms have emerged to help players improve their performance. However, most existing solutions focus on specific games or platforms, limiting their usefulness across different gaming scenarios. Some notable examples include:

1. Aim Lab: A popular aim trainer that focuses primarily on first-person shooter (FPS) games

2. KovaaK's: Another FPS aim trainer with detailed analytics

3. Game-specific analytics tools built into games like CS:GO and Valorant

AimMaxHit differentiates itself by offering:

- A platform-agnostic approach to performance analytics

- Real-time feedback and performance tracking

- Integration with modern cloud technologies for scalability

- Comprehensive user profile management and progress tracking

# 3. REQUIREMENTS

The requirements for our project were primarily determined during the Fall semester as part of COMP 4910. The initial requirements were documented in the form of a Project Assignment Form (PAF), which included essential project information such as the project code, title, team details, and a brief summary of the project goals and product concept. Following the PAF, the first version of the Requirements Specification Document (RSD v1.0) was created in a structured textual format, detailing the initial set of functional and non-functional requirements.

Later, these requirements were refined into RSD v2.0, which involved revising and structuring the requirements more clearly and representing them using UML diagrams. This version aimed to improve clarity and usability for readers and served as the foundation for our high-level design.

The final requirements for COMP 4910 are provided in Appendix A: Requirements Specifications Document, v2.0, which contains a comprehensive list of requirements structured to align with the project’s objectives and standards. This document serves as the definitive reference for the requirements finalized in the Fall semester.

During the Spring semester, as we progress with the design and implementation phases in COMP 4920, we anticipate revising the RSD to address various potential changes:

-Incorporating new ideas and features based on team discussions and advisor feedback.

-Removing or deprioritizing requirements that are deemed impractical or overly resource-intensive to implement within the semester’s constraints.

-Clarifying ambiguous or incomplete requirements to ensure that they are easily understandable and actionable.

-Restructuring poorly organized sections of the RSD to conform to the standards of COMP 4910 and COMP 4920, improving overall document quality and readability.

This iterative approach to refining requirements ensures the RSD remains a relevant and accurate guide throughout the project lifecycle. All updates to the RSD will be documented and referenced accordingly in future versions.

3.1. Functional Requirements

1. User Management:

- User registration and authentication using Firebase

- Profile customization and management

- Password recovery functionality

2. Performance Tracking:

- Real-time capture of gameplay metrics

- Accuracy and reaction time measurement

- Score tracking and history

- Performance trend analysis

3. Analytics Dashboard:

- Visual representation of performance metrics

- Historical data comparison

- Progress tracking over time

- Leaderboard integration

4. Game Interface:

- Interactive target practice environment

- Customizable difficulty levels

- Real-time feedback during gameplay

- Score and accuracy display

3.2. Non-Functional Requirements

1. Performance:

- Real-time data processing with minimal latency

- Smooth gameplay experience at 60+ FPS

- Quick database response times

2. Security:

- Secure user authentication

- Encrypted data transmission

- Protected user information

3. Usability:

- Intuitive user interface

- Clear performance feedback

- Easy navigation between features

- Responsive design

4. Scalability:

- Support for multiple concurrent users

- Efficient data storage and retrieval

- Cloud-based infrastructure

# 4. DESIGN

# 4.1. High Level Design

Our high-level design focuses on a modular architecture comprising a Unity-based front-end, a Firebase-powered backend, and data analytics components. This architecture was chosen for its scalability, real-time data handling capabilities, and seamless integration with gaming platforms. Unity provides a robust framework for developing an interactive and engaging user interface, while Firebase ensures efficient data storage, real-time updates, and reliable performance analytics.

The high-level design specification, including details of system components, communication between the front-end and backend, and database structure, is provided in Appendix B: Design Specifications Document. Specific sections of the appendix outline the data flow, API interactions, and backend configuration that support the functionality and performance of the system

# 4.2. Detailed Design

Database Design Refinement:  
- Further optimization of the Firebase database structure for scalability, ensuring efficient real-time data handling for large user bases.

- Implementation of advanced analytics models for player performance tracking and historical data analysis.

2. Advanced Gameplay Mechanics:

- Integration of complex gameplay features such as dynamic difficulty adjustment and user customization options.

- Enhanced interaction systems to improve user engagement and gameplay experience.

3. Backend Enhancements:

- Advanced API development for seamless communication between the Unity front-end and Firebase backend.

- Implementation of security measures, including user authentication and data encryption, to safeguard user data.

4. User Interface (UI) and Experience (UX):

-Design improvements based on user feedback to make the interface more intuitive and visually appealing.

-Development of additional features such as in-game tutorials and user settings customization.

5. Testing and Quality Assurance:

-Comprehensive testing of all system components to identify and resolve potential issues.

-Performance testing to ensure system reliability under various conditions.

# 4.3. Realistic Restrictions and Conditions in the Design

1. Technical Limitations:

- Network latency affecting real-time updates

- Processing power requirements for smooth gameplay

- Storage limitations for user data

2. Security Considerations:

- Data privacy requirements

- Authentication security

- Anti-cheat measures

3. Scalability Constraints:

- Database performance under heavy load

- Concurrent user limitations

- Resource optimization requirements

# 5. IMPLEMENTATION AND TESTS

# 5.1. Implementation of the Product

The implementation of AimMaxHit was carried out in several phases:

1. Unity Development:

- Implementation of player controls and camera system

- Development of target system and shooting mechanics

- Creation of scoring and timing systems

- Integration of accuracy calculation and performance metrics

- Development of user interface elements

2. Firebase Integration:

- Setup of Firebase project and configuration

- Implementation of authentication system

- Development of real-time database structure

- Integration of data synchronization mechanisms

3. Performance Analytics:

- Implementation of performance tracking systems

- Development of analytics dashboard

- Creation of leaderboard system

- Integration of real-time feedback mechanisms

# 5.2. Tests and Results of Tests

1. User Authentication Testing:

- Successful user registration and login

- Password recovery functionality

- User profile management

- Results: All authentication features working as expected

2. Game Mechanics Testing:

- Player movement and camera control

- Target hit detection

- Scoring system accuracy

- Timer functionality

- Results: Core gameplay mechanics functioning properly

3. Performance Testing:

- Frame rate stability

- Network latency

- Database response times

- Results: System maintains stable performance under normal load

4. Integration Testing:

- Unity-Firebase communication

- Real-time data synchronization

- Leaderboard updates

- Results: Successful integration with minor optimization needs

# 6. PROJECT MANAGEMENT

# 6.1. Project Plan

The project was executed in the following phases:

Week 1-2:

- Project initialization and team organization

- Technology stack selection

- Initial documentation

Week 3-4:

- Unity development setup

- Basic gameplay mechanics implementation

- Firebase integration planning

Week 5-6:

- Firebase setup and configuration

- User authentication implementation

- Database structure development

Week 7-8:

- Core gameplay features development

- Performance tracking implementation

- Initial testing phase

Week 9-10:

- Integration of all components

- User interface refinement

- System optimization

Week 11-12:

- Comprehensive testing

- Bug fixes and improvements

- Documentation completion

# 6.2. Project Effort/Manpower

The development of AimMaxHit is the result of dedicated teamwork and the combined efforts of four Computer Engineering students from Yaşar University. Each team member contributed their unique expertise to bring the project to life:

* **Kaan Mert Kozalı** focused on backend development and real-time data processing.
* **Efe Demirtaş** handled Unity development, creating an engaging and interactive user interface.
* **Umut Özgür İpin** contributed to both Unity and backend development, ensuring seamless integration.
* **Volkan Ege Kılınç** worked on Firebase integration, optimizing data management and scalability.

This collaboration required extensive planning, coding, and testing to ensure AimMaxHit delivers a user-friendly and reliable experience. For more details about the project, visit our website: [www.aimmaxhit.com](http://www.aimmaxhit.com).

# 6.3. Project Cost Analysis

Development Resources:

1. Software Tools:

- Unity (10$ Premium Chair Pack)

- Firebase (Free Tier)

- Visual Studio (Community Edition)

- Git for version control

- Website(Domain + Premium Editing Tool) = 1080 Turkish Lira

2. Hardware Requirements:

- Development computers

- Testing devices

- Network infrastructure

3. Human Resources:

- 4 team members

- 1 project advisor

- Weekly meetings and reviews

# 7. CONCLUSIONS

# 7.1. Summary

AimMaxHit has successfully achieved its primary objectives:

- Created a functional gaming performance analytics platform

- Implemented secure user authentication and data storage

- Developed real-time performance tracking and feedback systems

- Established a scalable and maintainable architecture

# 7.2. Benefits of the Project

The project provides several key benefits:

1. For Users:

- Improved gaming performance tracking

- Real-time feedback and analytics

- Progress monitoring and comparison

2. For Development Team:

- Practical experience with modern technologies

- Understanding of large-scale system integration

- Experience in agile development practices

# 7.3. Future Work

Potential areas for future development:

1. Additional Features:

- Advanced analytics algorithms

- Machine learning-based performance prediction

- More customization options

2. Technical Improvements:

- Enhanced performance optimization

- Additional platform support

- Advanced anti-cheat measures

3. User Experience:

- More detailed feedback systems

- Additional training scenarios

- Social features and community integration

# REFERENCES

[1] Unity Technologies. (2024). Unity User Manual. Retrieved from https://docs.unity3d.com/Manual/

[2] Firebase Documentation. (2024). Firebase Guides. Retrieved from https://firebase.google.com/docs

[3] Microsoft. (2024). .NET Documentation. Retrieved from https://docs.microsoft.com/en-us/dotnet/

[4] Sarıbatır, B. M. (2024). COMP4910-RulesRegulations-v4.3. Yaşar University.

[5] Unity Technologies. (2024). Unity Asset Store. Retrieved from https://assetstore.unity.com/

# APPENDICES

# APPENDIX A: REQUIREMENTS SPECIFICATION DOCUMENT

**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 | 16.01.2024 | 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, 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 :metin, ekran görüntüsü, diyagram, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldu

# Use Case Diagram :

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

# Activity Diagram :

ekran görüntüsü, diyagram, tasarım içeren bir resim

Açıklama otomatik olarak oluşturuldu

**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.

# APPENDIX B: DESIGN SPECIFICATION DOCUMENT

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

**Advisor: Behçet Melih SARIBATIR**

**AimMaxHit**

**Design Specifications Document**

**Revision 1.0**

**20.01.2025**

**By:**

**Efe Demirtaş, 20070001053**

**Kaan Mert Kozalı, 21070001064**

**Umut Özgür İpin, 20070001037**

**Volkan Ege Kılınç, 20070001058**

# Revision History

|  |  |  |
| --- | --- | --- |
| **Revision** | **Date** | **Explanation** |
| 1.0 | 20.01.2025 | Initial high level design |

# Table of Contents

[Revision History 2](#_Toc60144428)

[Table of Contents 3](#_Toc60144429)

[1. Introduction 4](#_Toc60144430)

[2. AIMMAXHIT System Design 4](#_Toc60144431)

[3. AIMAXHIT Software Subsystem Design 6](#_Toc60144437)

[3.1. AIMMAXHIT Software System Architecture 6](#_Toc60144438)

[3.2. AIMMAXHIT Software System Structure 7](#_Toc60144439)

[3.3. AIMMAXHIT Software System Environment 7](#_Toc60144440)

[4. AIMMAXHIT Software System Detailed Design: 9](#_Toc60144441)

[4.1. AIMMAXHIT Main Module/Class 8](#_Toc60144442)

[4.2. AIMMAXHIT Subsystem S1 Classes 8](#_Toc60144443)

[4.2.1. Class S1-C1 8](#_Toc60144444)

[4.2.2. Class S1-C2 8](#_Toc60144445)

[4.2.3. Class S1-C3, etc. 9](#_Toc60144446)

[4.3. AIMMAXHIT Subsystem S2 Classes 9](#_Toc60144447)

[4.3.1. Class S2-C1 9](#_Toc60144448)

[4.3.2. Class S2-C2 9](#_Toc60144449)

[4.3.3. Class S2-C3, etc. 10](#_Toc60144450)

[4.4. AIMMAXHIT Subsystem S3 Classes 10](#_Toc60144451)

[4.n. AIMMAXHITSubsystem Sk- Common Infrastructure Classes 10](#_Toc60144452)

[5. Testing Design 10](#_Toc60144453)

[References 10](#_Toc60144454)

# 1. Introduction

AimMaxHit is a gaming analytics platform designed to evaluate and enhance players' skills. The platform combines Unity for the frontend, .NET Core for backend operations, and Firebase for real-time data storage and user authentication. It captures real-time gameplay data, analyzes player performance, and provides actionable insights for improvement.

1. **Login and Registration Operations:**
   * User login
   * User registration
   * Forgot password / password reset
2. **Game Operations:**
   * Start the game
   * Play the game (receive moves from the user)
   * Check the game state
   * End the game
3. **Profile Operations:**
   * View profile
   * Edit profile (name, password, profile picture)
   * View scores
4. **Level and Scoring System:**
   * Level up
   * Scoring system (winners get more points)
   * View levels and scores
5. **Logout and Session Management:**
   * Log out
   * Exit the application
6. **UI Update Operations:**
   * Update the game board
   * Update user information
   * Update scores and levels
7. **Email Confirmation and Security Operations:**
   * Email verification operations
   * Security measures (strong password requirements, use of SSL, etc.)
8. **Error Handling:**
   * Notify users of errors
   * Security checks and error handling

# 2. AIMMAXHIT System Design

AimMaxHit is designed as a modular and scalable gaming analytics platform. Its architecture consists of three main layers:

1. **Presentation Layer**:
   * Built using Unity, providing an interactive and user-friendly interface.
   * Handles user interactions, such as login, gameplay, and viewing analytics.
2. **Application Layer**:
   * Developed with .NET Core to manage the business logic and system processes.
   * Handles real-time data capture, analytics processing, and leaderboard updates.
3. **Data Layer**:
   * Firebase offers secure, scalable, and real-time data storage.
   * Stores user profiles, gameplay data, performance metrics, and system logs.

**3. AIMMAXHIT Software Subsystem Design**

The system comprises the following subsystems:

1. **User Management Subsystem**:
   * Authentication via Firebase.
   * Handles user profiles, password reset, and email verification.
2. **Gameplay Subsystem**:
   * Manages game sessions, including start, play, and end states.
   * Captures real-time performance metrics.
3. **Analytics Subsystem**:
   * Processes gameplay data to generate performance insights.
   * Tracks user progress, levels, and scoring.
4. **Leaderboard Subsystem**:
   * Updates and displays rankings based on player scores.
   * Filters data for global or friend-specific leaderboards.
5. **Notification Subsystem**:
   * Sends real-time updates for achievements, system changes, or reminders.

# 3.1. AIMMAXHIT Software System Architecture

The architecture of AimMaxHit is designed to be modular, scalable, and maintainable, supporting real-time gameplay analytics and user interactions. The system architecture consists of three primary layers:

**1. Presentation Layer**

* **Technology Used**: Unity
* **Description**:
  + The presentation layer provides an interactive and visually engaging interface for the users.
  + It handles user interactions, such as login, registration, gameplay initiation, and data visualization.
  + Responsible for real-time updates to the user interface, such as gameplay data, leaderboard rankings, and notifications.
* **Key Components**:
  + **Login and Registration Interface**: Enables secure authentication and onboarding of users.
  + **Game Dashboard**: Displays key metrics, performance insights, and options for initiating gameplay.
  + **Game Board**: The core interface for gameplay, capturing user moves and providing visual feedback.

**2. Application Layer**

* **Technology Used**: .NET Core
* **Description**:
  + The application layer acts as the business logic engine, processing user requests and managing system operations.
  + It connects the frontend (Unity) with the backend (Firebase) through APIs.
  + Real-time gameplay data is processed here for analytics and leaderboard updates.
* **Key Responsibilities**:
  + **Game Session Management**: Manages game lifecycle, including initiation, real-time data capture, and conclusion.
  + **Performance Analytics**: Processes and analyzes gameplay data to generate actionable insights.
  + **Leaderboard Management**: Updates player rankings and provides data to the frontend for display.

**3. Data Layer**

* **Technology Used**: Firebase
* **Description**:
  + The data layer is responsible for storing and retrieving application data in a secure and efficient manner.
  + It ensures real-time synchronization across devices and users.
  + Designed to handle large-scale data storage and rapid retrieval for analytics.
* **Key Responsibilities**:
  + **User Data Management**: Stores user profiles, authentication data, and preferences.
  + **Game Data Storage**: Captures and stores gameplay metrics, session logs, and performance insights.
  + **Leaderboard Data**: Maintains global and friend-specific leaderboards.

**System Communication Flow**

1. **User Action**: Users interact with the Unity interface to log in, initiate a game, or view analytics.
2. **Frontend Request**: Unity sends requests to the .NET Core application layer via RESTful APIs.
3. **Backend Processing**: The application layer processes requests, interacts with Firebase for data retrieval or updates, and applies business logic.
4. **Real-Time Updates**: Firebase ensures real-time synchronization, pushing data updates to the Unity interface.
5. **Feedback to User**: Processed data is visualized in the Unity interface, providing users with immediate feedback.

**Security Measures**

* **Authentication**: Firebase Authentication ensures secure user access.
* **Data Encryption**: All communication between layers is encrypted using HTTPS.
* **Role-Based Access Control (RBAC)**: Limits access to administrative functionalities and sensitive data.

**Scalability Features**

* Firebase supports horizontal scaling for data storage and retrieval.
* .NET Core is optimized for high-concurrency scenarios, ensuring efficient request handling.
* Unity's modular design allows for easy updates and feature additions.

3.2. AIMMAXHIT Software System Structure

**3.2.1. Unity and C# Components**

*Package/Component Structure:*

* Unity Package: Represents the front-end environment for designing and implementing the user interface.
* C# Scripts Package: Contains scripts to manage user interactions, gameplay logic, and client-side calculations.

Discussion and Justification:

* Unity provides a robust framework for building an interactive and visually appealing gaming interface.
* The modular C# scripts ensure maintainability and scalability of gameplay mechanics.

**3.2.2. .NET Core Components**

*Package/Component Structure:*

* **Controllers Package: Handles API requests and business logic for AimMaxHit.**
* **Services Package**: Encapsulates reusable logic such as data processing and analytics generation.
* **Data Access Package**: Manages communication with Firebase for data storage and retrieval.

***Discussion and Justification:***

* *.NET Core's structured approach separates concerns, enhancing code readability and maintainability.*
* *The service-oriented architecture facilitates scalability and modular updates.*
* *Robust backend operations ensure high performance and reliability during gameplay.*

**3.2.3. Firebase Components**

Package/Component Structure:

* Authentication Module: Manages user login, registration, and session handling.
* Firebase Module: Stores real-time data such as gameplay stats, user profiles, and leaderboards.
* Cloud Functions Module: Executes serverless backend logic for tasks like email verification and notifications.

***Discussion and Justification:***

* *Firebase’s real-time database supports instant updates to the front-end.*
* *Its integrated authentication system enhances security and simplifies user management.*
* *Cloud Functions provide scalability and reduce server-side workload.*

**3.2.4. Python and Data Analysis Components**

Package/Component Structure:

* Data Processing Package: Implements Python scripts for analyzing gameplay metrics.
* Visualization Package: Generates insights and graphs for user performance.
* Machine Learning Module: (Future) Applies ML algorithms for advanced player recommendations.

***Discussion and Justification:***

* *Python’s simplicity accelerates data analysis and insight generation.*
* *Visualization tools enhance the user experience by presenting actionable feedback.*
* *Modular structure supports future enhancements, such as AI-based analysis.*

**3.2.5. API Component**

*Package/Component Structure:*

* **RESTful API Package:** Defines endpoints for client-server communication, such as login, game session data, and leaderboard updates.

*Discussion and Justification:*

* RESTful APIs enable efficient and stateless communication between the Unity front-end and .NET Core backend.
* This architecture supports scalability and integration with additional services like analytics or notifications.

# 3.3. AIMMAXHIT Software System Environment

**3.3.1. Hardware Environment**

***Description:*** *The AimMaxHit Application is tailored to operate seamlessly on conventional computer hardware, including desktops, laptops, and servers. It is optimized to support gaming peripherals and ensure low-latency operations for a smooth user experience.*

**3.3.2. System Software Environment**

***Description:*** *AimMaxHit is compatible with major operating systems, including:*

* *Windows (10 and later versions).*

**3.3.3. Middleware**

*Middleware integration enhances the application’s functionalities, enabling seamless communication between the front-end (Unity) and back-end (Firebase and .NET Core).*

**3.3.4. Programming Language and Frameworks**

* ***Programming Languages:***
  + ***C#:*** *Used for developing the Unity front-end.*
  + ***.NET Core:*** *Handles the back-end logic and APIs.*
  + ***Python:*** *Utilized for data analysis and performance insights.*
* ***Frameworks:***
  + ***Unity Framework:*** *Creates the interactive user interface and handles gameplay mechanics.*
  + ***Firebase Frameworks:*** *Manages real-time data storage and authentication.*
  + ***.NET Core Framework:*** *Ensures robust server-side operations and API management.*

**3.3.5. Database**

***Database Management System (DBMS):*** *Firebase is chosen for its:*

* *Real-time data synchronization.*
* *Scalability to handle large datasets.*
* *Cloud-based infrastructure ensuring global accessibility.*
* *Seamless integration with other platforms and applications*

**3.3.6. Software Tools**

* ***Development Environment:***
  + *Unity Editor for front-end development.*
  + *Visual Studio for back-end .NET Core development.*
  + *Python IDEs (PyCharm or Jupyter or Google Colab) for data analysis.*
* ***Version Control:*** *Unity Collab and Git is utilized for collaborative development, efficient tracking of code changes, and maintaining a robust version history.*

**3.3.7. UML Deployment Diagram**

*Discussion and Justification:*

* The outlined hardware and software environment ensures broad compatibility across a variety of user devices and operating systems.
* Middleware incorporation facilitates real-time communication and enhances overall application functionality.
* The selection of C#, .NET Core, and Python aligns with modern development practices, promoting efficiency and ease of maintenance.
* Firebase is chosen for its reliability, scalability, and suitability for real-time gaming applications.

# 4. AIMMAXHIT Software System Detailed Design:

Subsequent parts of this section is kept here for completeness and to give you an idea on what you will be doing in terms of detailed design in the context of COMP 4920 course next semester.

For the course COMP 4910,Section 5 will be almost empty, it is sufficient that you write “detailed design will be carried out in the context of COMP 4920 course next semester”.

The detailed design section should be organized in terms of classes of packages/components for a complex software system, for example Section 5.1 will contain all classes/methods of package/component 1, that is subsystem 1, Section 5.2 all classes/methods of package/component 2, that is subsystem 2, etc.

# 4.1 AIMMAXHIT Main Module/Class

The main module acts as the entry point for the system, coordinating initialization, module integration, and high-level operations.

# 4.2. AIMMAXHIT Subsystem S1 Classes

**4.2. AIMMAXHIT Application Class Structures**

**4.2.1. Class S1-C1: User Management**

* **Responsibilities**: Handles user registration, login, password reset, and profile management.
* **Methods**:
  + registerUser(details): Registers a new user with the provided details.
  + login(credentials): Authenticates a user with email and password.
  + resetPassword(email): Sends a password reset link to the user’s email.

**4.2.2. Class S1-C2: Game Mechanics**

* **Responsibilities**: Manages game logic, including bot and multiplayer modes.
* **Methods**:
  + initializeGame(gameType): Starts a new game of the specified type (bot or multiplayer).
  + executeMove(player, position): Processes a player’s move and updates the game state.

**4.2.3. Class S1-C3: Player Profile and Level System**

* **Responsibilities**: Manages player profiles, avatars, and level progression.
* **Methods**:
  + updateProfile(profileDetails): Updates user information.
  + uploadAvatar(image): Uploads a new profile avatar.
  + gainExperience(points): Increments user experience points for level progression.

**4.2.4. Class S1-C4: Game Board and Status Management**

* **Responsibilities**: Manages the game board state and determines game outcomes.
* **Methods**:
  + placeMarker(player, position): Places the player’s marker on the board.
  + checkWinner(): Checks if a player has won or if the game is a draw.

**4.2.5. Class S1-C5: Difficulty Settings and AI**

* **Responsibilities**: Adjusts the difficulty level and manages AI behavior.
* **Methods**:
  + setDifficulty(level): Sets the difficulty level for the game.
  + aiMove(): Calculates and executes the AI’s next move.

**4.2.6. Class S1-C6: Communication and Notifications**

* **Responsibilities**: Manages in-game communication and user notifications.
* **Methods**:
  + sendMessage(message, recipient): Sends a message to another user.
  + notifyUser(event): Sends a notification about a specific game event.

**4.2.7. Class S1-C7: Game Statistics and Analysis**

* **Responsibilities**: Collects and analyzes gameplay data.
* **Methods**:
  + collectStats(gameData): Collects statistics from a completed game.
  + generateReport(): Produces a detailed performance report for the user.

**4.3. AIMMAXHIT Subsystem S2 Classes**

**4.3.1. Class S2-C1: Game Progress Management**

* **Responsibilities**: Manages online game sessions and player matchmaking.
* **Methods:**
  + createOnlineGame(sessionDetails): Sets up a new online game session.
  + matchPlayers(): Matches players for an online game.

**4.3.2. Class S2-C2: Score and Statistics Management**

* **Responsibilities**: Tracks and manages player scores and statistics.
* **Methods**:
  + recordGameResult(gameData): Records the results of a completed game.
  + generateStatistics(): Generates aggregated statistics for players.

**4.3.3. Class S2-C3: Customization and Settings**

* **Responsibilities**: Handles UI customization and user preferences.
* **Methods**:
  + customizeInterface(options): Applies user-defined UI customization settings.
  + updateSettings(settings): Updates application settings based on user preferences.

# 5. Testing Design

**General Test Remarks:**

* **Unit Testing**: Each module, method, and class in the AIMMAXHIT application should undergo unit testing to validate individual functionalities.
* **Integration Testing**: Following unit testing, integration tests should be conducted to ensure that different parts of the application work together seamlessly.
* **User Acceptance Testing**: Critical for assessing the application’s usability and ensuring it meets the end-user requirements.
* **Performance Testing**: To ensure the application performs well under expected load conditions.
* **Security Testing**: Especially crucial for the user management module to safeguard personal and sensitive data.

**References**

During the development of the AIMMAXHIT application, the following references were consulted to ensure best practices, security measures, and optimal coding standards:

1. **Unity Documentation** - [Unity Documentation](https://docs.unity.com)
2. **Firebase Documentation** - [Firebase Documentation](https://firebase.google.com/docs?hl=tr)
3. **.NET Core Documentation** - [.NET Core Documentation](https://learn.microsoft.com/en-us/dotnet/core/)
4. **Python Documentation** - [Python Documentation](https://docs.python.org/3/)

# APPENDIX C: PROJECT MANAGEMENT DOCUMENTS

# APPENDIX C1: PROJECT PLAN

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **COMP 4910/4920 AimMaxHit: Game Performance Analyzer, Project Plan, 20.01.2025, v1.1** | | | | | | | | | | | | | | | | | |
| **Task No** | **Task Name** | **Weeks** | | | | | | | | | | | | | | | **Any Additional Notes** |
| **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** | **13** | **14** | **15** |
| 1 | Responsibility Assignment | X | X | X |  |  |  |  |  |  |  |  |  |  |  |  | |  | | --- | |  |  |  | | --- | | Assigned roles to group members and discussed tools to be used for the project | |
| 2 | Documentation Initiation |  | X | X | X |  |  |  |  |  |  |  |  |  |  |  | |  | | --- | |  |  |  | | --- | | Began project documentation including project structure, game mechanics, and data storage structures | |
| 3 | Research and Planning |  | X | X | X |  |  |  |  |  |  |  |  |  |  |  | Conducted detailed research on Unity libraries and backend integration |
| 4 | RSD Document Preparation |  |  | X | X |  |  |  |  |  |  |  |  |  |  |  | Detailed project requirements and goals were defined |
| 5 | Data Model Design and Testing |  |  |  | X | X |  |  |  |  |  |  |  |  |  |  | Designed and tested the data model for storing user scores |
| 6 | Firebase Setup and Simple Menu Design |  |  |  |  |  | X | X |  |  |  |  |  |  |  |  | Firebase was installed, and a basic user menu was implemented |
| 7 | Gameplay Mechanics Development |  |  |  |  |  |  | X | X | X | X |  |  |  |  |  | Implemented core gameplay elements like targeting, scoring, and timing |
| 8 | Unity Interface Enhancements and Final Demo |  |  |  |  |  |  |  | X | X | X | X | X | X | X |  | Focused on UI improvements and prepared the demo |

# APPENDIX C2: PROJECT EFFORT LOG- CONSOLIDATED

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **COMP 4910 Project Effort Log, AimMaxHit: Game Performance Analyzer, 20.01.2025, v1.0** | | | | | | | | | | |
| **Week** | **Dates** | **Efe Demirtaş** | | **Kaan Mert Kozalı** | | **Umut Özgür İpin** | | **Volkan Ege Kılınç** | | **Total Weekly Effort in Man-Hours** |
| **Work Done** | **Total Hours Spent** | **Work Done** | **Total Hours Spent** | **Work Done** | **Total Hours Spent** | **Work Done** | **Total Hours Spent** |
| **Week 1** | 28.10-3.11 | Initial Unity research and planning | 5 | Collaborated on initial Firebase setup planning | 6 | Assisted in project planning | 5 | Planned backend structure and tools | 6 | 22.00 |
| **Week 2** | 3.11-10.11 | Created Unity project setup, tested UI frameworks | 6 | Planned Firebase structure for scalability | 7 | Assisted in Unity setup and backend structure | 6 | Started Firebase configuration | 7 | 26.00 |
| **Week 3** | 10.11-17.11 | Developed UI skeleton and interaction frameworks | 7 | Designed scalable Firebase data models | 7 | Worked on Unity interaction and backend API integration | 7 | Configured backend scripts for Firebase communication | 7 | 28.00 |
| **Week 4** | 18.11-24.11 | Enhanced front-end UI with game mechanics | 7 | Assisted in designing the Unity gameplay interface | 8 | Assisted in front-end game mechanics | 7 | Configured Firebase for register / login system and user data storage | 7 | 28.00 |
| **Week 5** | 25.11-1.12 | Designed core gameplay mechanics | 8 | |  | | --- | |  |  |  | | --- | | Reviewed Firebase queries for faster data retrieval | | 7 | Collaborated on gameplay and API testing | 7 | Finalized database schema and tested queries | 7 | 30.00 |
| **Week 6** | 2.12-8.12 | Integrated gameplay mechanics with UI | 7 | Improved Firebase scalability performance | 8 | Worked on UI refinements and tested backend connections | 7 | Enhanced data retrieval functionalities | 7 | 28.00 |
| **Week 7** | 9.12-15.12 | Developed new features like score tracking | 8 | Monitored and refined database performance | 8 | Worked on advanced Unity mechanics | 7 | |  | | --- | |  |  |  | | --- | | Added data sync functionalities | | 7 | 30.00 |
| **Week 8** | 16.12-22.12 | Refined gameplay mechanics | 8 | Tested database performance under simulated load | 7 | |  | | --- | | Debugged gameplay interactions |  |  | | --- | |  | | 7 | Enhanced player analytics tracking in Firebase | 7 | 30.00 |
| **Week 9** | 23.12-29.12 | Improved UI for settings and controls | 7 | Assisted enhancing user experience and refining UI elements | 7 | Added interactive elements to UI | 7 | Tested real-time data retrieval | 7 | 28.00 |
| **Week 10** | 30.12-5.1 | Finalized UI elements and bug fixes | 8 | |  | | --- | |  |  |  | | --- | | Reviewed database structure and scalability testing | | 8 | |  | | --- | |  |  |  | | --- | | Final testing and debugging | | 8 | |  | | --- | |  |  |  | | --- | | Debugged and reviewed backend scripts | | 8 | 32.00 |
| **Week 11** | 6.1-12.1 | Polished gameplay for final demo | 8 | Reviewed data flow for demo presentation | 8 | Debugged Unity and Firebase integration | 8 | Prepared Firebase backend for final demo | 8 | 32.00 |
| **Week 12** | 13.1-19.1 | Tested UI and gameplay interactions | 7 | Debugged and optimized Firebase performance | 7 | Finalized Unity design | 7 | Set up an email system in Firebase for support functionality for users. | 7 | 28.00 |
| **Week 13** | 20.1-26.1 | Prepared final presentation and documentation | 6 | Compiled Firebase usage reports and data insights | 6 | Organized final codebase and documentation | 6 | Created login screen in the Unity interface, ensuring full functional connectivity with the application | 6 | 24.00 |
| **Week 14** | 27.1-2.2 |  |  |  |  |  |  |  |  | 0.00 |
| **Total Effort in Man-Hours** |  |  | **186.00** |  | **186.00** |  | **186.00** |  | **186.00** | **744.00** |
| **Total Effort in Man-Days** |  |  | **23.25** |  | **23.25** |  | **23.25** |  | **23.25** | **93.00** |
| **Notes, if any** | | | | | | | | | | |

# APPENDIX C3: PROJECT EFFORT LOGS FOR EACH TEAM MEMBER-

|  |  |  |  |
| --- | --- | --- | --- |
| **COMP 4910/4920 Project Effort Log, AimMaxHit: Game Performance Analyzer, 20.01.2025, v1.0 Team Member: Efe Demirtaş** | | | |
| **Week** | **Dates** | **Work Done in Some Detail** | **Total Hours Spent** |
| **Week 1** | 28.10-3.11 | Initial Unity research and planning | 5 |
| **Week 2** | 3.11-10.11 | Created Unity project setup, tested UI frameworks | 6 |
| **Week 3** | 10.11-17.11 | Developed UI skeleton and interaction frameworks | 7 |
| **Week 4** | 18.11-24.11 | Enhanced front-end UI with game mechanics | 7 |
| **Week 5** | 25.11-1.12 | Designed core gameplay mechanics | 8 |
| **Week 6** | 2.12-8.12 | Integrated gameplay mechanics with UI | 7 |
| **Week 7** | 9.12-15.12 | |  | | --- | |  |  |  | | --- | | Developed new features like score tracking | | 8 |
| **Week 8** | 16.12-22.12 | Refined gameplay mechanics | 8 |
| **Week 9** | 23.12-29.12 | Improved UI for settings and controls | 7 |
| **Week 10** | 30.12-5.1 | Finalized UI elements and bug fixes | 8 |
| **Week 11** | 6.1-12.1 | Polished gameplay for final demo | 8 |
| **Week 12** | 13.1-19.1 | |  | | --- | |  |  |  | | --- | | Tested UI and gameplay interactions | | 7 |
| **Week 13** | 20.1-26.1 | Prepared final presentation and documentation | 6 |
| **Week 14** | 27.1-2.2 |  |  |
| **Total Effort in Man-Hours** |  |  | **186.00** |
| **Total Effort in Man-Days** |  |  | **23.25** |
| **Notes:** | | | |
| **1. This table shows the team member project effort. One Man-Day is Eight Man-Hours. 2. Each team member must fill out the form periodically (preferably at the end of the week of any work done). 3. Each filled-out table must be emailed at the end of each month to a selected Project Member (cc to Project Advisor), who will produce a consolidated table.** | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **COMP 4910/4920 Project Effort Log, AimMaxHit: Game Performance Analyzer, 20.01.2025, v1.0 Team Member: Kaan Mert Kozalı** | | | |
| **Week** | **Dates** | **Work Done in Some Detail** | **Total Hours Spent** |
| **Week 1** | 28.10-3.11 | Collaborated on initial Firebase setup planning | 6 |
| **Week 2** | 3.11-10.11 | Planned Firebase structure for scalability | 7 |
| **Week 3** | 10.11-17.11 | Designed scalable Firebase data models | 7 |
| **Week 4** | 18.11-24.11 | Assisted in designing the Unity gameplay interface | 7 |
| **Week 5** | 25.11-1.12 | Reviewed Firebase queries for faster data retrieval | 8 |
| **Week 6** | 2.12-8.12 | Improved Firebase scalability performance | 7 |
| **Week 7** | 9.12-15.12 | Monitored and refined database performance | 8 |
| **Week 8** | 16.12-22.12 | Tested database performance under simulated load | 8 |
| **Week 9** | 23.12-29.12 | Assisted enhancing user experience and refining UI elements | 7 |
| **Week 10** | 30.12-5.1 | Reviewed database structure and scalability testing | 8 |
| **Week 11** | 6.1-12.1 | Reviewed data flow for demo presentation | 8 |
| **Week 12** | 13.1-19.1 | Debugged and optimized Firebase performance | 7 |
| **Week 13** | 20.1-26.1 | Compiled Firebase usage reports and data insights | 6 |
| **Week 14** | 27.1-2.2 |  |  |
| **Total Effort in Man-Hours** |  |  | **186.00** |
| **Total Effort in Man-Days** |  |  | **23.25** |
| **Notes:** | | | |
| **1. This table shows the team member project effort. One Man-Day is Eight Man-Hours. 2. Each team member must fill out the form periodically (preferably at the end of the week of any work done). 3. Each filled-out table must be emailed at the end of each month to a selected Project Member (cc to Project Advisor), who will produce a consolidated table.** | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **COMP 4910/4920 Project Effort Log, AimMaxHit: Game Performance Analyzer, 20.01.2025, v1.0 Team Member: Umut Özgür İpin** | | | |
| **Week** | **Dates** | **Work Done in Some Detail** | **Total Hours Spent** |
| **Week 1** | 28.10-3.11 | Assisted in project planning | 5 |
| **Week 2** | 3.11-10.11 | Assisted in Unity setup and backend structure | 6 |
| **Week 3** | 10.11-17.11 | Worked on Unity interaction and backend API integration | 7 |
| **Week 4** | 18.11-24.11 | Assisted in front-end game mechanics | 7 |
| **Week 5** | 25.11-1.12 | Collaborated on gameplay and API testing | 7 |
| **Week 6** | 2.12-8.12 | Worked on UI refinements and tested backend connections | 7 |
| **Week 7** | 9.12-15.12 | Worked on advanced Unity mechanics | 7 |
| **Week 8** | 16.12-22.12 | Debugged gameplay interactions | 7 |
| **Week 9** | 23.12-29.12 | Added interactive elements to UI | 7 |
| **Week 10** | 30.12-5.1 | Final testing and debugging | 8 |
| **Week 11** | 6.1-12.1 | |  | | --- | |  |  |  | | --- | | Debugged Unity and Firebase integration | | 8 |
| **Week 12** | 13.1-19.1 | Finalized Unity design | 7 |
| **Week 13** | 20.1-26.1 | Organized final codebase and documentation | 6 |
| **Week 14** | 27.1-2.2 |  |  |
| **Total Effort in Man-Hours** |  |  | **186.00** |
| **Total Effort in Man-Days** |  |  | **23.25** |
| **Notes:** | | | |
| **1. This table shows the team member project effort. One Man-Day is Eight Man-Hours. 2. Each team member must fill out the form periodically (preferably at the end of the week of any work done). 3. Each filled-out table must be emailed at the end of each month to a selected Project Member (cc to Project Advisor), who will produce a consolidated table.** | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **COMP 4910/4920 Project Effort Log, AimMaxHit: Game Performance Analyzer, 20.01.2025, v1.0 Team Member: Volkan Ege Kılınç** | | | |
| **Week** | **Dates** | **Work Done in Some Detail** | **Total Hours Spent** |
| **Week 1** | 28.10-3.11 | Planned backend structure and tools | 6 |
| **Week 2** | 3.11-10.11 | Started Firebase configuration | 7 |
| **Week 3** | 10.11-17.11 | Configured backend scripts for Firebase communication | 7 |
| **Week 4** | 18.11-24.11 | Configured Firebase for register / login system and user data storage | 7 |
| **Week 5** | 25.11-1.12 | Finalized database schema and tested queries | 7 |
| **Week 6** | 2.12-8.12 | Enhanced data retrieval functionalities | 7 |
| **Week 7** | 9.12-15.12 | Added data sync functionalities | 7 |
| **Week 8** | 16.12-22.12 | Enhanced player analytics tracking in Firebase | 7 |
| **Week 9** | 23.12-29.12 | Tested real-time data retrieval | 7 |
| **Week 10** | 30.12-5.1 | Debugged and reviewed backend scripts | 8 |
| **Week 11** | 6.1-12.1 | Prepared Firebase backend for final demo | 8 |
| **Week 12** | 13.1-19.1 | Set up an email system in Firebase for support functionality for users. | 7 |
| **Week 13** | 20.1-26.1 | Created login screen in the Unity interface, ensuring full functional connectivity with the application | 6 |
| **Week 14** | 27.1-2.2 |  |  |
| **Total Effort in Man-Hours** |  |  | **186.00** |
| **Total Effort in Man-Days** |  |  | **23.25** |
| **Notes:** | | | |
| **1. This table shows the team member project effort. One Man-Day is Eight Man-Hours. 2. Each team member must fill out the form periodically (preferably at the end of the week of any work done). 3. Each filled-out table must be emailed at the end of each month to a selected Project Member (cc to Project Advisor), who will produce a consolidated table.** | | | |