CSE 396 – Computer Engineering Project

Gebze Technical University

Spring 2024

Virtual Strike
Documentation of the Project Modules

Prepared By VS Games

(Group 4)

1. Group members

- Ahmet ÖZDEMİR
- Akif Safa ANGİ
- Aykut SERT
- Burak KURT
- Doğukan BAŞ
- İlkay BOLAT
- Muhammet Yasir GÜNEŞ
- Murat ERBİLİCİ

2. Introduction

Game name : Virtual Strike

Project scope: A shooting game where players shoot targets using VR glasses and a special aiming device.

The aim of the project: To offer players an interactive gaming experience with their real-life movements.

The "Virtual Strike" project is an innovative shooting game designed to provide players with an immersive and interactive gaming experience using Virtual Reality (VR) technology. The game integrates players' real-life movements into the virtual world, allowing them to control their in-game character and aim at targets through physical actions.

This document serves as the comprehensive "Documentation of Modules" for the "Virtual Strike" project. It provides detailed information about the various modules that comprise the project, including their purposes, functionalities, design considerations, implementation details, and interdependencies.

The primary goal of this document is to serve as a reference guide for the development team, facilitating a clear understanding of the project's modular structure, technical specifications, and overall architecture. It will also aid in the maintenance and future enhancements of the project by providing a well-documented codebase and design decisions.

The intended audience for this document includes:

1. The development team members actively involved in the implementation and integration of the modules.
2. Instructors and evaluators who will assess the project's technical quality and adherence to software engineering principles.
3. Future developers or researchers who may work on extending or modifying the project.

The document is organized into sections, each dedicated to a specific module or aspect of the project. Each module description includes details about its purpose, design, implementation, dependencies, testing strategies, and integration with other components of the system.
By thoroughly documenting the modules, this document aims to ensure a seamless development process,

effective collaboration among team members, and a solid foundation for future iterations and enhancements of the "Virtual Strike" project.

This project aims to provide the player with real-life sensations in the game to be played through VR glasses. The game is basically based on shooting targets. This game will provide players with a real-time and interactive gaming experience, giving them the feeling of a real shooter as they take aim at targets within the game. "Virtual Strike" will be played using a specialized aiming device integrated with the Unreal Engine game engine and VR (Virtual Reality) headsets. The main goal of the project is to transfer real-life physical activity into the game world while providing players with interactive entertainment. This will make the game both fun and attractive for players who want to spend time in a healthy way. The "Virtual Strike" project represents an innovative approach that combines technology and entertainment to offer players an immersive shooter experience in a virtual reality world. With the successful completion of the project, players will experience a unique gaming experience where they will be able to control their real-world actions in the virtual world. These components are used for this project:

- An Android phone

- A Computer

- A hand-held device as a remote control

- A VR glasses

Firstly, the game is opened on the computer, after the connection with the phone is established, the image of the game is reflected on the phone via the app we will write on Android. The player wearing the glasses with the Android phone will be able to move around in the game and direct the game as he wishes. The control in the player's hand will allow the character in the game to change the direction of movement, walk and interact with obstacles or targets (such as targeting and shooting).

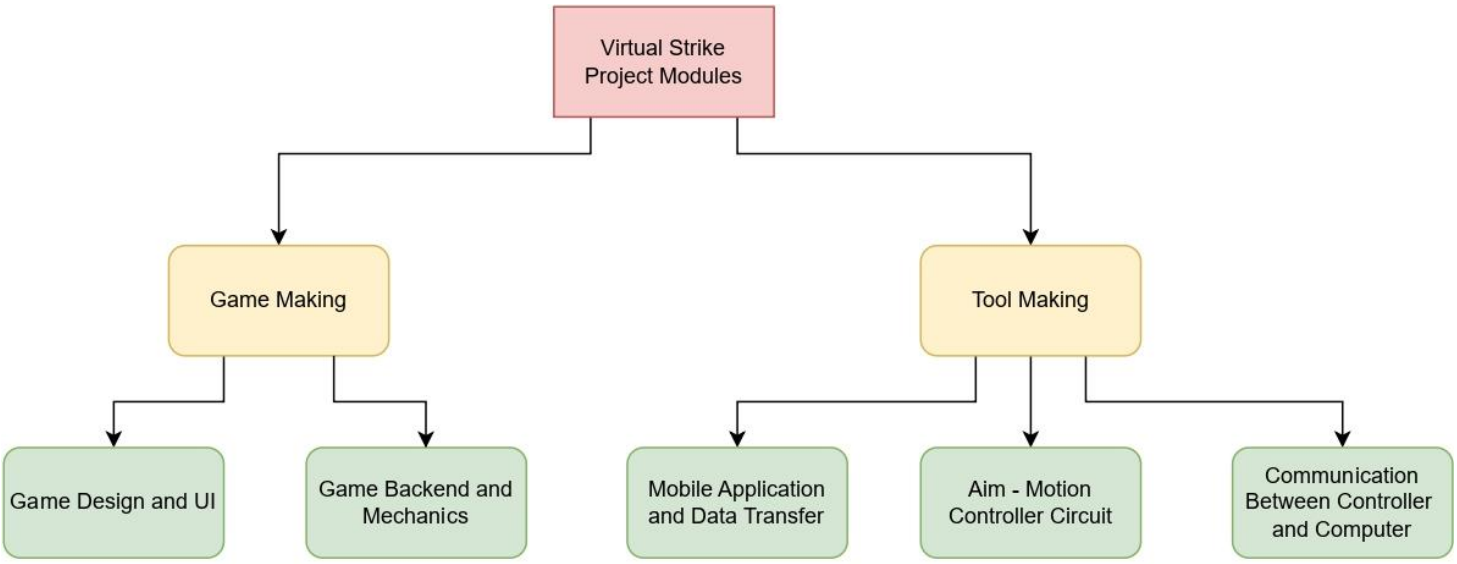
3. MODULES

Modules are separated in order to better manage the project and to better distribute the division of labour among the group members. First of all, we divided the project into two as game making and tool making.

The **game making** part will be divided into two modules, the first module will be **the mechanics behind the game**, in other words the backend part. The second part will be the **design and user interface** of the game.

The construction of the tools which is the **tool making** part of the project will include the remote control in the hand and the reflective app on the phone to which the android device is connected so we will handle these in the modules of **Mobile Application and Data Transfer** module and in the **Aim – Motion Controller Circuit** module. There will also be a module for **Communication Between the Controller and the Game.**

Here is a general overview of the modules:



3.1 - Overview of the Modules:

The "Virtual Strike" project is divided into several modules to facilitate efficient development, collaboration, and maintainability. The modules are designed to encapsulate specific functionalities and responsibilities, ensuring a modular and scalable architecture. The following is an overview of the modules:

1. **Game Backend and Mechanics**
    - This module encompasses the development of the core game mechanics, including player movement, target interactions, scoring systems, and game physics.
2. **Game Design and User Interface**
    - This module focuses on creating visually appealing game elements, intuitive controls, and an immersive user experience through game design and user interface development.
3. **Mobile Application and Data Transfer**
    - This module involves the development of a mobile application that mirrors the computer game's screen to a mobile device and implements a data transfer mechanism to capture the mobile device's motion data and transmit it to the computer.
4. **Aim - Motion Control Circuit Module**
    - This module focuses on developing a motion control circuit that enables users to control their in-game character's direction and aim using hand movements and a button interface using rasperry pi.
5. **Communication Between Controller Hardware and Computer**
    - This module involves creating a communication protocol that facilitates real-time data transmission between the motion control circuit and the computer, enabling in-game controls based on the user's physical movements.

These modules are interconnected and work together to create the complete "Virtual Strike" gaming experience.

The Game Backend and Mechanics module forms the core of the game, handling the game logic and mechanics.

The Game Design and User Interface module ensures an engaging and visually appealing gaming experience.

The Mobile Application and Data Transfer module bridges the gap between the physical world and the virtual environment by capturing the user's movements and transmitting them to the game.

The Aim - Motion Control Circuit module and **Communication Between Controller Hardware and Computer** module collaborate to provide the hardware interface and communication channel for translating the user's real-world movements into in-game actions.

The modular structure of the project allows for parallel development, clear separation of concerns, and easier integration and testing of individual components. Furthermore, it facilitates future expansions and modifications by encapsulating specific functionalities within their respective modules.

3.2 – Module Descriptions:

3.2.1 Game Backend and Mechanics:

This module is responsible for developing the core functionality and mechanics that drive the "Virtual Strike" game. It encompasses the implementation of essential features such as player movement, target interactions, scoring mechanisms, and game physics. The module's primary purpose is to ensure a seamless and engaging gaming experience by translating the user's real-world movements into virtual actions within the game environment.

Key responsibilities of this module include:

1. **Player Movement System**: Accurately capture the player's movements using VR glasses and the handheld device. Translate these real-world movements into in-game character actions, allowing players to navigate the virtual environment seamlessly.

2. **Target Interaction**: Develop mechanisms for spawning and managing targets within the game environment. Implement hit detection algorithms to register successful shots on targets and provide appropriate feedback to the player.

3. **Scoring System**: Design and implement a scoring system that takes into account factors such as accuracy, speed, and precision. Ensure that scores are calculated and updated in real-time during gameplay.

4. **Game Physics**: Integrate realistic physics simulations to enhance the gaming experience. Implement projectile physics for bullets and other in-game objects to create a believable and immersive environment.

This module will be developed using the Unreal Engine game development platform, leveraging its powerful tools and frameworks for game mechanics implementation, VR integration, and physics simulations.

3.2.2 Game Design and User Interface:

This module focuses on creating an engaging and visually appealing gaming experience by designing game elements, environments, and user interfaces. It plays a crucial role in immersing players in the "Virtual Strike" world and providing an intuitive and enjoyable user experience.

Key responsibilities of this module include:

1. **Game Design**: Develop a compelling game concept and storyline that captivates players. Design engaging game levels, environments, and elements that contribute to an immersive and enjoyable experience. Ensure consistency in art style, theme, and narrative throughout the game.

2. **User Interface Design**: Create intuitive and user-friendly interfaces that allow players to navigate the game menus, access settings, and interact with in-game elements seamlessly. The user interface should provide clear feedback and information to enhance the overall gaming experience.

3. **Audio and Visual Assets**: Develop or acquire high-quality audio and visual assets, such as 3D models, textures, sound effects to create a visually stunning and atmospheric gaming environment.

4. **Usability and Playtesting**: Conduct usability tests and playtesting sessions to gather feedback on game design and user interface elements. Iterate and refine designs based on user feedback to improve player engagement and enjoyment.

This module will leverage various tools and software for game design, 3D modeling, texturing, and user interface development, ensuring a cohesive and visually appealing gaming experience.

3.2.3 Mobile Application and Data Transfer:

This module encompasses the development of a mobile application and a data transfer mechanism that facilitates the communication between the mobile device and the computer running the "Virtual Strike" game.

Key responsibilities of this module include:

1. **Mobile App Development**: Develop a mobile application capable of capturing the computer game's screen and mirroring it to the mobile device in real-time. This allows players to experience the game through VR glasses connected to the mobile device.

2. **Data Transfer Mechanism**: Implement a robust data transfer mechanism that captures the mobile device's motion data, such as accelerometer and gyroscope readings. Transmit this motion data to the computer at regular intervals, ensuring that the player's real-time movements are accurately reflected in the game environment.

3. **Performance Optimization**: Optimize the mobile application and data transfer mechanism for efficient performance, minimizing latency and ensuring a smooth and responsive gaming experience.

This module will leverage mobile app development frameworks and technologies, such as Android Studio or Flutter, to create the mobile application. Additionally, it will utilize communication protocols like **WebSockets or TCP/IP** for efficient data transfer between the mobile device and the computer.

3.2.4 Aim - Motion Control Circuit Module:

This module focuses on developing a motion control circuit that enables users to control the direction of their in-game character (aim) based on real-life hand movements. Additionally, users can aim and shoot at target locations using a button interface that interacts with position data from the mobile device. The circuit will consist of a motion sensor or a joystick like component, a button, and a Raspberry Pi microcontroller.

Key responsibilities of this module include:

1. **Hardware Integration**: Integrate a motion sensor (e.g., accelerometer or gyroscope) capable of accurately detecting hand movements or a joystick like component. Incorporate a tactile button that triggers actions such as shooting or initiating specific commands.

2. **Microcontroller Programming**: Develop software for the Raspberry Pi microcontroller to process input data from the motion sensor and button, and generate corresponding output signals for in-game actions.

3. **Communication Protocol**: Implement a wireless communication protocol (e.g., Bluetooth or Wi-Fi) to enable real-time data exchange between the Raspberry Pi and the computer running the game.

4. **Calibration and Testing**: Conduct extensive testing and calibration to ensure optimal performance of the motion sensor and button interface across different hand gestures and button inputs. Verify compatibility and stability with the mobile device and computer.

This module will involve hardware prototyping, embedded systems programming, and the integration of wireless communication protocols to create a robust and reliable motion control circuit for the "Virtual Strike" game.

3.2.5 Communication Between Controller Hardware and Computer:

This module is responsible for establishing a communication protocol that facilitates real-time data transmission between the motion control circuit (developed in the previous module) and the computer running the "Virtual Strike" game. The controller hardware will transmit motion and button information to the computer, enabling in-game controls based on the user's physical movements.

Key responsibilities of this module include:

1. **Communication Protocol Design**: Define and implement a communication protocol that supports reliable, low-latency, and real-time data transmission between the controller hardware and the computer.

2. **Data Transmission Mechanism**: Develop a data transmission mechanism that accurately transfers motion and button information from the controller hardware to the computer, ensuring minimal data loss and latency.

3. **Hardware and Software Integration**: Integrate hardware interfaces compatible with the chosen communication protocol on both the controller hardware and the computer side. Develop or integrate software components on the computer to receive and process the incoming data.

4. **Testing and Debugging**: Create comprehensive test plans to verify the accuracy and reliability of the communication protocol and data transmission mechanism. Identify and address any errors or issues encountered during the testing process.

This module will leverage various technologies and tools, such as hardware interfaces (e.g., serial ports, Bluetooth modules), communication protocols (e.g., TCP/IP, WebSockets), and software development frameworks, to ensure seamless communication between the controller hardware and the computer running the "Virtual Strike" game.

The collaboration and successful integration of all these modules will result in a fully functional and immersive "Virtual Strike" gaming experience, where players can control and interact with the virtual environment using their real-world movements and the motion control circuit.

4. Design and Architecture

The "Virtual Strike" project follows a modular design approach, where each module encapsulates specific functionalities and responsibilities. This modular architecture promotes separation of concerns, code reusability, and maintainability. The design and architecture of each module are outlined below:

4.1 Game Backend and Mechanics

- **Architecture:** This module follows a layered architecture pattern, with distinct layers for game logic, physics simulations, input handling, and data management.

- **Design Patterns:** The module incorporates design patterns such as the Observer pattern for event handling, the State pattern for managing game states, and the Singleton pattern for managing global game resources.

- **Core Components:**
    - *Game Logic Component:* Responsible for managing game rules, player interactions, scoring, and progression.
    - *Physics Engine Component:* Handles realistic physics simulations for projectiles, collisions, and rigid body dynamics.
    - *Input Management Component:* Captures and processes input from the motion control circuit and translates it into in-game actions.
    - *Data Management Component:* Handles game data storage, retrieval, and persistence.

4.2 Game Design and User Interface

- **Architecture:** This module follows a component-based architecture, where game elements, UI components, and asset management are separated into distinct components.

- **Design Patterns:** The module employs the Model-View-Controller (MVC) pattern to separate the game logic from the presentation layer, and the Decorator pattern for adding visual elements and behaviors to game objects.

- **Core Components:**
    - *Game Element Component*: Manages the creation, rendering, and behavior of game elements such as levels, environments, and interactive objects.
    - *User Interface Component:* Handles the rendering and interaction with menus, heads-up displays (HUDs), and other UI elements.
    - *Asset Management Component:* Responsible for loading, caching, and managing game assets such as 3D models, textures, sounds, and animations.

4.3 Mobile Application and Data Transfer

- **Architecture:** This module follows a client-server architecture, where the mobile application acts as the client, and the computer running the game acts as the server.

- **Design Patterns:** The module utilizes the Observer pattern for handling asynchronous data transfer events and the Singleton pattern for managing the communication channel between the mobile device and the computer.

- **Core Components:**
    - *Mobile Application Component:* Responsible for capturing the game screen and rendering it on the

mobile device, as well as capturing motion data from the device's sensors.
- *Data Transfer Component:* Handles the transmission and reception of data between the mobile device and the computer, ensuring reliable and efficient data transfer.
- *Communication Channel Component:* Manages the establishment and maintenance of the communication channel between the mobile device and the computer.

4.4 Guidance - Motion Control Circuit Module

- **Architecture:** This module follows an embedded systems architecture, with a microcontroller (Raspberry Pi) acting as the central processing unit.

- **Design Principles:** The module adheres to principles of low-level hardware programming, real-time systems design, and efficient data processing.

- **Core Components:**
  - *Motion Sensor Component:* Responsible for capturing and processing motion data from the hardware sensor.
  - *Button Input Component:* Handles the input from the tactile button and triggers corresponding actions.
  - *Microcontroller Firmware:* The software running on the Raspberry Pi microcontroller, responsible for processing sensor data, button input, and communicating with the mobile device or computer.

4.5 Communication Between Controller Hardware and Computer

- **Architecture:** This module follows a client-server architecture, where the controller hardware acts as the client, and the computer running the game acts as the server.

- **Design Patterns:** The module employs the Observer pattern for handling asynchronous data reception and the Adapter pattern for translating data between different formats or protocols.

- **Core Components:**
  - *Communication Protocol Component:* Responsible for implementing the chosen communication protocol and enabling reliable data transmission between the controller hardware and the computer.
  - *Data Translation Component:* Handles the translation of data received from the controller hardware into a format compatible with the game engine or the computer's operating system.
  - *Data Reception Component:* Manages the reception and processing of data from the controller hardware, ensuring efficient handling of incoming data streams.

The modular design and architecture of the "Virtual Strike" project promote code organization, separation of concerns, and scalability. Each module's design and architecture are tailored to address the specific requirements and constraints of its responsibilities, while adhering to best practices in software engineering and game development.

5. Implementation Details

5.1 Game Backend and Mechanics

- **Technologies and Tools**:

    - Unreal Engine: The core game mechanics and backend systems will be developed using Unreal Engine, leveraging its powerful game development framework and C++ programming capabilities.
    - VR Development Kit: Integration with a VR development kit compatible with Unreal Engine will be implemented to enable VR functionality.
    - Version Control System: A version control system like Git and Github will be utilized to manage collaborative development, track changes, and facilitate code merging and integration.
    - Integrated Development Environment (IDE): Visual Studio or another compatible IDE will be used for C++ development and blueprint scripting within the Unreal Engine environment.

- **Key Implementation Details**:

    - **Player Movement System**: This system will capture input data from the motion control circuit and translate it into character movement within the game world.

    - **Target Interaction**: Algorithms for target spawning, hit detection, and collision handling will be implemented. Techniques like raycast or sphere-cast may be used for accurate target detection and collision resolution.

    - **Scoring System**: The scoring system will be implemented based on factors like accuracy, speed, and precision. Real-time score calculation and updates will be handled through event-driven programming and user interface integration.

    - **Game Physics**: Integration with Unreal Engine's built-in physics engine (PhysX or Chaos) will be leveraged to simulate realistic projectile physics, collisions, and rigid body dynamics. Physics materials, constraints, and forces will be applied to achieve desired behaviors.

5.2 Game Design and User Interface

- **Technologies and Tools**:

    - Unreal Engine game development platform and Unreal Engine built-in or community assets.
    - 3D Modeling Software (e.g., Maya, Blender, 3ds Max): Professional 3D modeling tools will be used for creating game assets such as characters, environments, and props or built-in and community made assets might be used.
    - Texture Painting and Image Editing Software (e.g., Substance Painter, Photoshop): Tools for creating and editing textures, materials, and user interface elements or built-in and community made assets might be used.
    - Audio Editing Software (e.g., Audacity, FMOD Studio): Software for recording, editing, and processing sound effects and background music or built-in and community made assets might be used.

- **Key Implementation Details**:

    - **Game Design**: The game concept, storyline, and level designs will be documented and iteratively refined based on feedback from playtesting sessions. Game design documents and level blueprints will guide the implementation process.

- **User Interface Design**: User interface mockups and prototypes will be created using design tools, and the final UI will be implemented within the Unreal Engine environment, leveraging its UI rendering and input handling capabilities.
- **Asset Integration**: Game assets such as 3D models, textures, animations, and audio files will be imported into the Unreal Engine project and properly configured for use within the game.
- **Shaders and Post-Processing Effects**: Specialized shaders and post-processing effects will be implemented to enhance the visual quality and atmosphere of the game, such as lighting, particle effects, and environmental effects.

5.3 Mobile Application and Data Transfer

- **Technologies and Tools**:
  - Mobile App Development Framework (e.g., Android Studio with Java or Kotlin, Flutter): A framework for developing the mobile application that captures the game screen and transmits motion data.
  - Communication Protocol (e.g., WebSockets, TCP/IP): A protocol for establishing and maintaining real-time communication between the mobile device and the computer running the game.
  - Game Streaming Libraries (e.g., Moonlight, Steam Link): Libraries or tools that facilitate efficient game streaming from the computer to the mobile device.

- **Key Implementation Details**:
  - **Mobile App Development**: The mobile application will be developed using a chosen framework like Android Studio or Flutter, leveraging platform-specific APIs for capturing screen content and accessing device sensors (accelerometer, gyroscope).
  - **Game Streaming**: Techniques for efficient game streaming from the computer to the mobile device will be implemented, ensuring low latency and high visual quality. Game streaming libraries or custom streaming protocols may be utilized.
  - **Data Transfer Mechanism**: A communication protocol like WebSockets or TCP/IP will be implemented to enable real-time transmission of motion data from the mobile device to the computer running the game. Data serialization and deserialization techniques will be employed for efficient data transfer.
  - **Performance Optimization**: Techniques for optimizing the mobile application's performance, such as multithreading, caching, and efficient resource management, will be implemented to ensure a smooth and responsive gaming experience.

5.4 Aim - Motion Control Circuit Module

- **Technologies and Tools**:
  - Motion Sensor (e.g., Accelerometer, Gyroscope) – or Joystick like component: A hardware sensor capable of detecting motion and orientation changes will be integrated into the circuit.
  - Microcontroller (Raspberry Pi): A Raspberry Pi or similar microcontroller will be used as the central processing unit for the motion control circuit.
  - Embedded Programming Language (C++): C++ for embedded systems development on the chosen microcontroller platform.
  - Wireless Communication Protocol (e.g., Bluetooth, Wi-Fi): A wireless communication protocol will be implemented for data transmission between the motion control circuit and the mobile device or computer.

- **Key Implementation Details**:
  - **Motion Sensor Integration**: The chosen motion sensor (accelerometer, gyroscope, or a combination) will be integrated with the microcontroller, and the necessary drivers and libraries will be installed for reading sensor data.
  - **Button Input Handling**: The tactile button input will be connected to the microcontroller, and the necessary code will be implemented to detect button presses and triggers corresponding actions.
  - **Microcontroller Firmware**: The firmware running on the microcontroller will be developed using C++. It will handle sensor data processing, button input handling, and communication with the mobile device or computer.

- **Wireless Communication**: A wireless communication protocol like Bluetooth or Wi-Fi will be implemented to enable real-time data transmission between the motion control circuit and the mobile device or computer running the game.

5.5 Communication Between Controller Hardware and Computer

- **Technologies and Tools**:
  - Communication Protocol (e.g., TCP/IP, WebSockets, Serial Communication): A suitable communication protocol will be chosen and implemented for enabling data transmission between the controller hardware and the computer.
  - Hardware Interfaces (e.g., Serial Ports, Bluetooth Modules): Hardware interfaces compatible with the chosen communication protocol will be integrated into both the controller hardware and the computer.
  - Data Parsing and Serialization Libraries: Libraries or frameworks for parsing and serializing data in the required formats will be utilized for efficient data transmission and reception.

- **Key Implementation Details**:
  - **Communication Protocol Implementation**: The chosen communication protocol (e.g., TCP/IP, WebSockets, or serial communication) will be implemented on both the controller hardware and the computer side, ensuring reliable and efficient data transmission.
  - **Data Parsing and Serialization**: Techniques for parsing and serializing data in the required formats will be implemented, ensuring compatibility between the controller hardware and the computer. Libraries or frameworks like Google Protobuf or JSON may be utilized for data serialization and deserialization.
  - **Hardware Interface Integration**: The necessary hardware interfaces, such as serial ports or Bluetooth modules, will be integrated into both the controller hardware and the computer to enable physical communication between the two systems.
  - **Data Reception and Processing**: On the computer side, a component will be developed to receive and process the incoming data from the controller hardware. This component will translate the received data into a format compatible with the game engine or the computer's operating system, enabling seamless integration with the game.

The implementation details outlined above provide a comprehensive overview of the technologies, tools, and techniques that will be employed in the development of each module. These details will serve as a guide for the development team, ensuring a consistent and coherent implementation approach across all modules while adhering to industry standards and best practices.

6. Dependencies and Integrations

6.1 Game Backend and Mechanics

- **External Dependencies**:
    - Unreal Engine: The Game Backend and Mechanics module heavily relies on the Unreal Engine game development framework, which provides the core functionality for rendering, physics simulation, input handling, and more.
    - VR Development Kit: Integration with a VR development kit whic is built-in in Unreal Engine is necessary to enable virtual reality functionality within the game.

- **Integration with Other Modules**:

    - **Integration with Game Design and User Interface Module**: The Game Backend and Mechanics module will integrate with the Game Design and User Interface module to ensure that the game mechanics and backend systems are seamlessly integrated with the visual and user interface elements.
    - **Integration with Mobile Application and Data Transfer Module**: The motion data captured by the Mobile Application and Data Transfer module will be transmitted to the Game Backend and Mechanics module for processing and translating into in-game actions.
    - **Integration with Communication Between Controller Hardware and Computer Module**: The data received from the controller hardware through the Communication Between Controller Hardware and Computer module will be processed and integrated into the Game Backend and Mechanics module to enable user input and control.

6.2 Game Design and User Interface

- **External Dependencies**:
    - Unreal Engine game development platform and Unreal Engine built-in or community assets.
    - 3D Modeling Software (e.g., Maya, Blender, 3ds Max): These external tools are necessary for creating and importing 3D models, animations, and other game assets into the Unreal Engine project.
    - Texture Painting and Image Editing Software (e.g., Substance Painter, Photoshop): External software for creating and editing textures, materials, and user interface elements.
    - Audio Editing Software (e.g., Audacity, FMOD Studio): External tools for recording, editing, and processing sound effects and background music.

- **Integration with Other Modules**:
    - **Integration with Game Backend and Mechanics Module**: The Game Design and User Interface module will integrate with the Game Backend and Mechanics module to ensure that the visual and user interface elements are correctly synchronized with the game mechanics and backend systems.

6.3 Mobile Application and Data Transfer

- **External Dependencies**:
    - Mobile App Development Framework (e.g., Android Studio, Flutter): External frameworks and tools for developing the mobile application.
    - Game Streaming Libraries (e.g., Moonlight, Steam Link): External libraries or tools for efficient game streaming from the computer to the mobile device.

- **Integration with Other Modules**:
    - **Integration with Game Backend and Mechanics Module**: The motion data captured by the Mobile Application and Data Transfer module will be transmitted to the Game Backend and Mechanics module for processing and translating into in-game actions.
    - **Integration with Communication Between Controller Hardware and Computer Module**: The Mobile Application and Data Transfer module may integrate with the Communication Between Controller Hardware and Computer module to receive data from the controller hardware and transmit it to the computer running the game.

6.4 Aim - Motion Control Circuit Module

- **External Dependencies**:
    - Motion Sensor Hardware (e.g., Accelerometer, Gyroscope): External hardware components, such as an accelerometer or gyroscope, will be integrated into the motion control circuit.
    - Microcontroller Development Environment (e.g., Raspberry Pi OS). External development environments and tools for programming the microcontroller.

- **Integration with Other Modules**:
    - **Integration with Communication Between Controller Hardware and Computer Module**: The Guidance - Motion Control Circuit module will integrate with the Communication Between Controller Hardware and Computer module to transmit motion and button data to the computer running the game.

6.5 Communication Between Controller Hardware and Computer

- **External Dependencies**:
    - Communication Protocol Libraries (e.g., WebSocket libraries, Serial Communication libraries): External libraries or frameworks for implementing the chosen communication protocol.
    - Data Parsing and Serialization Libraries (e.g., Google Protobuf, JSON libraries): External libraries for parsing and serializing data in the required formats.

- **Integration with Other Modules**:
    - **Integration with Game Backend and Mechanics Module**: The Communication Between Controller Hardware and Computer module will integrate with the Game Backend and Mechanics module to transmit the data received from the controller hardware, enabling user input and control within the game.
    - **Integration with Guidance - Motion Control Circuit Module**: The Communication Between Controller Hardware and Computer module will integrate with the Guidance - Motion Control Circuit module to receive motion and button data from the controller hardware.
    - **Integration with Mobile Application and Data Transfer Module (optional)**: Depending on the architecture and design decisions, the Communication Between Controller Hardware and Computer module may integrate with the Mobile Application and Data Transfer module to receive data from the mobile device and transmit it to the computer running the game.

The dependencies and integrations outlined above highlight the various external dependencies each module relies on and how the modules interact and integrate with each other to form a cohesive and functional system. Proper management of these dependencies and seamless integration between modules are crucial for the successful implementation and operation of the "Virtual Strike" project.

7. Testing and Validation

Testing and validation are crucial steps in ensuring the reliability and overall quality of the "Virtual Strike" project. Various testing strategies will be employed to validate the functionality and performance of each module.

7.1 Game Backend and Mechanics

- **Integration Testing**: Integration tests will be conducted to validate the seamless integration of the various components within the Game Backend and Mechanics module, as well as its integration with other modules, such as the Game Design and User Interface module.
- **Performance Testing**: Performance tests will be carried out to ensure optimal performance, particularly for computationally intensive tasks like physics simulations and collision detection algorithms.
- **Stress Testing**: Stress tests will be performed to evaluate the module's behavior under extreme conditions, such as high load scenarios or edge cases.

7.2 Game Design and User Interface

- **Visual Testing**: Manual testing will be conducted to validate the visual appearance and behavior of game elements, user interfaces, and assets within the Unreal Engine environment.
- **Usability Testing**: Usability tests will be performed with a sample group of users to gather feedback on the user experience, intuitiveness of the user interface, and overall game design.
- **Asset Validation**: Automated tests or scripts will be developed to validate the integrity and correct importing of game assets, such as 3D models, textures, and audio files.

7.3 Mobile Application and Data Transfer

- **Integration Testing**: Integration tests will be conducted to validate the communication between the mobile application and the game running on the computer, ensuring reliable data transfer and synchronization.
- **Compatibility Testing**: Compatibility tests will be performed to ensure the mobile application and data transfer mechanisms work correctly across different mobile devices, operating systems, and hardware configurations.
- **Performance Testing**: Performance tests will be carried out to evaluate the mobile application's responsiveness, frame rate, and data transfer latency, ensuring a smooth gaming experience.

7.4 Aim - Motion Control Circuit Module

- **Hardware Testing**: Extensive hardware testing will be conducted to validate the accurate detection and processing of motion sensor data and button input under various conditions and scenarios.
- **Communication Testing**: Tests will be performed to ensure reliable and efficient communication between the motion control circuit and the mobile device or computer using the chosen wireless communication protocol.

7.5 Communication Between Controller Hardware and Computer

- **Integration Testing**: Integration tests will be conducted to validate the seamless communication and data exchange between the controller hardware and the computer running the game.
- **Load Testing**: Load tests will be performed to evaluate the communication protocol's performance and reliability under high data transfer rates and concurrency scenarios.
- **Compatibility Testing**: Compatibility tests will be carried out to ensure the communication protocol and data transfer mechanisms work correctly across different hardware configurations.

In addition to the module-specific testing strategies, the following general testing practices will be employed:

- **Code Reviews**: Regular code reviews will be conducted to ensure adherence to coding standards, best practices, and to identify potential issues or areas for improvement.
- **Version Control and Continuous Integration**: A version control system like Git will be used for source code management, and a continuous integration (CI) pipeline will be established to automate the build, testing, and deployment processes.
- **Bug Tracking and Issue Management**: A bug tracking and issue management system will be utilized to report, prioritize, and track bugs and issues encountered during the development and testing phases.

By employing these testing and validation strategies, the development team can ensure the quality, reliability, and overall robustness of the "Virtual Strike" project, minimizing the risk of critical issues or bugs in the final product.

8. Modules and their Distribution among Developers:

Virtual Strike	Game Backend and Mechanics	Game Design and User Interface	Mobile Application and Data Transfer	Aim - Motion Control Circuit Module	Communication Between Controller Hardware and Computer
Ahmet Özdemir			+	+	+
Akif Safa Angi	+	+			+
Aykut Sert	+	+			+
Burak Kurt	+	+			+
Doğukan Baş		+	+	+	+
İlkay Bolat	+		+	+	
M. Yasir Güneş			+	+	+

Murat Erbilici	+	+	+		