# Next-Step: Requirements

---


Jett Koele
Benjamin Huntoon
Naima Ontiveros
Kendall Callison

Sponsored by:
Jack R Williams & Zachary Lerner

Mentored by:
Scott Larocca

Version: 3.0

Client Signature:
Jack R. Williams

Team Signature:
Jett Koele

# Table of Contents

## 1. Introduction

In the United States, over 7.5 million individuals face difficulty walking and engaging in physical activity due to conditions that impair mobility. For these individuals, traditional treatment approaches such as physical therapy (PT) and passive leg braces are commonly prescribed. However, despite PT's potential benefits, maximizing the amount and effectiveness of therapy can be challenging.

The Biomechatronics Lab at Northern Arizona University (NAU), led by Dr. Zachary Lerner, is working to address these challenges by developing a fully open-source exoskeleton system known as OpenExo. The current system is designed to help overcome barriers to the widespread use of wearable exoskeletons by making them more accessible. OpenExo, combined with an open-source Python API, enables real-time operation and assessment of the exoskeleton. Still, a key challenge remains: how to keep users engaged during rehabilitation to ensure they get the maximum benefit from using the device.

Our capstone project focuses on addressing this engagement challenge. We aim to develop a gamified rehabilitation training tool that interfaces with the OpenExo system through its already-built Python API. This component we will add to the Python API will communicate with Bluetooth sensors and built-in sensors to collect real-time data and provide interactive, game-like feedback to patients as they use the exoskeleton for different exercises. By transforming rehabilitation into an engaging experience, we hope to improve patient motivation and enhance the overall effectiveness of therapy. To make the tool more accessible and encourage collaboration, we are making it open-source. This will let developers, therapists, and researchers customize and improve it to fit different rehabilitation needs. With an open-source approach, we hope to build a community that keeps the tool innovative, flexible, and available to everyone.

We are building on an existing program that was developed by the Biomechatronics Lab at NAU to enhance its functionality for rehabilitation robotics applications. This includes expanding the core modules, such as chart_data.py, exoData.py, exoDeviceManager.py, exoTrial.py, openPythonApi.py, and realTimeProcessor.py. Additionally, we are integrating new functionalities within the provided GUI.py and BioFeedback.py files to create a more user-friendly interface and improve real-time biofeedback capabilities. Our goal is to create an interactive tool that can support both research and practical applications in rehabilitation therapy.

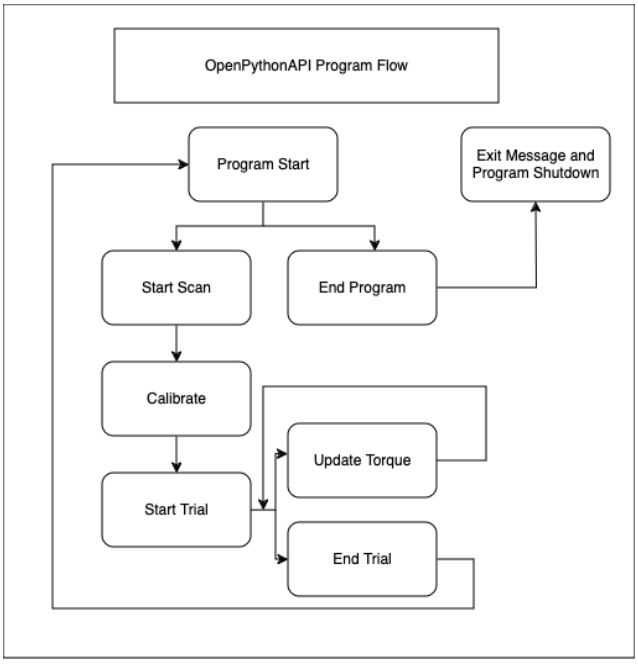
This requirements document outlines the essential needs and expectations for developing our gamified rehabilitation tool compatible with the exoskeleton system. It specifies core features like real-time data collection via Bluetooth, providing researchers with valuable data and offering interactive feedback to keep patients engaged. Key requirements include user engagement tools, such as rewards and progress tracking, as well as detailed functional requirements that define the system’s core functions and features. Additionally, performance

requirements outline expected responsiveness, speed, accuracy, and usability, ensuring that the tool provides helpful feedback. The document also addresses environmental requirements, covering hardware and software compatibility, along with potential risks that could impact the system's success. Guidelines are provided to ensure an intuitive, easy-to-use interface accessible to both therapists and patients, regardless of their technical skills. To make the tool adaptable and accessible to a wider audience, we have established open-source requirements, allowing developers, therapists, and researchers to customize the tool to meet various rehabilitation needs.

## 2. Problem Statement

The Biomechatronics Lab at Northern Arizona University (NAU) has developed the OpenExo exoskeleton system, designed to aid rehabilitation for individuals with mobility impairments. The system is controlled via the OpenPythonAPI, an open-source Python-based application that connects to the exoskeleton using Bluetooth Low Energy (BLE). This API provides a means for therapists and researchers to collect movement data during therapy sessions, making OpenExo an accessible and customizable tool for rehabilitation. However, the current system lacks interactive features and gamified elements that could significantly improve patient engagement and therapy effectiveness.

**2.1 OpenPythonAPI Program Flow**: This diagram displays the current workflow of the OpenPythonAPI, showing the sequence from program initiation to Bluetooth scanning, device calibration, and trial operation. Key actions during a trial include updating torque or ending the trial, which generates a CSV output file. The workflow highlights the need for patient feedback and engagement features.



### 2.2 Current Workflow of the Client's System

In the current setup, the OpenExo system is operated through the command line, where users start the program by initializing openPythonApi.py with keyboard input. The program scans for Bluetooth devices matching the exoskeleton's Universally Unique Identifier (UUID) and then calibrates the device before starting a therapy trial. During a trial, the system runs in a loop that allows users to adjust torque settings or end the session. When a session concludes, the data is saved to a CSV file containing all collected exoskeleton data, such as movement and torque information, which can be analyzed later by therapists. The current GUI.py file provides only basic interaction capabilities, offering limited functionality for enhancing patient involvement.

### 2.3 Limitations in the Existing Workflow

**Limited User Interface Functionality:** The current GUI.py and command-line interface provide only basic interaction, lacking features that could make therapy sessions more engaging and interactive for patients.

**Lack of Real-Time Feedback for Patients:** Although the system collects data and saves it to CSV files, this data is not reorganized or displayed in a way that offers immediate feedback or progress updates to the patient, which could help increase motivation and understanding of their performance.

**No Gamification or Motivational Elements**: The current workflow does not include gamified elements (such as rewards, levels, or achievements) that are known to improve user engagement and encourage patients to reach therapy goals.

## 3. Solution Vision

To address the engagement and feedback challenges identified in the current OpenExo system, we are developing an interactive, gamified rehabilitation tool that integrates with the existing OpenPythonAPI. This tool will provide real-time feedback and motivating game elements to enhance the patient experience, making therapy sessions more engaging and effective. By creating a more enjoyable interface, our solution will encourage patients to remain motivated during rehabilitation exercises, which can improve therapy outcomes.

### 3.1 Key Features of Our Solution

**External Interactive Interface**: We will develop an external window that connects to the existing GUI.py, enhancing it with new features that allow for real-time feedback and interaction with the patient.

**Unity-Based Game Integration**: Our initial game, "Grape Smasher," will be designed to respond to patient movements, where applying pressure while stepping allows them to "crush" grapes and make juice. This provides immediate, game-like feedback based on real-time data from the exoskeleton.

**User-Friendly Game Expansion**: We plan to add additional games that are simple and enjoyable for users, ensuring they remain engaged while performing therapeutic exercises. Each game will be designed to make the exercises feel accessible and fun.

**Reorganized Data for Progress Tracking**: The system will transform data collected during exercises into user-friendly statistics and interactive progress indicators, enabling patients to view their achievements and for researchers to track their therapy progress in an efficient format.

### 3.2 Data Collection, Processing, and Usage

**Data Input**: The system collects movement data from Bluetooth-enabled sensors within the exoskeleton, tracking metrics such as pressure and range of motion. By using the Bleak library for Bluetooth communication, we ensure a reliable connection between the exoskeleton and the system.

**Real-Time Data Translation and Feedback**: Data from the sensors is processed in real-time, translating user movements into game actions with the help of vGamepad for virtual controller integration. This setup allows each exercise action (e.g., applying pressure) to have an immediate, visible impact on the game, enhancing the patient's understanding of their progress.

**Data Output and Long-Term Feedback**: Beyond immediate feedback, the data is reorganized into interactive statistics that patients can view to understand their long-term progress. The game interface will show metrics and achievements, helping patients track their improvements and stay motivated over time.

**Therapist Data Access**: For therapists and researchers, CSV files will continue to be generated to preserve detailed session data for professional analysis. This ensures that while patients engage with a simplified visual format, therapists still have access to in-depth data for evaluation and customization of therapy plans.

### 3.3 System Architecture and Tool Integration

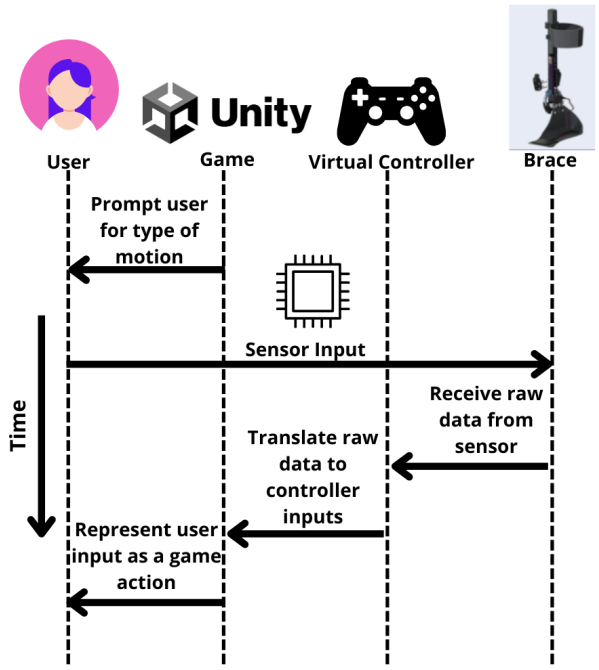
To achieve this game integration, we are using Python as the primary programming language, with Bleak handling Bluetooth communication and vGamepad supporting virtual controller inputs. Unity, alongside Pygame for certain interactive components, powers the game visuals, creating an immersive experience for users. This architecture supports real-time data

flow from the exoskeleton sensors to the game interface, providing accurate and responsive feedback while preserving the underlying data collection framework for therapist access.

### 3.4 Impact on the Client's Workflow

Our gamified tool will transform the system from a simple data collection tool into an active rehabilitation aid that greatly improves the patient experience and motivation. Therapists will still have access to CSV files for detailed analysis, but patients will now receive engaging, real-time feedback and a better understanding of their progress, helping them stay committed to their therapy. This approach enhances therapy outcomes without adding extra complexity to the client's workflow. The client will also be able to build upon the tool since it will be open source.

We considered other options, such as improving only the existing GUI.py or using simpler game mechanics, but chose Unity for its flexibility and robust visual capabilities. Integrating an external interface with GUI.py allows us to deliver a highly interactive experience while preserving the core functionality of the existing API, making the solution adaptable and scalable. By making the OpenExo system more engaging and accessible, we hope to enhance rehabilitation outcomes for patients with mobility challenges. This open-source solution will also encourage collaboration among developers, therapists, and researchers, enabling the creation of additional games or adaptations for varied rehabilitation needs. This approach allows for more innovation in rehabilitation technology, advancing patient-centered care.



**3.5 User Interaction Workflow for Gamified Rehabilitation:** This diagram shows the flow of data from the user's motion to the Unity game interface, via sensor input from the exoskeleton. As the user performs a motion, the sensor captures raw data, which is then translated into game controller inputs that represent the action in the game. This loop provides real-time feedback, enhancing user engagement during rehabilitation exercises.

## 4. Project Requirements

Expanding on the problem statement, which highlighted the limitations of traditional rehabilitation methods such as low engagement, lack of real-time feedback, and limited customization, this section outlines the requirements to address these issues. A gamified rehabilitation system provides a dynamic solution by offering interactive exercises that keep users engaged while equipping researchers with tools to customize and monitor sessions effectively.

These requirements ensure the system is functional, efficient, and adaptable to meet the needs of users and researchers. By focusing on input controls, feedback systems, and environmental compatibility, the system will deliver an intuitive, engaging, and effective rehabilitation experience. Addressing performance and hardware limitations further ensures smooth, reliable operation, laying a strong foundation for meaningful user progress.

### 4.1 Functional Requirements

**4.1.1 User Input & Control:** With regards to input and control requirements, we are to ensure that input is collected from two sensors built into the exoskeleton and convert that biofeedback into game input through a virtual controller so that it can be used to control Unity and Pygame built games. We also want the game movements and controls to focus on one area of the body (e.g. legs, arms depending on the game being played and the exercise conducted) to avoid overwhelming and confusing the user. Finally, we are going to enable game customization for researchers by enabling editable code sections, this would effectively allow researchers to modify the user experience before starting the game, selecting the game time and amount of exercise required for game completion.

**Key Requirements:**

**4.1.1.1 Collect Data from Exoskeleton Sensors:**

The system will rely on two sensors integrated into the exoskeleton to collect biofeedback data. These sensors will provide real-time input that reflects the user’s movements. This data will be processed by a virtual controller, converting it into

game-compatible commands. This ensures accurate and responsive control of games built using Unity or Pygame, allowing the system to deliver a seamless and engaging user experience.

#### 4.1.1.2 Focus Game Movements on a Specific Part of the Body:

Game mechanics will be designed to target one specific area of the body, such as the legs or arms, depending on the requirements of the game and the rehabilitation exercise. By limiting focus to a single body part, the system minimizes cognitive overload and ensures that users can comfortably interact with the game. This approach makes the rehabilitation process intuitive and accessible, particularly for users who may be new to video games or physical therapy.

#### 4.1.1.3 Enable Researcher Customization:

The system will provide researchers with tools to customize the user experience. Editable code sections will allow researchers to modify key aspects of the game, such as:

- Setting the game duration to align with session goals.
- Defining the amount of exercise required for game completion.
  This flexibility ensures that the rehabilitation process can be tailored to meet the specific needs of each user, making the system adaptable for a wide range of therapeutic scenarios.

**4.1.2 Feedback & Statistics:** Before the session begins researchers should have the ability to set certain game metrics to adjust difficulty, each level of difficulty should vary for each game implemented into the final system. During the gamified rehabilitation session, we are to provide the user with real-time audio and visual stimulation to indicate session progress as the user approaches the predetermined success threshold. The researcher should be able to change the difficulty level of the game during the game session by pausing the game and adjusting the settings accordingly. Some indicators of session progress should be visible on screen in a manner that is immediately recognizable to the user (e.g. a progress bar, or a change in screen color). For session statistics, we are to collect key metrics like success rate and sensor thresholds for the eventual uploading to an external CSV file.

**Key Requirements:**

#### 4.1.2.1 Calibrate the Exoskeleton Before the Game:

The exoskeleton must be calibrated before each session to ensure accurate data integration, though hardware thresholds are handled by the device itself.

#### 4.1.2.2 Adjust Game Difficulty Based on Researcher Metrics:

Researchers can set difficulty levels tailored to each game before the session begins, ensuring appropriate challenges for users.

#### 4.1.2.3 Real-Time Feedback and Visual Indicators:

Users receive real-time audio and visual feedback, such as progress bars or color changes, to track session progress and maintain engagement.

#### 4.1.2.4 Mid-Session Difficulty Adjustment:

Researchers can pause the game to adjust difficulty settings based on user performance during the session.

#### 4.1.2.5 Data Collection and Export:

Key metrics like success rates and sensor thresholds are recorded and exported to a CSV file for analysis.

#### 4.1.2.6 Simultaneous Access to Game and Python Application:

The game screen and Python application remain accessible during sessions, enabling monitoring and adjustments without interruptions.

**4.1.3 Device Difficulty Adjustment:** Difficulty can be defined as the threshold the use of the exoskeleton must reach. Difficulty should be easy to modify and set for researchers with a default setting of one. The number one will represent the user's ability to move the exoskeleton. If they needed a lot of assistance then the threshold will be under one and if they didn't rely on the exoskeleton to assist then the threshold will be 1 or greater. Difficulty auto adjustment should also be implemented in the game, to promote the user to become less reliant on the device and not discourage the user. The auto adjustment should focus on increasing or decreasing the session difficulty based on user performance over time. This auto-adjusted setting should override any previously set difficulty. The researcher should be able to change the settings to auto adjustment during the game session as well.

#### Key Requirements:

#### 4.1.3.1 Calibration is Properly Set Beforehand:

The exoskeleton's calibration must be completed before gameplay to ensure accurate threshold settings for the session.

#### 4.1.3.2 Threshold Autoset After Calibration:

The system will automatically set the difficulty threshold based on the user's calibration data, ensuring it aligns with their current capabilities.

#### 4.1.3.3 Game Reflects Threshold Requirements:

Game mechanics will clearly indicate whether the user is meeting the required threshold, providing feedback on their progress.

#### 4.1.3.4 Auto-Adjustment of Thresholds:

The system will automatically adjust the difficulty during gameplay, increasing or decreasing the threshold based on the user's performance to encourage progress and independence.

#### 4.1.3.5 Manual Adjustment by Researchers:

Researchers can manually adjust the threshold settings or disable the auto-adjustment feature during a session to ensure the difficulty remains appropriate for the user.

**4.1.4 Calibration & Documentation:** Any code implemented for either the virtual controller or the rehabilitation games should practice modularity and be well documented both with comments and README files. The core idea is that the biomechatronics researchers should be able to easily understand how our capstone is structured for future development. With respect to session calibration, all researcher input must be implemented into the game session, giving full control of the session environment to the researcher and full control of the game to the user.

**Key Requirements:**

#### 4.1.4.1 Well-Defined README Files:

README files will be detailed, clearly explaining execution processes, program setup, and usage.

#### 4.1.4.2 Description of Core Functions:

The documentation will include clear descriptions of all core functions within the program to ensure researchers can easily understand the code structure.

#### 4.1.4.3 Examples of Program Usage:

Practical examples will be provided to demonstrate how the program can be used effectively in different scenarios.

#### 4.1.4.4 Easy Modification Instructions:

The documentation will ensure that instructions for modifying the program are simple, enabling researchers to make changes without confusion.

#### 4.1.4.5 Comprehensive Development Documentation:

All development aspects will be thoroughly documented to support future enhancements by researchers.

#### 4.1.4.6 Clear Usage Instructions:

The program will include straightforward instructions for both researchers and users to operate the system seamlessly.

## 4.2 Performance Requirements

**4.2.1 Response Time:** To meet performance needs, the device should deliver feedback at a rate of 500 Hz, ensuring feedback and game actions occur before the user's foot completes a cycle. Proper hardware allocation is crucial for achieving these response times, as inadequate allocation could delay feedback to the user.

**Key Requirements:**

#### 4.2.1.1 Provide Feedback Before Movement Cycle Completion:

The system must deliver feedback to users before they complete a full movement cycle, ensuring real-time responsiveness during gameplay.

#### 4.2.1.2 Proper Hardware Allocation and Initialization:

Hardware resources must be properly allocated and initialized to maintain high-performance standards and prevent delays in feedback.

#### 4.2.1.3 Minimum Feedback Rate of 500 Hz:

The system will deliver feedback at a minimum rate of 500 Hz to ensure smooth and accurate real-time interactions.

**4.2.2 Sensor Data Handling:** As noted, the program must handle sensor data effectively, providing feedback even if data is compromised. The exoskeleton device can track up to 10

sensors simultaneously and must support real-time data for each. In gamified mode, feedback will focus on two active sensors as per client specifications but we will use one sensor for the game. The researcher should be able to change the sensor according to which leg they want to focus on in real time.

**4.2.3 Game Launch & Ease of Use:** The game application should start within a minute and have an intuitive interface that allows researchers to quickly set up parameters and objectives. The GUI must be clear for both researchers and users, enabling easy monitoring of goals and performance feedback.

**Key Requirements:**

**4.2.3.1 Clear and Well-Structured GUI for the Python Application:**

The Python application will feature a user-friendly GUI that is well-organized and intuitive, allowing researchers to navigate and use its functions with ease.

**4.2.3.2 Search and Launch Multiple Games:**

The application will include functionality to search for and launch multiple games, streamlining the setup process for researchers.

**4.2.3.3 Clear and Well-Structured GUI for the Game:**

The game's GUI will be designed to provide users and researchers with a straightforward interface for gameplay and monitoring.

**4.2.3.4 Easy Parameter Setup:**

Researchers will be able to quickly set up parameters and objectives within the GUI to prepare for sessions efficiently.

**4.2.3.5 Game Monitoring from the GUI:**

The GUI will allow researchers to monitor game progress and user performance during sessions.

**4.2.3.6 Modify Game Parameters from the GUI:**

Researchers will have the ability to adjust game parameters directly from the GUI during gameplay, ensuring flexibility and adaptability.

### 4.3 Environmental Requirements

**4.3.1 Physical Environment:** Testing will occur in controlled environments like treadmills or other structured setups to simulate real-world conditions for rehabilitation. Safety protocols will be in place to prevent falls and aid users in the event of a fall. Cable management will be crucial to avoid harm from hardware malfunctions.

**Key Requirements:**

**4.3.1.1 Controlled Testing Environments:**

Testing will take place in controlled environments, such as treadmills or structured setups, to simulate real-world rehabilitation conditions effectively.

**4.3.1.2 Safety Protocols:**

Safety protocols will be implemented to prevent falls and to provide assistance to users in the event of an accident or device malfunction.

**4.3.1.3 Cable Management:**

Proper cable management will be ensured to avoid hazards or hardware malfunctions, maintaining a safe testing environment for users.

**4.3.2 Device Capabilities:** The exoskeleton has specific constraints, including a 30-minute battery life, which limits continuous gameplay. Motor torque restrictions affect the level of assistance, and the exoskeleton can support only two joints at a time, reducing complexity and conserving power by limiting the number of motors and sensors.

**Key Requirements:**

**4.3.2.1 Limited Battery Life:**

The exoskeleton has a battery life of 30 minutes, which restricts continuous gameplay to the same duration.

**4.3.2.2 Gameplay Duration Reflects Battery Life:**

Game sessions will be designed to align with the exoskeleton’s battery capacity, with a maximum gameplay time of 30 minutes per session.

#### 4.3.2.3 Utilization of Two Joints/Sensors:

To conserve power and reduce complexity, only two joints or sensors will be active at a time during gameplay.

**4.3.3 User Limitations:** The game design must account for users who may not be familiar with video games, focusing on simple actions throughout gameplay. Movements should be easy and low-impact, avoiding strenuous activities like running.

**Key Requirements:**

#### 4.3.3.1 Simple and Accessible Game Actions:

The game design will focus on straightforward, low-impact actions that users can easily perform, ensuring accessibility for individuals unfamiliar with video games.

#### 4.3.3.2 Alternative Actions Reflected in Gameplay:

The system will allow users to perform alternative but similar movements if needed, with these actions accurately represented in the game to maintain continuity and engagement.

**4.3.4 Hardware Compatibility:** Hardware limitations, such as processing resources, must be considered, along with accommodating users of various ages and sizes. The program will include fallback procedures for sensor anomalies, such as foot sliding or shoelace interference, to maintain reliable performance.

**Key Requirements:**

#### 4.3.4.1 Accommodate Participants of Different Sizes:

The system will be designed to support users of varying ages and body sizes, ensuring inclusivity and adaptability for a wide range of participants.

#### 4.3.4.2 Fallback Procedures for Sensor Anomalies:

Fallback mechanisms will be implemented to handle hardware or software failures, such as sensor issues caused by foot sliding or shoelace interference, ensuring reliable performance during sessions.

**4.3.5 System Compatibility:** Since this program is open-source, it must be compatible with a range of computer systems, including low-resource machines. Minimizing resource usage is essential due to the variability in device specifications. Bluetooth communication must maintain low latency to prevent lag and ensure reliable data transmission.

**Key Requirements:**

**4.3.5.1 Compatibility with All Operating Systems:**

As an open-source program, the system will be compatible with a wide range of operating systems to ensure accessibility for all users including Windows and macOS users.

**4.3.5.2 Support for Low-Resource Machines:**

The program will be optimized to run efficiently on low-resource devices, accommodating variability in hardware specifications.

**4.3.5.3 Low-Latency Bluetooth Communication:**

Bluetooth communication will be designed to maintain low latency, ensuring smooth, real-time data transmission without lag.

## 5. Potential Risks

While the gamified rehabilitation system offers many benefits, some risks could affect its performance and user experience. This section highlights key technical and hardware challenges, their potential impact on the user experience, and strategies we plan to implement to address these risks. Identifying and addressing these issues early will help ensure a smoother and more effective rehabilitation process.

### 5.1 Technical and Hardware Risks:

**5.1.1 Sensor Failure or Data Inconsistency:** Sensors may provide faulty data due to foot sliding, shoelace issues, or disconnections. This could lead to inaccurate feedback or unexpected game responses, impacting the user experience.

**5.1.2 Lag or Latency in Bluetooth Communication:** Lag between sensors and the game could disrupt real-time feedback, affecting user engagement and reducing the effectiveness of rehabilitation exercises.

### 5.2 Hardware Limitations:

5.2.1 Constraints such as battery life (30 minutes), limited motor torque, and joint movement restrictions may impact the duration and intensity of exercises, limiting rehabilitation effectiveness.

5.2.2 Extended or high-intensity exercises could lead to power exhaustion or overheating, causing sessions to be prematurely halted and impacting the user's rehabilitation schedule.

**5.3 Processor and Graphics Constraints:**

5.3.1 Low-performance computers may struggle with real-time plotting and feedback, leading to delayed or inconsistent responses, which could frustrate users or limit system applicability across devices.

**5.4 Usability and Accessibility Risks:**

**5.4.1 Overwhelming Users with Complexity:** Complex controls or too many inputs could overwhelm users, especially children or individuals unfamiliar with gaming. This may reduce the rehabilitation tool's effectiveness if users find it challenging to understand or control.

**5.4.2 Unintended Exertion Requirements:** If the game unintentionally requires movements beyond a user's current capabilities, it could hinder rehabilitation progress or increase injury risks, especially if difficulty adjustments aren't calibrated carefully.

**5.5 Development and Maintenance Risks:**

**5.5.1 Difficulty of Code Modifications:** Code complexity could hinder researchers from customizing game goals or thresholds, especially if documentation is lacking. Without a clear modular design, research teams may struggle to adapt the tool for different rehabilitation studies.

**5.5.2 Real-Time Feedback Requirements Not Met:** Failure to achieve the 500 Hz response rate or timely feedback (before foot contacts ground) could disrupt the user experience, potentially limiting the training tool's effectiveness in rehabilitation.

**5.6 Software or Game Logic Failures:**

Unexpected crashes or game logic errors could frustrate researchers and users alike, especially if software instability disrupts rehabilitation sessions.

### 5.7 Environmental and Safety Risks:

**User Physical Safety Risks:** Hardware malfunctions or software missteps could cause falls or unsafe conditions, especially in treadmills or active movement settings. This may be compounded by issues such as cable entanglement or falls in case of disconnections.

### 5.8 Risk Mitigation:

We will include a force stop and pause button to let users safely interrupt sessions if needed. Safety features will ensure any potential risks are managed effectively. To handle sensor issues, we will implement systems to record data smoothly, even during malfunctions. Bluetooth communication will be optimized to reduce delays and keep feedback consistent. For hardware concerns, we will monitor performance and ensure the exoskeleton device is working correctly. Clear instructions and safety guidelines will also be provided to improve reliability and user confidence.

## 6. Project Plan

The goal of this project is to create an engaging, interactive rehabilitation game for patients using a physical therapy tool that tracks and encourages user engagement through progressive levels and targeted exercises. The project will have planned out milestones. Each milestone will ensure the integration of the functional requirements we stated earlier in the document.

### 6.1 Requirement Specification

Week 1: Submit the draft of functional requirements by the end of the week. Begin gathering detailed requirements through discussions with the client. Define core functionalities, identify key performance indicators, and establish integration expectations with the game.

Week 2: Submit the finalized functional requirements. Conduct any follow-up consultations needed for clarity. Create a functional requirements document that outlines all key gameplay mechanics, UI needs, reward systems, input tracking, and data output functionalities.

### 6.2 System Design

Week 3: Begin developing the system architecture, including the Unity game/Pygame environment setup. Outline each game component (UI, inputs, data tracking) and define integration points for Unity and the built-in inputs. Develop a system blueprint detailing user flow, input processing, and reward system.

Week 4: Refine the system architecture based on client feedback. Finalize technical design specifications for each feature. Ensure that the design accommodates the real-time feedback requirements and supports progressive difficulty levels. Complete the system architecture document, including diagrams for integration points and data flow.

### 6.3 Initial Game Prototype Development

Week 5: Begin Unity development for the basic ‘Grape Smasher’ game. Set up the game environment, implement primary user interface components (start, pause, and success screens), and ensure UI elements align with usability guidelines for accessibility. We will start working on other games as well that have a similar concept to the ‘Grape Smasher’ game.

Week 6: Finalize the prototype’s game mechanics, allowing users to interact by smashing grapes. Ensure the UI components respond smoothly to user actions. Conduct basic functionality testing to verify that all primary interactions work as expected. Document any identified issues to be addressed in subsequent phases.

### 6.4 Input Integration

Week 7: Implement input handling, connecting the data of the device with Unity. Map physical interactions (e.g., pressure and motion) to game actions, enabling responsive real-time feedback. Test and calibrate the response rates to ensure a smooth gameplay experience.

Week 8: Complete integration testing for input accuracy and responsiveness. Conduct a round of functional testing to ensure that inputs consistently trigger the desired in-game responses. Document the results and adjust as needed to optimize real-time interaction.

### 6.5 Difficulty Implementation

Week 9: Develop different levels of difficulty for each game, so they can be set according to what the researcher wants before and during the game. Also, implement success metrics to track user progress and adjust in-game rewards accordingly.

Week 10: Conduct testing on each difficulty setting to confirm that they offer progressive challenges and accurate tracking of success metrics. Ensure each level runs smoothly and delivers appropriate feedback to users based on performance.

### 6.6 Reward System and Progress Tracking

Week 11: Develop the reward system, designing visual and auditory feedback that aligns with user success targets. Begin coding progress tracking to record user achievements across sessions.

Week 12: Integrate the reward system with progress tracking. Test for consistency in reward feedback and accuracy in tracking user progress, ensuring that users receive real-time updates on achievements and motivation to improve.

**6.7 Data Output**

Week 13: Implement data output functionality to export user performance data into CSV format, allowing researchers to review the data. Set up data parameters, such as session duration, success rates, and progress over time.

Week 14: Test data export accuracy by conducting several test sessions and verifying that the CSV files contain reliable and complete information. Adjust any formatting issues or missing data fields to ensure therapists can easily interpret the exported data.

**6.8 Usability Testing and Feedback**

Week 15: Conduct usability testing with real users and the client, observing interactions and noting feedback on UI, game mechanics, and input responses. Gather qualitative and quantitative feedback to get a better understanding of engagement levels and ease of use.

Week 16: Analyze feedback and identify areas for improvement. Develop a prioritized list of adjustments based on user input, including UI refinements, game mechanics tweaks, or input responsiveness improvements.

**6.9 System Refinements and Additional Games**

Week 17: Apply necessary adjustments based on usability testing. Implement minor refinements for a smoother user experience, including enhancements to accessibility features or user guidance within the game.

Week 18: Introduce one or two additional simple games to add variety, ensuring they integrate seamlessly with the system's tracking and reward mechanisms. Conduct initial tests to verify functionality.

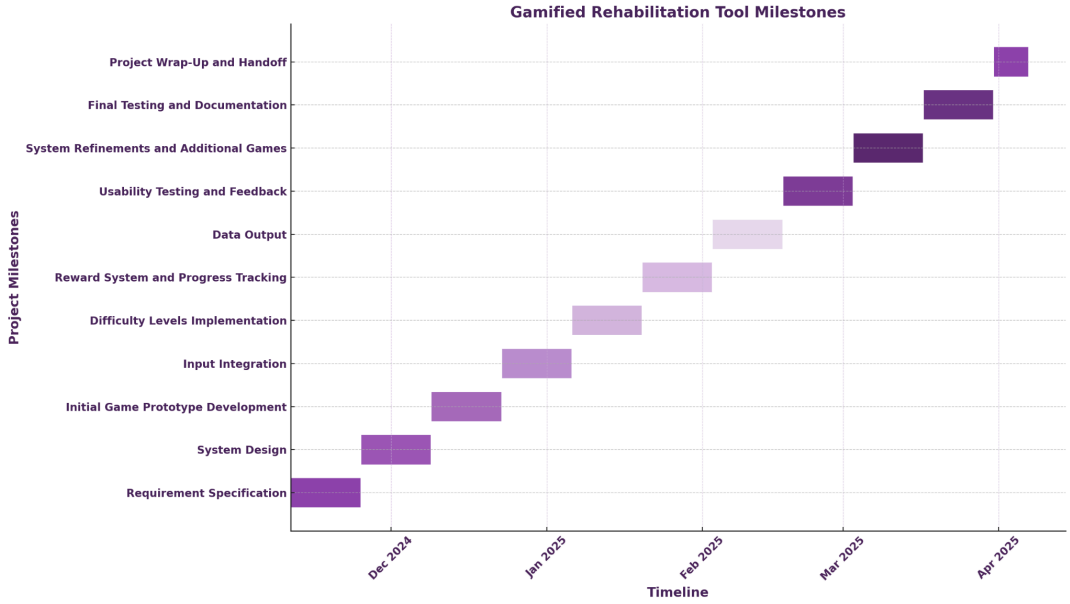
**6.10 Final Testing and Documentation**

Week 19: Conduct final system testing to confirm stability across all components and gameplay. Address any last-minute bugs or inconsistencies identified during the final testing.

Week 20: Prepare comprehensive documentation, including a user guide, technical specifications, and a setup tutorial. The documentation will support future maintenance and updates.

**6.11 Finalization of Project**

Week 21: Present the final project to the client and provide all necessary source files, data, and documentation. Conduct a final demonstration to showcase the game's functionality, usability, and rehabilitation effectiveness.




## 7. Conclusion

In conclusion, this project aims to create an engaging and adaptable rehabilitation tool by using Unity's flexibility to build an interactive game prototype integrated directly with the current device input tracking. Rehabilitation games are vital for helping patients stay motivated and committed to their therapy, and our design focuses on delivering a dynamic experience with real-time feedback and multiple settings of difficulty that the researcher/therapist can decide for the user. This document outlines the needed requirements, critical milestones, and a structured timeline for development, covering each phase from requirements gathering to final testing. We are confident this project will deliver an effective, user-centered tool that enhances engagement in physical therapy. This gamified solution not only supports patients in their recovery journey but also creates a foundation for broadening the tool's application across various rehabilitation areas, with the potential to transform future therapy experiences.