Motion: Virtual Physiotherapy Technical Documentation

by

Amine El Hammdaoui, Hoemeirra Abdoelrahman, Gavin Tjin-A-Soe,
Patrick Wals, Ilias Morabet, Susant Budhathoki

Amsterdam University of Applied Sciences
Developed using Unity 6 (6000.0.40f1)
June 2025

Table of Contents

Preface

This technical documentation was created to accompany the development of Motion: Virtual Physiotherapy, a VR application designed to assist users with lower back rehabilitation. It is intended for developers, academic evaluators, and technical stakeholders involved in the design, implementation, and future expansion of the project. The document outlines the system's architecture, development environment, core features, and implementation details to support both maintenance and continued development.

1 | Project Overview

Motion: VR Physiotherapy is an immersive virtual reality application developed to assist users suffering from lower back pain. By guiding users through physiotherapist-approved exercises in an engaging VR environment, the app aims to improve compliance, form accuracy, and rehabilitation outcomes.

Target Audience:

- Adults aged 18-35 with mild to moderate lower back pain
- Physiotherapists seeking tools to remotely monitor and guide their patients

Primary Goals:

- Provide real-time guidance and feedback during physiotherapy routines
- Collect movement data for performance tracking
- Enable physiotherapists to tailor routines based on patient needs

2 | System Requirements

Hardware:

- Meta Quest 2 or newer VR headset
- 2.0m x 2.0m minimum play space
- Wi-Fi connection for data sync (optional but recommended)

Software:

- Unity 6 (6000.0.40f1)
- XR Interaction Toolkit (v3.0+)
- OpenXR Plugin
- Custom-built backend (optional, for data sync)

Rendering Pipeline:

- Unity Universal Render Pipeline (URP)

Input System:

- Unity Input System Package (v1.7+)

3 | Application Architecture

Scene Hierarchy Overview:

- Main Menu Scene
- Balance Test Scene
- Routine Selector Scene
- Exercise Execution Scene

Key Components:

- **ExerciseManager:** Handles sequence of exercises and rest periods
- **NPCController:** Manages exercise demonstration using animations
- **UserTracker:** Tracks head and hand movement for posture analysis
- **FeedbackSystem:** Displays visual/audio cues during user performance

Modularity with ScriptableObjects: Each exercise is defined using a ScriptableObject which includes:

- Animation clip reference
- Repetition count
- Required hold duration
- Subtitle text
- Audio cue triggers

Code Structure:

-

4 | Exercise System

Balance Test:

- Mandatory initial test before routines unlock
- Measures user's stability using head movement variance
- Results stored locally or online

Exercise Flow:

1. NPC demonstrates exercise
2. Animation pauses at key hold point
3. User mimics movement
4. Visual timer and cue indicates when to reset
5. NPC continues next rep or rest period

Locked Routines:

- Users cannot access routines until they pass the balance test
- Unlocking logic handled by ProgressManager script

Feedback Types:

- Subtitles (toggleable)
- Visual highlights (incorrect form indicators)
- Optional audio encouragement

5 | User Flow

1. **Launch App**
2. **Intro & Safety Instructions**
3. **Balance Test** (required first-time use)
4. **Routine Selection** (if balance test completed)
5. **Guided Exercise Session**
6. **Session Summary / Cool Down**
7. **Automatic or Manual Exit**

6 | Data Logging & Integration

What is Logged:

- Balance test results
- Time spent on each exercise
- User posture deviations (basic tracking)
- Routine completion status

Data Handling:

- Stored locally with optional upload to central server
- Physios access data via dedicated web portal
- Based on results, physios can assign new routines or review progress

7 | Settings

Current Options:

- Subtitle toggle
- Audio cue toggle
- Routine difficulty presets

Planned Features:

- Voice command integration
- UI scaling for visual comfort
- Colorblind and contrast-friendly modes

8 | Development Notes

Known Issues:

- Some NPC animations may clip
- Subtitles not yet fully implemented
- Occasional floor alignment bug on Quest standalone mode

Feature Roadmap:

- Expand exercise library to include core and upper body routines
- Improve posture tracking accuracy with ML model integration
- Add physiotherapist-side app for in-VR consultations

Debugging:

- The application works in the Unity editor without a headset
- The developers can skip exercises by pressing the spacebar during a routine

9 | Glossary & Appendix

ScriptableObject: Unity asset type used to store modular data
Repetition Count: Number of times an exercise is performed
NPC: Non-playable character used for demonstration purposes
XR Interaction Toolkit: Unity package for VR interaction handling

10 | Store and fetch data using the API

To store or fetch data from the server, navigate to the service/ folder. Each service class in this folder contains the logic for making structured API calls to the backend. For local development, make sure to update the baseUrl in the ApiClient to: http://localhost:8080/api and that the backend is running.

Important: When sending data to the server, ensure that your **Data Transfer Object (DTO)** in the frontend (Unity) exactly matches the corresponding DTO in the backend.
You can refer to the existing methods in the service files as examples.

11 | Authentication

To track your progress, you must log in. In the Unity app, users can log in using their username and pincode. After successful authentication, the JWT token returned by the server is stored in PlayerPrefs. Additionally, the username is stored locally in a JSON file to simplify the login process the next time the app is launched. Navigate to the files **LoginManager.cs** and **User.cs** to see how this functionality has been implemented.