D6.3
INTEGRATED PROTOTYPE DEMONSTRATION

Grant Agreement nr	**856998**
Project title	Personalised recovery through a multi-user environment: Virtual Reality for Rehabilitation
Project Acronym	PRIME-VR2
Start day of project (dur.)	October 1st 2019 (3 years)
Document Reference	PRIME-VR2_D_WP6_KRL_D6.3-Integrated prototype demonstration
Type of Report	PU
Document due date	30/09/2021
Actual date of delivery	30/09/2021
Leader	KRL
Responsible	Gábor Vadász (KRL)
Additional main contributors (Name, Partner)	Robert Sarosi, KRL Alexander Baranov, CPD Anthony Demanuele, FSG Matthew Azzopardi, FSG David Swapp, UCL Felix Thiel, UCL
Document status	[Reviewed by Philip Farrugia (UOM)]

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 856998

Table of contents

VERSION HISTORY

COMMENTS	*RESPONSIBLE*	*VERSION*	*DATE*
First draft	Gabor Vadasz	1.0	16/09/2021
Review of the first draft	Philip Farrugia	1.0	22/09/2021
Final version	Gabor Vadasz	1.0	25/09/2021

EXECUTIVE SUMMARY

The Integrated Prototype Demonstration of the PRIME-VR2 platform contains detailed information on the work the WP6 team achieved in the recent period.

Besides explaining the technical background used for the implementation of the Web Platform, the Unity SDK and the Games, this deliverable provides more information about the achieved performance metrics and validation criteria.

Deliverable 6.3 is the result of the discussion of all the technical challenges of the project made by WP6 members. Discussions were extremely important to define the initial list of business and technical requirements for the overall Platform, and for each single component. The three Platform components have been detailed in this document: Web Platform, VR Ecosystem, and Rehabilitation Games.

Web Platform

This deliverable contains detailed information on the technical background used for the implementation of the Web Platform, including reasons for the selected technology stack. A detailed description is provided on the software components used and on the Cloud infrastructure, together with the deploying process.

VR Ecosystem

The VR ecosystem consists of the Unity SDK, Platform Loader, and the VR games. This deliverable contains detailed information regarding the integration of the Platform Loader and the VR games with the use of the Unity SDK.

Results achieved in WP6 till M24

- Submission of deliverable D6.1 Platform Implementation Plan
- Communication and management software and routes were discussed and agreed on
- 7 focus group meetings have been organised to collect users' stories
- Definition of the requirements of the PRIME-VR2 Web Platform
- Completion of the specification of the PRIME-VR2 Web Platform and its functionalities
- Definition of the PRIME-VR2 Web Platform design mock-ups, including the modules and communications, with the definition of use cases and scenarios of interaction among the components
- 22 meetings for WP6 team have been organized for discussing the platform implementation plan
- Creation of Web Platform functional and technical documentation.
- Presentation of the coding Guidelines to the team to suggest changes and guidelines
- The Unity SDK functionality has been abstracted into a set of separate modules that will be used as technological standard (platform loader, service module, game module, controller input module)

1 INTRODUCTION

1.1. Terminology

Term	Meaning
Game Engine	A game engine is a software-development environment designed for people to build video games. Developers use game engines to construct games for consoles, mobile devices, and personal computers.
Unity3D	Unity is a cross-platform game engine.
Unity SDK	A software development kit (SDK) is a collection of software development tools in one installable package. They ease the creation of applications by having compiler, debugger and possibly a software framework.
Web Portal	Web Portal is a web application that coordinates the activity of hospital supervisors, doctors and patients in order to achieve successful VR therapy outcomes.
API	An Application Programming Interface (aka API) is an interface or communication protocol between different parts of a computer program intended to simplify the implementation and maintenance of software. An API may be developed for a web-based system, operating system, database system, computer hardware, or software library.
API Key	A special string that is functioning as a secret password between the API and the API caller. This secures the API so unauthorized calls would not be processed and can help to prevent data leakage.
REST	Representational State Transfer (REST) is a software architectural style that defines a set of constraints to be used for creating Web services or API. Web services that conform to the REST architectural style, called RESTful Web services, provide interoperability between computer systems on the Internet.
JSON	JSON (JavaScript Object Notation) is a lightweight data-interchange format. It is easy for humans to read and write. It is easy for machines to parse and generate. It is based on a subset of the JavaScript Programming Language Standard ECMA-262 3rd Edition - December 1999. JSON is a text format that is completely language independent but uses conventions that are familiar to programmers of the C-family of languages, including C, C++, C#, Java, JavaScript, Perl, Python, and many others. These properties make JSON an ideal data-interchange language.
User	User is a person who has some level of access in the system. The user interacts with the system and get results based on permissions.

User Account / Profile	The User Account / Profile contains personal (e.g.: Name, Email) and non-personal information (e.g.: UserID, Group, Permissions) about the User.
Group	A Group is a logical set of Users. It has a Group Name and a Group can have a special set of permissions what defaults to all Users who are included in the Group.
Permission	All available operations in the system and whether different user types can perform them.
Authentication	The Authentication is the process when the system checks the User's credentials. This is a gatekeeper to secure the system from unwanted access. This is mostly happening when the User logs in or getting back to the system after a while.
Authorization	The Authorization is the process when the system checks the User's permissions. This is a gatekeeper to secure the system from unwanted access. This happening at every API call.
JWT	JSON Web Tokens are an open, industry standard RFC 7519 method for representing claims securely between two parties (mostly between client and backend or backend and backend communications). Built on top of JSON.
C#	C# is a general-purpose, multi-paradigm programming language encompassing strong typing, lexically scoped, imperative, declarative, functional, generic, object-oriented, and component-oriented programming disciplines. Developed and maintained by Microsoft. Developers using C# to develop a Unity3D application.
NodeJS	Node.js is an open-source, cross-platform, JavaScript runtime environment that executes JavaScript code outside of a browser.
Container	A Container is a standard unit of software that packages up code and all its dependencies, so the application runs quickly and reliably from one computing environment to another. Containerization is the process when an application is packaged together as a container to be able to run in a virtual environment.
Docker	Docker is a set of platforms as a service products that use OS-level virtualisation to deliver software in packages called containers. Containers are isolated from one another and bundle their own software, libraries, and configuration files; they can communicate with each other through well-defined channels.
Kubernetes	Kubernetes is an open-source container-orchestration system for automating application deployment, scaling, and management. It was originally designed by Google and is now maintained by the Cloud Native Computing Foundation.
Version Control	A component of software configuration management, version control, also known as revision control or source control is the management

System	of changes to documents, computer programs, large web sites, and other collections of information.

2 TECH STACK

2.1. Programming Language

There is a wide variety of programming language available today. Older and more robust ones like C / C++ and newer ones like C#, NodeJS, Go which provides better solutions for today's coding problems (async, multi-threading, heavy network communications). There is no ideal programming language, but some are better than others when facing specific challenges. With a monolithic approach, the developer chooses only one programming language which is used exclusively in the whole software package. Another hallmark of monolithic approaches is that all is contained in one software package. In polyglot programming the developer has the freedom to write different parts of the software in different languages. This way the developer can leverage most of the benefits of each language. A danger of polyglot approaches is that the codebase becomes fragmented in many different languages and requires a variety of skillsets to maintain.

In the PRIME-VR2 project the polyglot approach was chosen but limiting the number of languages to a small set: C#, NodeJS. For certain heavy data and network traffic handling, using the language Go language is possible in the future.

C# is developed and maintained by Microsoft. Because it is wide adoption in education it is easy to find a C# developer in any seniority level. Also, the chosen game engine, Unity3D using C# as its main programming language.

NodeJS is a free, open-source language built on top of Javascript. One of the ideas behind the language is to reuse the front-end web knowledge when developing the backend. Most of the web programmers know how to code in Javascript. This makes it easier to find developers and maintain the code base. Javascript also has a huge online community, which makes finding answers and solutions easy.

Go is an open source, compiled and statically typed programming language developed and maintained by Google. The main goal is to reach the C++ performance but without the difficulties what C++ could mean. Go is only a few years old language, but it is gaining popularity(https://trends.google.com/trends/explore?date=all&q=golang, https://opensource.com/article/17/11/why-go-grows, https://blog.jetbrains.com/go/2021/02/03/the-state-of-go/). It is offering a simple syntax, a fast compiler and supports concurrent programming from the very beginning.

3 UNITY SDK

3.1. Introduction

This section further describes the integration of the PRIME-VR2 Unity SDK as described in deliverable 6.1. This section will go into detail about the implementation of the SDK and how it is integrated to establish a connection between the Platform Loader, VR Games, and the backend. Figure 1 depicts this implementation.

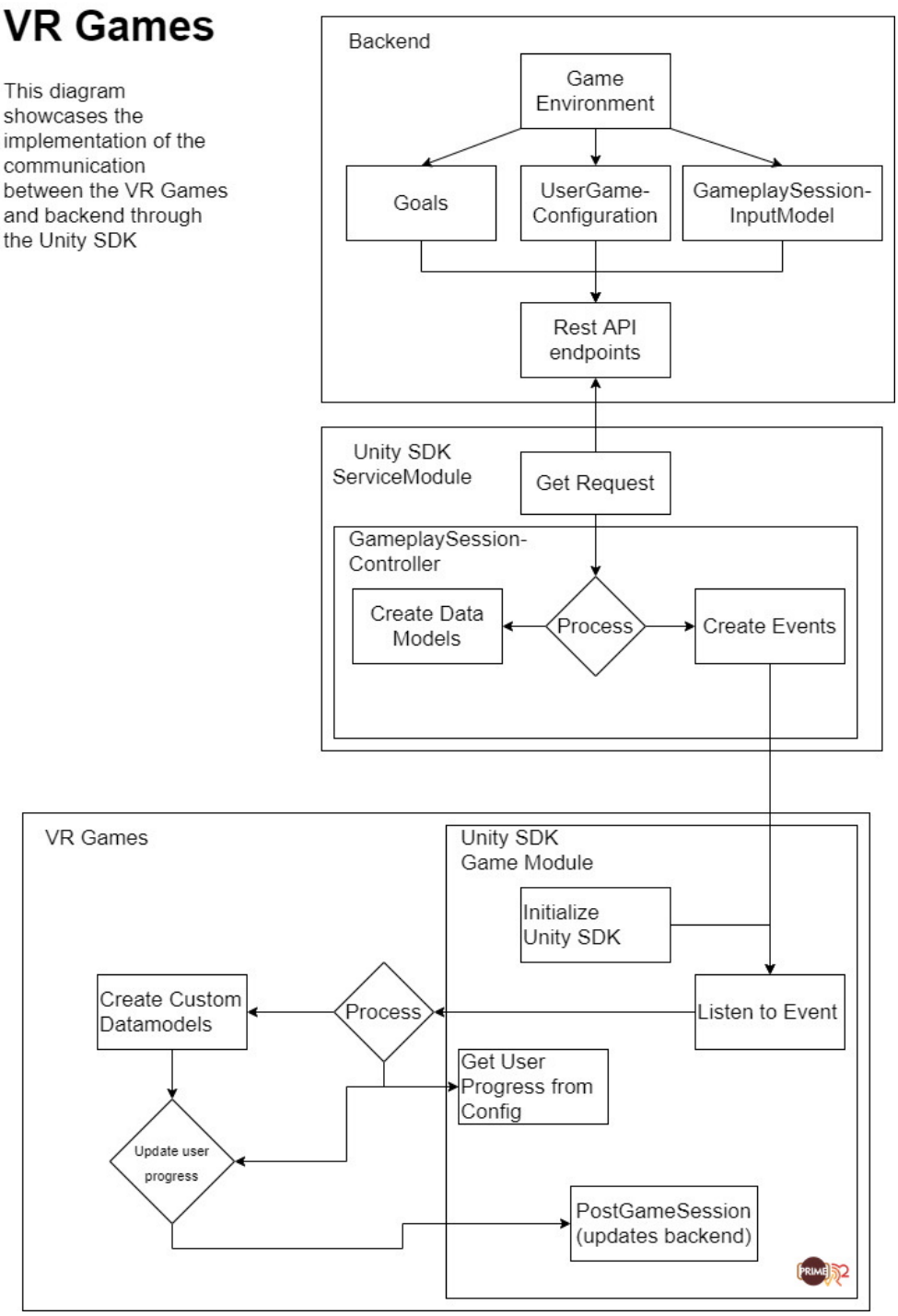

Figure 1 – Unity SDK integration with backend and VR games

3.2. Platform Loader

The platform loader is an application used to handle the user authentication and game management. The platform loader is our distribution platform to ensure each user can download and install the games assigned to them. The platform loader can be downloaded from the web portal as shown in Figure 2.

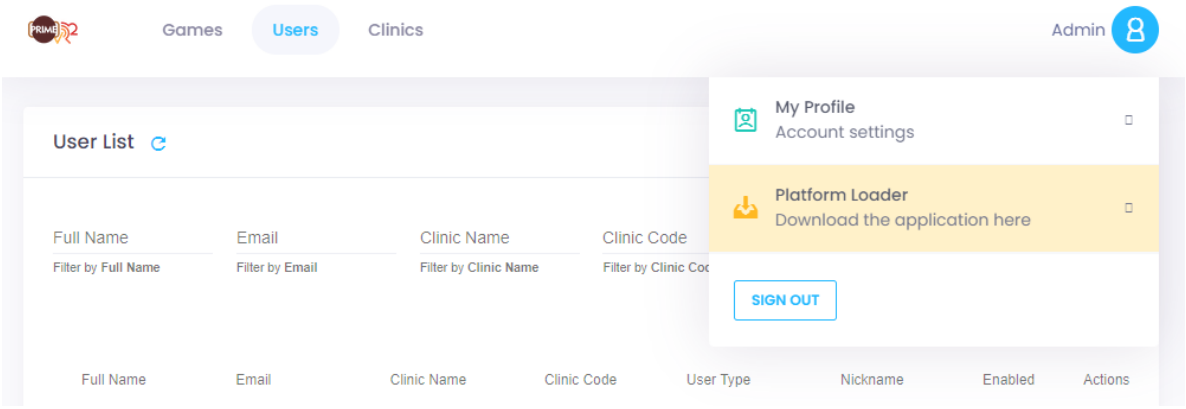

Figure 2 – Web portal Platform Loader download location

Users can log in through the platform loader by using a login screen. A temporary user token is locally stored for the logged in user to be used within the VR games. This token is used to retrieve the list of games that are assigned to the user as shown in Figure 3.

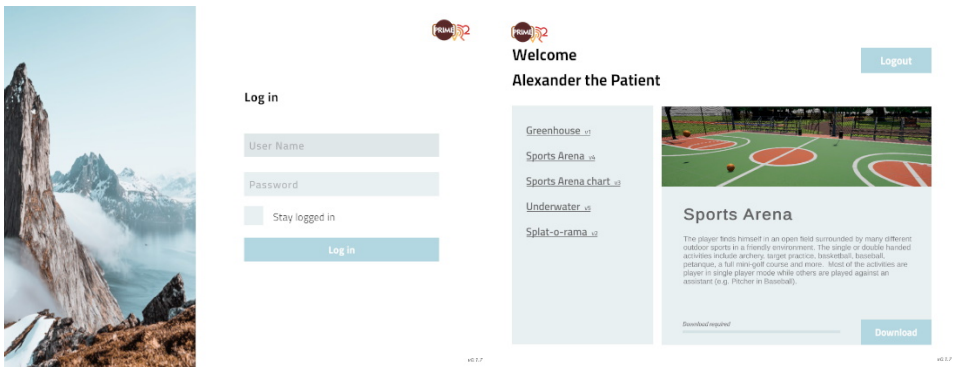

Figure 3 – Login and Game selection screens

The users can select one of the games assigned to them and download/install the VR game application. Once the VR game is set up for use, the user can run it through the platform loader. As shown in Figure 4, the platform loader feeds the token to the VR game through the Prime-VR2 Unity SDK when a user runs the VR game. This token can be used within the VR games to authenticate future calls to the backend to retrieve or save data relevant for the backend.

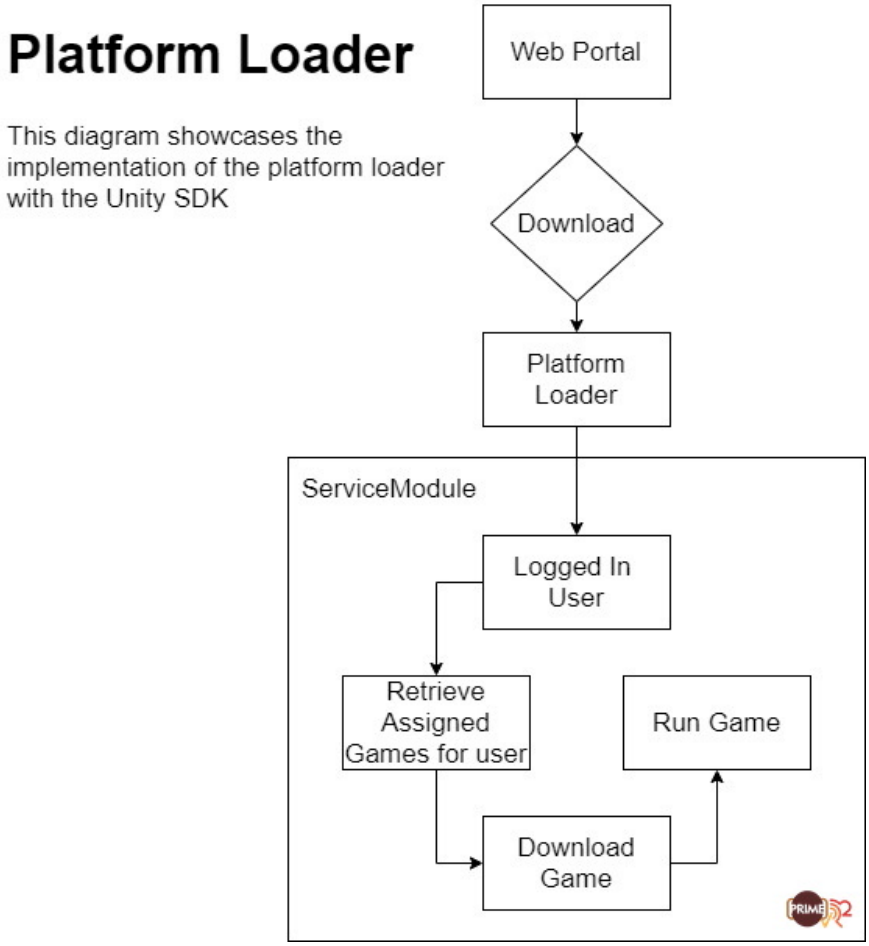

Figure 4 – Platform Loader flow

3.3. VR games

To retrieve or send the progress of the current user, there first needs to be a connection established within the VR game with the backend. This can be done through the GameplaySessionController.cs class. The SDK has an example scene, "post-gamesession.scene", showcasing the usage of this class. When the GameplaySessionController is initialized, it will establish a connection with the current user token which has been received from the platform loader. The GameplaySessionController has events which the VR game can listen to to handle any necessary logic when they are triggered. These events are necessary to process data like user progress or required exercise duration. The following list is a brief overview of the available events.

Event name: OnInitialized
Event description: This event is invoked right after the SDK is done initialization. This includes the logging in of the current active user.

Event name: OnConfigJsonSchemaReceived
Event description: This event is invoked right after the SDK has received the raw data containing the configuration set up for the current user and the current VR game. This is useful

to process custom logic required for the VR game which the SDK could not predict beforehand because of the customisable nature of the VR game. For example, if an exercise has a specific scoring or measurement logic, like holding a pan stable for a certain amount of time, a variable 'held_pan_stable' can be set up for a specific game. This data is sent as raw data from the backend and exposed through the Unity SDK for the VR game environment. Storing custom data would not be useful for quick development iterations and therefore we have chosen to expose the raw data to ensure the VR games can go through quicker iteration cycles.

Event name: OnConfigProcessed
Event description: This event is fired right after the SDK has received the raw data containing the configuration setup for the current user and the current VR game and has processed this raw data into usable default data models setup by the SDK. These data models include information regarding the progress of the user and the custom configuration for the user.

Event name: OnUserInfoProcessed
Event description: This event is fired right after the user has logged in and their profile has been loaded. This is useful to display personalised information, like nickname, when displaying the progress of the user within the VR game.

Event name: OnGameDetailsProcessed
Event description: This event is fired right after the personal game details have been retrieved and processed. The personal game details contain information like recommended playtime and sessions per week and the goals. This is useful to display this kind of information for the user within the VR game to ensure the user has a clear indication of the amount of time they need to spend on exercises.

3.4. Backend

The VR games are set up on the web portal with the use of configurations. These custom configurations decide what kind of settings and progress can be made for each VR game.
The SDK contains default data models for these configurations with the option to add custom data models for the customized VR games.
The SDK allows data to be updated once the user has achieved a certain milestone or any progress at all within the VR game. This is accomplished by cumulatively adding the necessary data throughout the VR game session, and finally sending it to the backend. The GameplaySessionController.cs has a variety of functions available for this. Below is a list of some of the functions which we expect to be used the most.

Function: AddAchievement
Parameters: AchievementData
Returns: nothing
Logic: Adds the achievement which has been unlocked by the user and aggregates it into a list, preparing it to be sent to the backend.

Function: AddScore
Parameters: int/number
Returns: nothing
Logic: Adds the score obtained by the user and aggregates it into a list, preparing it to be sent to the backend.

Function: AddDuration
Parameters: int/number
Returns: nothing
Logic: Adds the duration which the user has spent within the VR game and aggregates it into a list, preparing it to be sent to the backend.

Function: AddGameplaySessionInputVariable
Parameters: GameplaySessionInputVariable
Returns: nothing
Logic: Adds the GameplaySessionInputVariable (which is a custom definition for each VR game containing any kind of useful score, progress, feedback, difficulty setting) for the user and inserts the value so that it can be aggregated into a list and prepared to be sent to the backend.

On top of the above functions, the GameplaySessionController has a variety of Get and Set functions and properties available for ease of use for developers. The full functionality can be found within the code documentation.

Once the user is done with an exercise the GameplaySessionController.PostSession function can be used to send all of the aggregated data to the backend so that it can be displayed in the web portal. This includes progress reports and charts which have been setup in a custom manner for each VR game. The web portal will then display the received data in the predefined format for each specific VR game.

4 GAME IMPLEMENTATION

4.1. Introduction

As documented in D6.1, the work done in Phase 1 encompasses the discovery phase around the needs of the living labs within the consortium. This involved multiple brainstorming sessions to determine the ideal environment scenarios where patients will eventually play the rehabilitative games in virtual reality as part of the VRHAB-IT platform. This phase led Flying Squirrel Games ltd., to the identification of 10 scenarios which were followed up with a written description of each, conceptualised in art form to determine layouts, views, scales and activities within each of these unique places. A video recording for each of these game scenarios can be seen here[1].

[1] https://tinyurl.com/primevr2-scenarios

Later on, each of these was 3D modelled and shown to stakeholders within the consortium to gather feedback specifically from WP7, make the necessary changes and eventually demonstrate as navigable areas in virtual reality. A short-listing selection process started earlier this year with 2 living labs choosing their environment scenarios for the following selection of mini-games.

KNRC chose the Crafts room and the Greenhouse concepts which were then merged into a single environment hosting the first game in that environment called 'Garden Master' (see section 4.3.3). STJH chose the Sports arena as their first environment which led to the development of Splatorama (see section 4.3.1), Shape Shifter (see section 4.3.2) with other mini-games in progress (see section 4.4). GDIH are still awaiting ethics approval by the time of writing but clinicians within this particular living lab have already indicated their need for games that help with improving the quality of life in daily functions in a kitchen environment. This led to the development to Steady Hands (see section 4.3.4).

Several other simple activities were prototyped as a showcase to GDIH and to support the development of the be-spoke controller design. These were only used for demonstration purposes and hence they will not be listed with the 4.3 mini-games section below. The most noteworthy ones to mention here are the water carrying demoes (Figure 6), the stacking dice in cup and the freewriting with white board markers at the office environment (Figure 5).

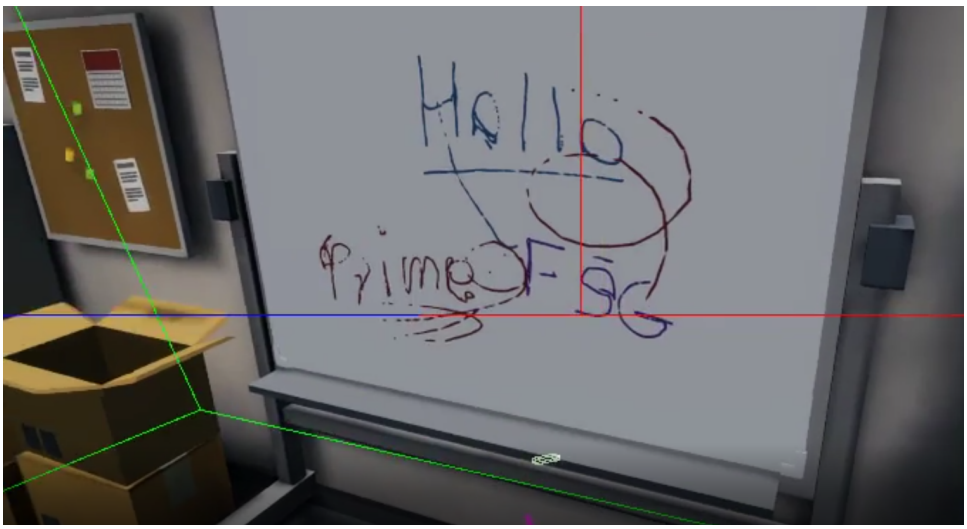

Figure 5 - Writing at office[2]

[2] https://tinyurl.com/writingatoffice

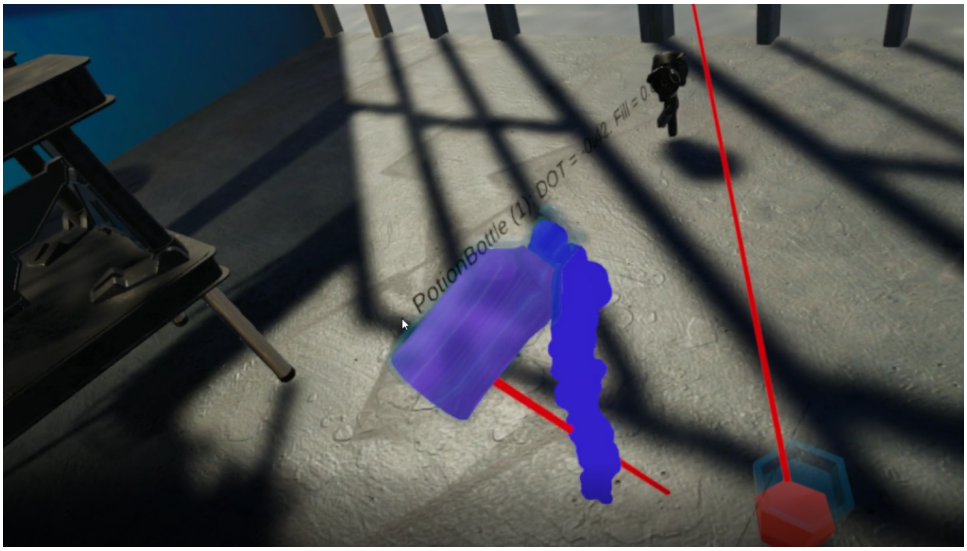

Figure 6 -Liquid test[3]

4.2. Game vs Exercise matrix

The following matrix shows the exercise focus areas for the games described in the following sections. The matrix highlights the primary focus in dark green, the secondary area in lighter green and the unused areas in light blue. As the bespoke controllers evolve, Table 1 is being updated to match these capabilities. For instance, the Garden Master game is meant for post-stroke patients through the KNRC Living Lab. At the time of writing, this game is being updated so it can be played using the wrist module as well as the finger module (separately). The same applies to Splatorama.

Table 1: Game vs Exercise matrix

Game	Shoulder	Elbow	Wrist flexion	Wrist extension	Fingers
Splatorama (STJH)					
Shape Shifter (STJH)					
Garden Master (KNRC)					
Steady Hands (GDIH)					
Basketball (STJH)					
Mini-Golf (STJH)					
Petanque (KNRC)					
Rhythm in the Kitchen (GDIH)					

Legend:

Primary	Secondary	Not used

[3] https://tinyurl.com/liquidTest

4.3. Phase 2 - Mini games

4.3.1. Splatorama

This mini-game sets the Player in a virtual target shooting range set in a forest diorama inside the Sports Arena. Armed with a hypothetical paint-blob-firing blaster gun the Player is tasked with hitting as many cardboard cut-out animals as possible as they pop in and out of the immediate view.

The Player's blaster can shoot paint in two colours: orange and blue - only targets hit with the matching colour paint will register as hit. This information is relayed to the player in a brief tutorial before the game starts as seen in Figure 8. The Player will thus have to prioritize which targets to hit as well as to hit them with the corresponding paint colour (see Figure 9 and Figure 10). Aiming is done by moving the arm and flexing/extending the wrist to determine the colour of the paint being projected.

Each session has a predefined set of targets that appear in different placements and exist for a definite amount of time before disappearing to make way for the remainder of targets in the set. The session ends when the last target has expired or has been taken down. The exception to this rule exists in Easy Mode where targets will not expire until they are hit by the Player. In this manner, progress through the targets is governed by the Player's natural pacing. With respect to the other games available, Splatorama differentiates itself as one of the most challenging ones, in part because the Player can never tell what patterns and colours the targets will have.

The game can be configured in a number of ways:

1. Number of sessions per week
2. Number of targets per session (defined by pre-sets rather than perfectly granular)
3. Allow targets to expire / hold targets until knocked down by player
4. Speed of targets
5. Target hit-points amount

The following data can be stored and visualised on the VRHAB-IT platform as seen in Figure 7

1. Time taken for Player to complete whole session
2. Total number of shots from player
3. Hit versus Miss shot ratio (accuracy)
4. Hit versus Miss shots for correct target colour
5. Targets missed (applies only when targets expire)

Figure 7 - Sample player data as seen on the VRHAB-IT platform.

Difficulty:

The difficulty of this mini-game hinges on two factors: the speed at which the targets move and the time for which they remain visible before popping back down. As such the difficulty modes are:

1. **Easy**: Target movement speed at 50%, targets remain visible until hit
2. **Medium**: Target movement speed at 100%, targets remain visible until hit
3. **Hard**: Target movement speed at 100%, targets disappear after a few seconds
4. **Pro**: Target movement speed at 150%, targets disappear after a few seconds

Scoring:

Consistent across all mini-games, at the end of a session a score from 1 to 100 is given to the Player. The final score considers primarily the amount of good hits versus the total hits fired (the time taken to clear a session is not factored in).

Figure 8 - Tutorial mode

Figure 9 - Blue paint

Figure 10 - Orange paint.

Splatorama footage is available for sample gameplay[4] and the introductory tutorial[5].

4.3.2. Shape Shifter

In this mini-game in the Sports Arena, the Player is presented with a set of predefined 3D shapes that represent a line path through 3D space. The objective is to have the Player trace his hand through the path as accurately as possible, the action is akin to tracing a gesture with the controller-mounted hand. This shape can be fairly planar or it can wrap around the facing hemisphere of the Player.

Each gesture shape has a starting and an end point. Tracing begins by positioning the hand where the shape begins and as the hand starts to follow the projected path the shape starts taking form. Should the tracing stray too far from the projected path then the Player is allowed to resume from his/her last known good position. Alternatively, an easy variant of the game

[4] Link: https://tinyurl.com/splatoramaGameplay

[5] Link: https://tinyurl.com/splatoramaTutorial

exists for beginner Players to get accustomed to - in this mode the Player is allowed to trace the shape in any order irrespective of beginning and end. Information on how to play is conveyed to the player at the start of the session through a voice over clip.

Gameplay is split into sessions, where each session consists of a set of gestures that the Player has to trace and follow as accurately as possible. A gesture is considered complete once it has been traced from start to finish. Similarly, a session is complete when every gesture in the set has been completed. There are currently 8 distinct gestures with the possibility of adding more complex gestures in the future. Furthermore, this number can be multiplied when gestures are rotated and/or mirrored, where applicable.

The game can be configured in a number of ways:

1. Sessions of games per week
2. Number of shapes per session
3. Minimum distance from path for input to be valid
4. Size of brush
5. Difficulty sets of shapes (complex shapes inherently more difficult to trace)
6. Use Continuous or any-sequence style shapes

The following data can be stored and visualised on the VRHAB-IT platform as seen in Figure 11

1. Time taken for Player to complete the whole session.
2. Number of times tracing was broken due to over threshold, per shape
3. Accuracy: measure of how close the hand position was to the midpoint of each shape joint, per shape/session cumulative

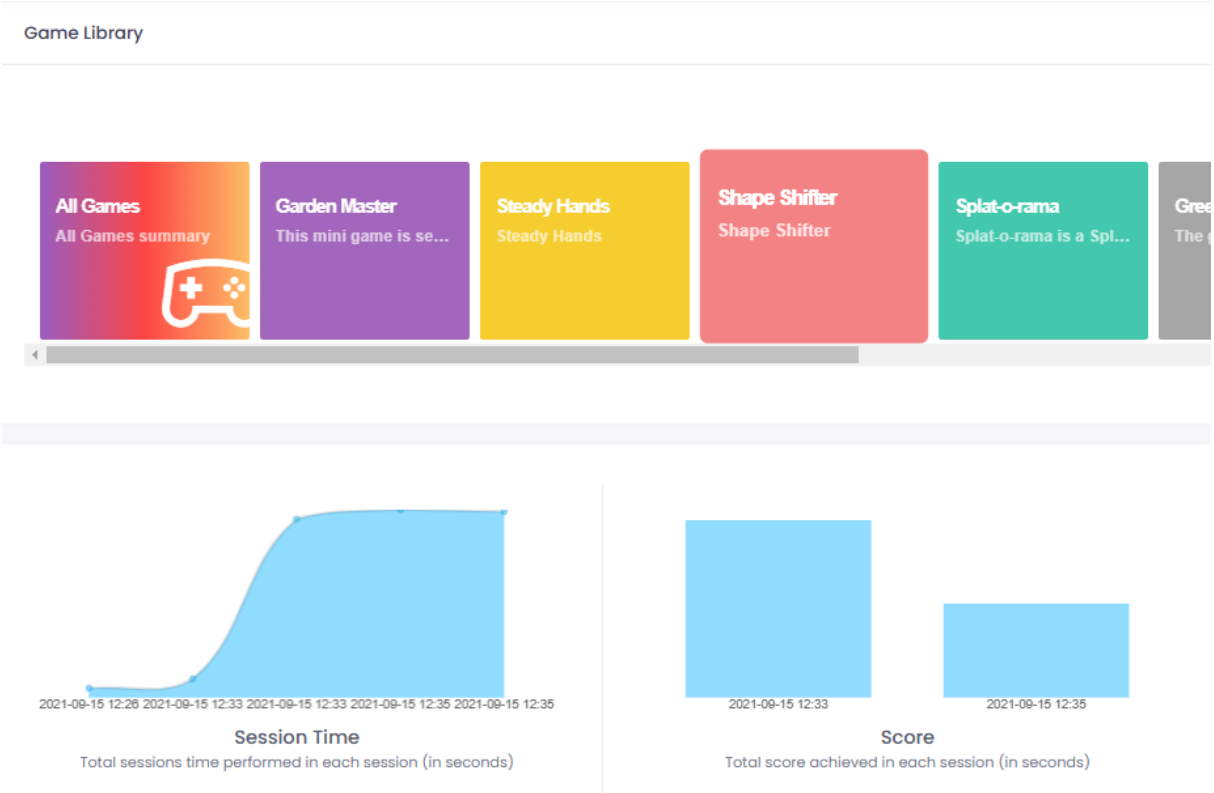

Figure 11 - Sample player data as seen on the VRHAB-IT platform.

Difficulty:

There are 4 difficulty modes, each based on a combination of brush sizes and whether or not a shape's path needs to be traced continuously from beginning to end as can be seen in Figure13, Figure 14 and Figure 15. The modes are:

1. **Easy**: Fill all shapes in any given sequence using Normal brush size
2. **Medium**: Fill all shapes in any given sequence using small brush size
3. **Hard**: Fill all shapes continuously starting from Green to Red markers (Normal brush size)
4. **Pro**: Fill all shapes continuously starting from Green to Red markers (Small brush size)

Scoring:

Scoring is based on two primary factors: time taken to clear a given sequence and the hit/miss ratio of shape segments (note that the concept of a 'miss' does not exist in Easy and Medium difficulty modes). For each session this data is processed through a formula to produce a final score from 1 to 100. A Rank (from A to D) is associated with ranges on the final score and presented to the Player at the end of a session. Working examples and tuneable values to adjust difficulty levels for this can be seen in Figure 12 and are available on this spreadsheet[6].

	Input	Workings							Score
Example	Variable Session Time	MinAverageTime	SessionTime - MinAverageTime	MaxAverageTime	SessionTime - MaxAverageTime	Result + Invalid/ValidHits	1- Result	Clamp 0 to 1	Final result base 100
1									
2	20	45	-25	60	-0.42	-0.37	1.37	1.00	100.00
3	30	45	-15	60	-0.25	-0.20	1.20	1.00	100.00
4	40	45	-5	60	-0.08	-0.03	1.03	1.00	100.00
5	50	45	5	60	0.08	0.13	0.87	0.87	86.67
6	60	45	15	60	0.25	0.30	0.70	0.70	70.00
7	70	45	25	60	0.42	0.47	0.53	0.53	53.33
8	80	45	35	60	0.58	0.63	0.37	0.37	36.67
9	90	45	45	60	0.75	0.80	0.20	0.20	20.00
10	100	45	55	60	0.92	0.97	0.03	0.03	3.33
Notes	ValidHits	100							
	InvalidHits	5							

Figure 12 - Workings and samples of typical game play scenarios.

[6] Link: https://tinyurl.com/shapeShifterExamples

Figure 13 - Wave shape with start and end point

Figure 14 - Spiral with start and end point

Figure 15 - Shape that requires reaching out in three dimensions

Shape Shifter footage is available for the Tutorial[7], Medium[8] and Hard[9] modes.

4.3.3. Garden Master

This mini-game is set in a gardening environment where it requires the Player to carefully water and grow a set number of plants until they flourish. Watering the plants is done via a water hose which can be picked up by the Player and directly aimed at patches of land containing the seeds that will eventually grow into plants when watered.

As plants are watered, they gradually begin to grow. Each plant displays its own 'water reserve meter' indicating how much water it currently has on its dirt patch. The meter also shows an ideal level of water that results in optimal growth (labelled in green) as can be seen in Figure 19: below this level results in poor growth (labelled with neutral colour) while above it (labelled in red) will result in no growth at all. The idea is thus to manage water such that all plants are kept happy and growing optimally.

To water the plants the Player must not only aim the water hose (see Figure 18) using general arm positioning but also use flexion/extension of the wrist/finger motions to maintain water pressure so as to be able to keep watering the plants. The drop in water pressure is indicated by the range of the water jet, the Player will not be able to water the plants further back without producing a few pumping motions.

Each session contains a predefined number of plants that the Player will need to grow during that session. There are 5 patches of land, 2 in close proximity to the Player while the remainder a step further away, with each patch hosting one plant seed. Once a plant is grown to its full extent it is removed from the patch and replaced with the next seed in the list. Once all

[7] Link: https://tinyurl.com/Shape-Shifter-Tutorial
[8] Link: https://tinyurl.com/Shape-Shifter-Medium-Mode
[9] Link: https://tinyurl.com/Shape-Shifter-Hard-mode

seeds/plants have been grown the session is complete and the Player will be presented with his overall performance and score on a virtual billboard as seen in Figure 20.

The game can be configured in a number of ways:

1. Number of sessions per week
2. Number and type of plants per session
3. Amount of pressure added with each flexion/extension motion

The following data can be stored and visualised on the VRHAB-IT platform as seen in Figure 16

1. Time taken for Player to complete session
2. Time taken for each plant to grow
3. Hit versus miss on all plants (cumulative)
4. Number of flexions/extensions performed

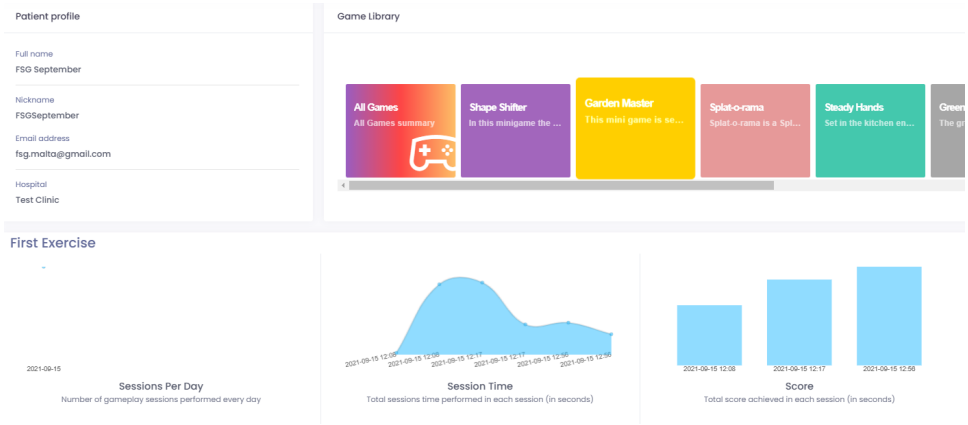

Figure 16 - Sample player data as seen on the VRHAB-IT platform.

Difficulty:

The difficulty of this game is primarily governed by the plant's water reserve meter's red, green and neutral zones. As a plant is watered the amount of water that plant currently has is represented by a blue bar in the meter. This bar gradually drops as water is absorbed by the plant to grow. Having the water bar in any colour region except for green will not optimally grow the plant, therefore the difficulty curves alter how difficult it is for the Player to keep the water bar in the green region. Higher levels of play make it more difficult to grow the plant optimally by shrinking the green optimal zone while increasing the red zone.

- **Easy**:
  - Red zone range: 10-20%
  - Green zone range: 50-80%
  - Number of plants to grow: 3

- **Medium**:
  - Red zone range: 15-30%

- Green zone range: 40-70%
- Number of plants to grow: 4

- **Hard**:
  - Red zone range: 20-40%
  - Green zone range: 30-60%
  - Number of plants to grow: 4

- **Pro**:
  - Red zone range: 30-50%
  - Green zone range: 20-50%
  - Number of plants to grow: 5

Scoring:
The Player is rewarded for being consistent and watering the plants in their optimal green zone while avoiding leaving the plants too dry (neutral zone) or over-watered (red zone). Thus, the main criteria behind scoring the player's performance is the time spent on each of these zones. For each plant the total amount of time spent on each zone is summated and normalized to produce a weighted percentage. This information is presented to the Player at the end of the session in the form of a pie chart (see figure xxx). Finally, the final score is calculated as the average of all the weighted averages from all the plants. More detailed score workings and examples can be found in Figure 17 and on this spreadsheet[10].

	Example time in zones				Normalized Time in zones				Tunable values			Weighted scores			Scores	
Example	Time in Grey Zone	Time in Green Zone	Time in Red Zone	Total time	Normalized Time in Grey zone	Normalized Time in Green zone	Normalized Time in Red zone	Sanity check	Weighted score multiplier for Grey zone	Weighted score multiplier for Green zone	Weighted score multiplier for Red zone	Weighted score for Grey zone	Weighted score for Green zone	Weighted score for Red zone	Total score base 100	Normalized Score
1	5	20	0	25	0.20	0.80	0.00	1.00	0.60	1.20	0.75	0.12	0.96	0.00	108	100
2	10	40	5	55	0.18	0.73	0.09	1.00	0.60	1.20	0.75	0.11	0.87	0.07	105	100
3	15	40	10	65	0.23	0.62	0.15	1.00	0.60	1.20	0.75	0.14	0.74	0.12	99	99
4	20	20	15	55	0.36	0.36	0.27	1.00	0.60	1.20	0.75	0.22	0.44	0.20	86	86
5	25	25	20	70	0.36	0.36	0.29	1.00	0.60	1.20	0.75	0.21	0.43	0.21	86	86
6	30	30	25	85	0.35	0.35	0.29	1.00	0.60	1.20	0.75	0.21	0.42	0.22	86	86
7	25	30	30	85	0.29	0.35	0.35	1.00	0.60	1.20	0.75	0.18	0.42	0.26	86	86
8	20	30	25	75	0.27	0.40	0.33	1.00	0.60	1.20	0.75	0.16	0.48	0.25	89	89

Note:	
	A plant needs about 180 seconds in the green zone to grow
	A plant needs about 35 seconds in the green zone to grow
	A plant needs about 90 seconds in the green zone to grow

Figure 17 - Workings and samples of typical game play scenarios.

Figure 18 - Start watering the plant

[10] Link: https://tinyurl.com/gardenMasterExamples

Figure 19 - Watch the meter goes to ‘Green’

Figure 20 - End of session results view

Gameplay footage is available for Pro Mode[11] and the Tutorial[12].

4.3.4. Steady Hands

Set in the Kitchen environment, this mini-game is set to challenge the Player’s dexterity and cognitive skills. The game revolves around having the Player handle a slippery, medium-sized plate in an attempt to balance edible objects of different shapes (see Figure 21 and Figure 22) before finally tilting the plate one way or another such that the object falls into the appropriate category container.

When the Player positions the plate in a predetermined location an edible object will spawn some distance above and land on the plate, he/she is holding. Initially objects are ‘blacked out’ such that only their silhouette can be inferred. However, upon touching the plate the edible

[11] Link: https://tinyurl.com/gardenMasterProGameplay

[12] Link: https://tinyurl.com/gardenMasterTutorial

object starts taking colour such that after a matter of few seconds its identity should be able to be determined easily. The remaining action for the Player then is to tilt the plate left or right such that the item in question falls into its appropriately labelled container; for example: Fruits versus Nuts. Items that land outside the correct containers count as a miss. The cycle repeats itself with the Player being required to reposition the plate to a predetermined position so the next item can be spawned.

Each session contains a set of edible objects of which they fall into exactly either of two categories. The session ends when the last edible object in the sequence has been spawned and has either landed in one of the containers or on the counter.

The game can be configured in a number of ways:

1. Number of sessions per week
2. Number of objects per session
3. Number of groups per session (default min: 2)
4. Time to completely reveal edible item
5. Difficulty (Sensitivity, Friction, Size of plate)

The following data can be stored and visualised on the VRHAB-IT platform as seen in Figure 23

1. Time taken for each session
2. Hit vs miss items sorted
3. Overall score

Figure 21 – Balancing a tomato (round shape object – hard mode)

Figure 22 - Balancing a banana (Easy mode)

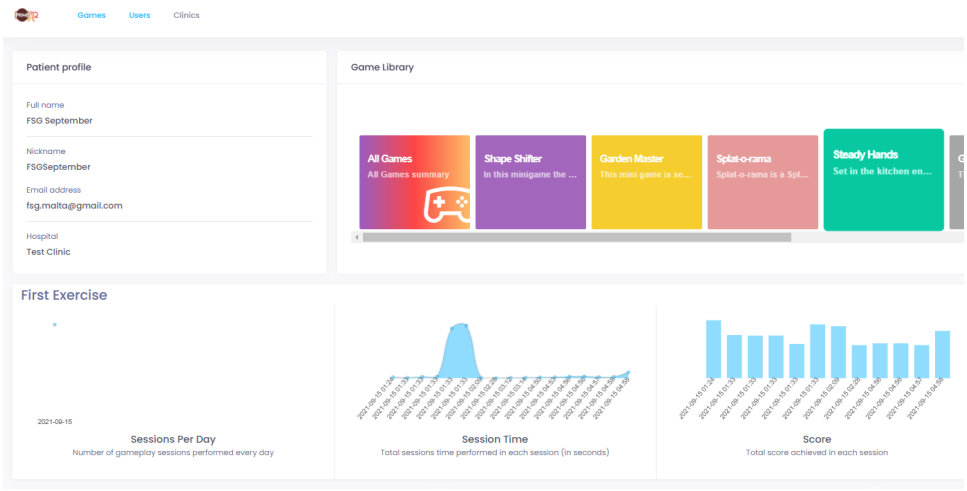

Figure 23 - Sample player data as seen on the VRHAB-IT platform.

4.4. Games under development

4.4.1. Mini Golf

The premise of this game within the Sports Arena (STJH) is precisely what its real counterpart entails: a relaxing outing across a compact, yet challenging, set of mini-golf courses practice session. The objective is simple: to 'hole' the golf ball using as few swings as possible. For a total of 9 courses, each course grows progressively more complex than the previous one, featuring slopes, sand traps and other staples of this sport.

In keeping the gameplay as simple in scope as possible, the Player can only swing one type of club: the putter, and can swing for an unlimited number of tries. To accommodate Players of different physical abilities, swinging the putter can be assigned to the shoulder movement or elbow movement.

Because the courses are relatively small (see Figure 26), the amount of force needed to hit the ball is still realistic (see Figure 24 and Figure 25) when factoring non-traditional input methods such as elbows and wrists. This means that the therapeutic focus of this game is to exercise wrist flexion and extension in order to get the precise power need for each shot.

The difficulty for this game varies the par value for each course, with higher difficulties lowering the par such that fewer shots are allowed to put the ball before the Player goes over par. The score takes into account all shots performed and determines the overall grade achieved.

Figure 24 – Precision shot

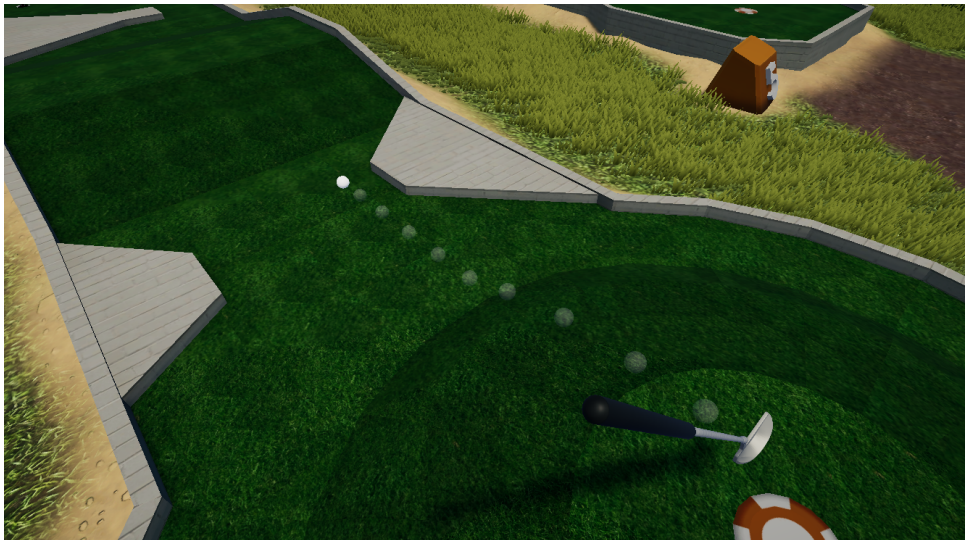

Figure 25 – Direction of shot

Figure 26 - Sample gameplay from a few of the 9 available courses.

4.4.2. Basketball

A real-life single-player practice scenario is being built in the Basketball court for STJH (see Figure 27). This game is established within the Sports Arena and is meant to exercise the shoulder and wrist using the bespoke controller with the wrist module attached.

Basketball balls are picked up from different spots around the court (see Figure 28 and Figure 29) and the action for the player is to use the shoulder/elbow combo to aim in the right direction while extending and flexing the wrist to get that power control according to the distance from target. In the easier levels, a parabola line will be shown to guide the player on the power and aim requirement as an aid. In harder levels, the aim guide is removed, and the balls are further away from the target hoop. The player may try as many attempts as possible to make the basket and may also move to other throwing spots and re-visit a tricky one at a later time.

The number of attempts as well as the accuracy for each shot are considered when scoring the final result for the session.

Figure 27 – Basketball court in Sports Arena

Figure 28 – Far left shot

Figure 29 – Under the hoop shot

4.4.3. Petanque

The last mini-game planned for STJH within the Sports Arena is Petanque (see Figure 30). In this game the focus is once again on wrist movement combined with gross motor movement of the shoulder and elbow to get the general aim. This game is played using the bespoke controller with the wrist module attached.

General aiming direction is set through the shoulder and elbow while the power control is set according to the wrist extension amount and the resistance set on the controller itself. A simple AI NPC will play along this game to keep the session interesting until the best player wins. The number of mistakes committed by the AI may be fine-tuned according to the difficulty level set by the therapist.

The resistance band on the bespoke controller should also be set to match the weight of a petanque ball which could be anywhere between 650 and 800g. This will help with strengthening through resistance while giving the game a more accurate real-life experience.

Figure 30 - Petanque

4.4.4. Rhythm in the kitchen

Reference has been made to popular rhythm games like Beat Saber[13] by GDIH and how these games are not playable by patients with Dystonia even through the use of assistive controllers. The concept here builds on top of the games referenced below where players will be asked to use their HMD to 'look' at objects around the room and perform simple quick time events (QTEs) using very gross motor movement to score better in the game. The 3 referenced games below in Figure 31, use this gesture mechanic to make progress in the game.

These actions may include:

1. Reach the ceiling
2. Touch the floor
3. Stretch Left
4. Stretch Right
5. Draw gestures in the air

Another variation (harder level) of the game will ask the players to perform the above actions but toward objects in the room, therefore the player must first locate this item, then perform the QTE at the right direction.

An MVP of this game will be developed and tested with dystonic patients because as the living labs duly noted, none of this has been tested before, so literature or other documented studies are not available. The plan is to test this out with clinicians first, then followed by real patients to gather feedback through WP7 and adjust accordingly.

[13] https://store.steampowered.com/app/620980/Beat_Saber/

Castlevania: Dawn of Sorrow (Konami/Nintendo DS, 2005) - asks the player to draw a particular shape with the stylus to defeat milestone bosses.

Darklings (MildMania/iOS, 2013) - relies on the player making gestures on the touch screen to strategically defeat the enemy monsters around the player's avatar.

Super Mario Party (Nintendo/Switch, 2019) - "Looking for love" asks the player to use the joystick to look at the given shape in any location up/down/left/right.

Figure 31 - Reference games with similar quick time events.

5 WEB PLATFORM

5.1. Introduction

The PRIME-VR2 Web Portal is a web application that coordinates the activity of hospital supervisors, doctors, and patients to achieve successful VR therapy outcomes.

The system consists of two main functional units:

- **Web portal:** A human-friendly user interface where users can interact with the system.
- **Gameplay API:** A machine-friendly API that games can use to query configuration options and upload gameplay data.

5.2. SELECTED TECHNOLOGY STACK

As stated in deliverable D6.1 KRL considered different languages and technologies to create a robust, future proof back-end and front-end for the PrimeVR2 Web Platform.

The technology requirements for the Web Platform:

1. Meets the complex requirements of Prime-VR2 and gives a great user experience
2. Scalable, when it comes to commercialisation the tech stack could handle hundreds, thousands, or tens of thousands of users. Further development, optimisation and tweaking will be needed, but the tech stack carries the possibility to scale up if needed.
3. Popular, open-source technologies, that makes it easier to hire developers for future improvements
4. Future proof, the underlying technology will be maintained and developed in the upcoming years
5. Easy to maintain, cost effective to create updates, new features.

5.2.1. Runtime Environment: NODEJS

Node.js is an open-source, cross-platform, back-end JavaScript runtime environment that runs on the V8 engine and executes JavaScript code outside a web browser. Node.js allows developers use JavaScript to write command line tools and for server-side scripting—running scripts server-side to produce dynamic web page content before the page is sent to the user's web browser. Consequently, Node.js represents a "JavaScript everywhere" paradigm, unifying web-application development around a single programming language, rather than different languages for server-side and client-side scripts.

Programming Language: TYPESCRIPT

TypeScript is a “strongly typed” programming language which builds on JavaScript giving you better tooling at any scale.

JavaScript (also known as ECMAScript) started its life as a simple scripting language for browsers. At the time it was invented, it was expected to be used for short snippets of code embedded in a web page — writing more than a few dozen lines of code would have been somewhat unusual. Due to this, early web browsers executed such code pretty slowly. Over time, though, JS became more and more popular, and web developers started using it to create interactive experiences.

Web browser developers responded to this increased JS usage by optimising their execution engines (dynamic compilation) and extending what could be done with it (adding APIs), which in turn made web developers use it even more. On modern websites, average browsers are frequently running applications that span hundreds of thousands of lines of code. This is long and gradual growth of “the web”, starting as a simple network of static pages, and evolving into a platform for rich applications of all kinds.

More than this, JS has become popular enough to be used outside the context of browsers, such as implementing JS servers using node.js. The “run anywhere” nature of JS makes it an attractive choice for cross-platform development. There are many developers these days that use only JavaScript to program their entire stack.

5.2.2. Package Manager: NPM

NPM is one of the world's largest software registries. Open-source developers use npm to share and borrow packages, and many organisations use npm to manage private development as well.

5.2.3. Framework: ANGULARJS

Angular is a development platform, built on TypeScript. As a platform, Angular includes:

- A component-based framework for building scalable web applications
- A collection of well-integrated libraries that cover a wide variety of features, including routing, forms management, client-server communication, and more
- A suite of developer tools to help you develop, build, test, and update your code

With Angular, one takes the advantage of a platform that can scale from single-developer projects to enterprise-level applications. Angular is designed to make updating as straightforward as possible, so it exploits the latest developments with a minimum of effort.

Above all, the Angular ecosystem consists of a diverse group of over 1.7 million developers, library authors, and content creators.

5.2.4. Asset Bundling: WEBPACK

Webpack is a static module bundler for JavaScript applications. This enables the developers to take a fully dynamic application and package it into static files, which can be uploaded and deployed to the server.

The webpack gathers all the application's dependencies, which includes not just code, but other assets as well, and generate a dependency graph. By a bundler like webpack one can make sure that every software component is prepacked and shipped within the bundle.

5.2.5. Containerization: DOCKER

Docker is an open platform for developing, shipping, and running applications. Docker enables developers and DevOps to separate the applications from the infrastructure so they can deliver software quickly. With Docker, DevOps can manage your infrastructure in the same ways they manage the applications. By taking advantage of Docker's methodologies for shipping, testing, and deploying code quickly, one can significantly reduce the delay between writing code and running it in production.

5.3. SOFTWARE COMPONENTS

The D6.1 describes the PRIME-VR2 Web Portal as a web application that coordinates the activity of hospital supervisors, doctors, and patients in order to achieve successful VR therapy outcomes. Two main functional units was defined: Backend / API and Frontend (Website)

During the implementation phase new components become individual parts of the solution, such as Data Transfer Objects (DTO), Auth Service and the File Service. KRL decided to split the functionalities into micro services. Each service contains only a subset of features based on its scope and role in the project. For example, every functionality that is tied to authentication and authorization was implemented in the Auth Service. Everything about file upload, storage management was implemented in the File Service. In broad these are parts of the backend and API but implemented as a separate application, to make it easier to maintain and separate the responsibilities.

5.3.1. Web Platform Backend and API

The Backend and API implements the specification of the following sections that are described in the deliverable D6.1.

- 5.5 Hospital Management
    - Hospital List
    - Create Hospital
    - Manage Hospital Users
    - Edit Hospital

- 5.6 Games
    - Games List
    - Create Game
    - Edit Game
    - Disable/Enable Game Globally

- 5.7 Patient Overview
    - List of Games
    - Goal Review
    - Activity Feed
    - Game Definition
    - Create Session
    - Delete Session
    - Edit Session
    - Leaderboard

The Backend and API does not implement the 5.8 Messaging, 5.9 Notifications due to change in the project requirements. During the iterations the Living Labs did not require these proposed features anymore. On the other hand, a more detailed Achievement and Goals system resulted as a must-have requirement.

Technology

- **NodeJS:** server-side JavaScript engine
- **TypeScript:** statically typed programming language based on JavaScript
- **PostgreSQL:** relational database engine
- **Knex:** SQL query builder library, Knex is a JavaScript library that helps programmers write correct and secure SQL queries by exposing an API that prevents common mistakes.
- **AJV (Another JSON Validator):** JSON schema validation, AJV is the most popular JSON Schema validator in the Node ecosystem. The validation performance is great (quicker than xml schema or other json approaches), well-supported, and its development team responds quickly to bug reports and security issues.
- **Mailjet:** email service
- **Superagent:** HTTP client

Code Structure

For detailed information please check the source code of the project.

- **package.json**: npm dependencies and scripts
- **package-lock.json**: locked npm package versions and checksums
- **tsconfig.json**: TypeScript compilation settings
- **knexfile.js**: Knex database access options
- **Dockerfile**: commands and settings for building a Docker image of the project
- **index.ts**: application entry point
- **src**/: TypeScript source code
  - **config**/: code that supports reading project configuration
  - **controllers**/: code that defines public API endpoints
  - **errors**/: custom error entities
  - **helpers**/: common utility functions
  - **migrations**/: database version control code
  - **models**/: data entities
  - **seeds**/: database initialization
  - **services**/: common backend code used by controllers
  - **constants.ts**: common constants used across the project

5.3.2. Web Platform Frontend

Technology

- **Angular:** Single page application framework
- **Metronic Framework:** theme and style framework for Angular, Metronic is a flexible and modern theme for the Angular framework. Its flexibility allows customizing the look and feel of a project to a very high degree.
- **Chart.js:** chart library, Chart.js is the most popular JavaScript chart library. The charts it provides are user and developer friendly and highly performant.
- **Lodash:** utility library

- **Moment:** date library
- **RxJS:** functional reactive utility library, Lodash, Moment and RxJS are industry standard utility libraries used across the frontend and Node ecosystem.

Code Structure

- **angular.json**: Angular configuration
- **Dockerfile**: commands and settings for building a Docker image of the project
- **package.json**: npm dependencies and scripts
- **package-lock.json**: locked npm package versions and checksums
- **tsconfig.json**: TypeScript compilation settings
- **tslint.json**: TypeScript code quality configuration
- **webpack.config.js**: deployment configuration
- **src**/:
  - **app**/: application source code
    - **core**/: common reusable components
    - **views**/: page components and their supporting code
  - **assets**/: static files (images, styles, etc.)
  - **environments**/: configuration options for different execution environments
  - **lib**/: vendored-in libraries

5.3.3. DTO (Data Transfer Objects)

Technology

- **NodeJS:** server-side JavaScript engine
- **TypeScript:** statically typed programming language based on JavaScript

Code Structure

- **package.json**: npm dependencies and scripts
- **package-lock.json**: locked npm package versions and checksums
- **tsconfig.json**: TypeScript compilation settings
- **tslint.json**: TypeScript code quality configuration
- **src**/: source code of the DTO definitions

5.4. CLOUD INFRASTRUCTURE & HOSTING

When an IT infrastructure is needed to be set up for a modern web application, there are three possibilities:

- **Traditional On-premises**
- **IaaS**: Infrastructure as a Service
- **PaaS**: Platform as a Service

The **traditional On-premises** have the advantage that all the control of the infrastructure is given locally. In high-security environments this is a must-have.
On the other hand, it means tremendous upfront cost and resources: buying the hardware, installing operation software, and running the final application. In this case, the owner is in charge of maintaining everything, this means regular hardware updates, OS and lib patching, etc.

Besides that, to provide a 24/7 service, there is a need to create a secure environment with a secure internet connection, which requires a professional server room with temperature and climate control, failover electrical and communication solutions in place.

Due to the above reasons, this option was not chosen to run the Web Platform components. The number of costs and efforts (setup and maintenance) in an on-premises environment are not balanced with the goals of the project.

The second option is called **Infrastructure as a Service (IaaS)**. Most cloud providers give these by default. Servers, storage, networking, and virtualisation layers are managed by the cloud vendor. The OS, middleware, runtime, data, and the application layers are managed by the client. This is a good balance if an application needs a special environment to run within. Although, the upfront investment into setting up everything and maintaining throughout the project lifetime is still considered too high for the Prime-VR2 project.

The third option is the **Platform as a Service (PaaS)**. It is still a cloud solution like the IaaS. But all infrastructure layers (Networking, Storage, Servers, Virtualization, OS, Middleware, Runtime) are managed by the cloud vendor. That means the provider take care of all the hardware and software level updates and maintenance (in practice they are buying hardware components like CPU and memory, HDD, manages changes if something is corrupted without any outage. This also applies to OS and software level. Operation team at the Cloud provider updates and manages the software level also). This option gives the fastest time to value and makes it possible to focus more on the data and application layers than the infrastructure itself.

When the PRIME-VR2 project started, it was decided to go with IaaS within Google Cloud (EU datacenter). That means KRL created and set up a VPS environment to run the application containers and the database.

During the implementation phase Google updated their services and introduced a solution called “Cloud Run”. Cloud Run service makes it possible to develop and deploy highly scalable containerized applications (applications that are packaged in a container for running on the server) on a fully managed serverless platform. This seemed to be a much better option,

because this is a PaaS solution, so the developer can focus on the application and data layers while the underlying infrastructures are managed and giving a better value to time ratio.

It was decided to change the environment and created a new setup and ported the app to this new environment (Figure 32).

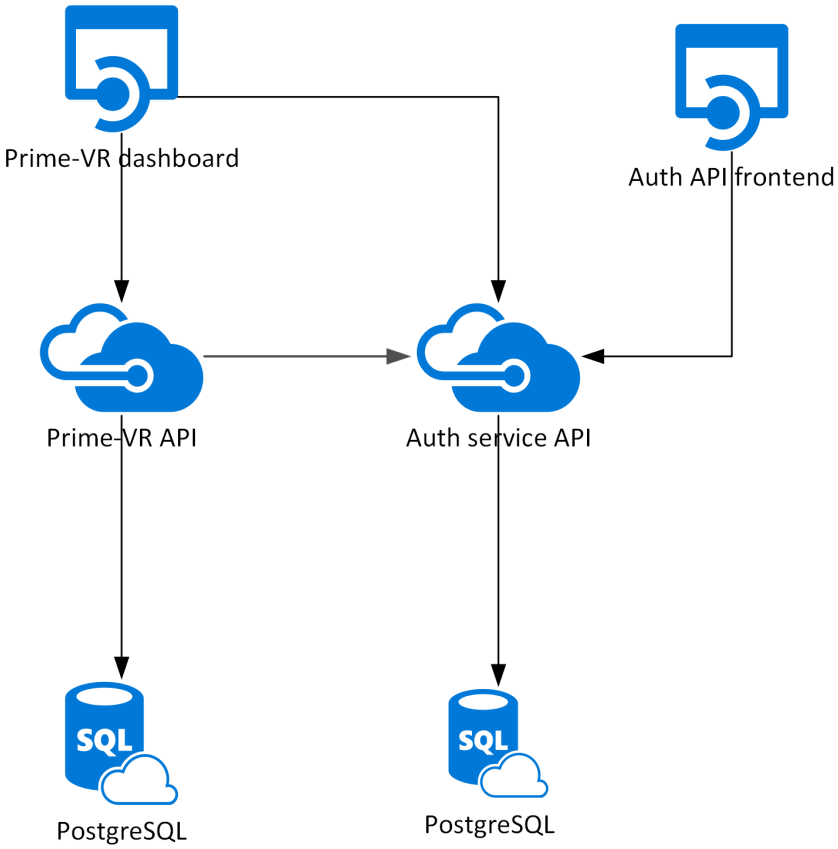

Figure 32 – Web platform architecture chart

5.4.1. Database

PostgreSQL is a powerful, open-source object-relational database system with over 30 years of active development that has earned it a strong reputation for reliability, feature robustness, and performance. Google offers this as a managed, scalable CloudSQL service. This means all the underlying infrastructure and the security patches automatically keeps updated.

PostgreSQL is widely used, because of its data integrity, extensibility, and the dedication of the open-source community behind the software to consistently deliver performant and innovative solutions. PostgreSQL runs on all major operating systems, has been ACID-compliant since 2001. PostgreSQL comes with many features aimed to help developers build applications, administrators to protect data integrity and build fault-tolerant environments, and help you manage your data no matter how big or small the dataset. In addition to being free and open source, PostgreSQL is highly extensible. For example, one can define his/her own data types, build out custom functions, even write code from different programming languages without recompiling a particular database.

5.4.2. Storage, Google Storage

Cloud Storage is a service for storing blob objects in Google Cloud. An object is an immutable piece of data consisting of a file of any format. Objects are stored in containers called buckets.

5.4.3. Containerisation

Today's modern web applications are using containerization. A container is a standard unit of software that packages up code and all its dependencies, so the application runs quickly and reliably from one computing environment to another. That solves the long run problem of different runtimes on dev, staging and production environments that leads to slow and unreliable deploy processes and hard times with tracking down different bugs. By leveraging container technology, we can make sure that the same environment and runtime are consistent through the deploy process and no unexpected errors will happen.

There are several options for containers, like Docker, Rocket or LXC. KRL decided to go with Docker technology, because this is one of the most common and well supported also by Google Cloud.

All the components of the web application are containerized and run by Cloud Run service.

5.4.4. Networking & Security

KRL decided to use Google Clouds VPC service because it was deemed suitable when dealing with health-related data security systems.

VPC is a managed network functionality for the dedicated Google Cloud account. It makes it possible to create a secure network communication environment within the account.

5.4.5. Cloudflare

Furthermore, it was decided to add a second layer of security for the networking by introducing Cloudflare into the mix. Cloudflare is a global network designed to make everything one connects to the Internet secure, private, fast, and reliable. Secures websites, APIs, and Internet applications. Protect corporate networks, employees, and devices. Cloudflare provides a secure tunnel between the PrimeVR2 application components and the end user, while constantly examining the transferred data packets looking for threats and animalities. It provides an advanced firewall before the application and its runtime environment.

5.4.6. Scalability

If cloud technologies are used well, creating a scalable solution is become relatively easy. Cloudflare, Google Cloud Run and Store makes it able to PrimeVR2 web platform handle small or large number of users and traffic. The service decisions that were described above makes the selected solutions a future proof one.

5.5. DEPLOY PROCESS

KRL created a seamless process to maintain fast deployments to support iterations and quick provides quick fixes. However, this does not mean, it implements a full Continuous Deployment (CD) at the moment, but it is possible to develop it further when project reaches its commercialisation development phase.

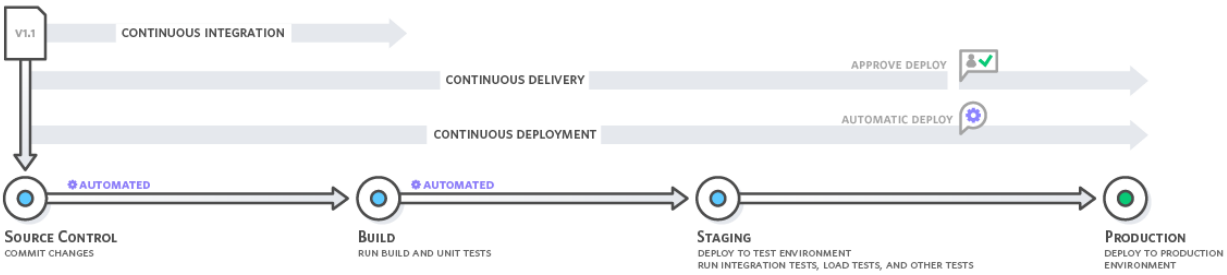

Figure 33 – Deployment process

The current stages of the Deploy Process:

1. Developers push their code to the Version Control System (GIT)
2. Pull Request created, Code Review needed from a peer developer
3. Fix the code if needed
4. Approval of the Pull Request
5. Approved code merges into the Version Control System
6. Automatic Cloud Build starts on a specific branch
7. If the Build succeeded an Artifact created on the Cloud Storage / Container Registry
8. Cloud Run can use this Container Registry and the Artifact to upgrade to the last version without an outage.

During commercialisation more steps could be considered, such as automated unit testing or smoke, integration, end-to-end and UI tests.

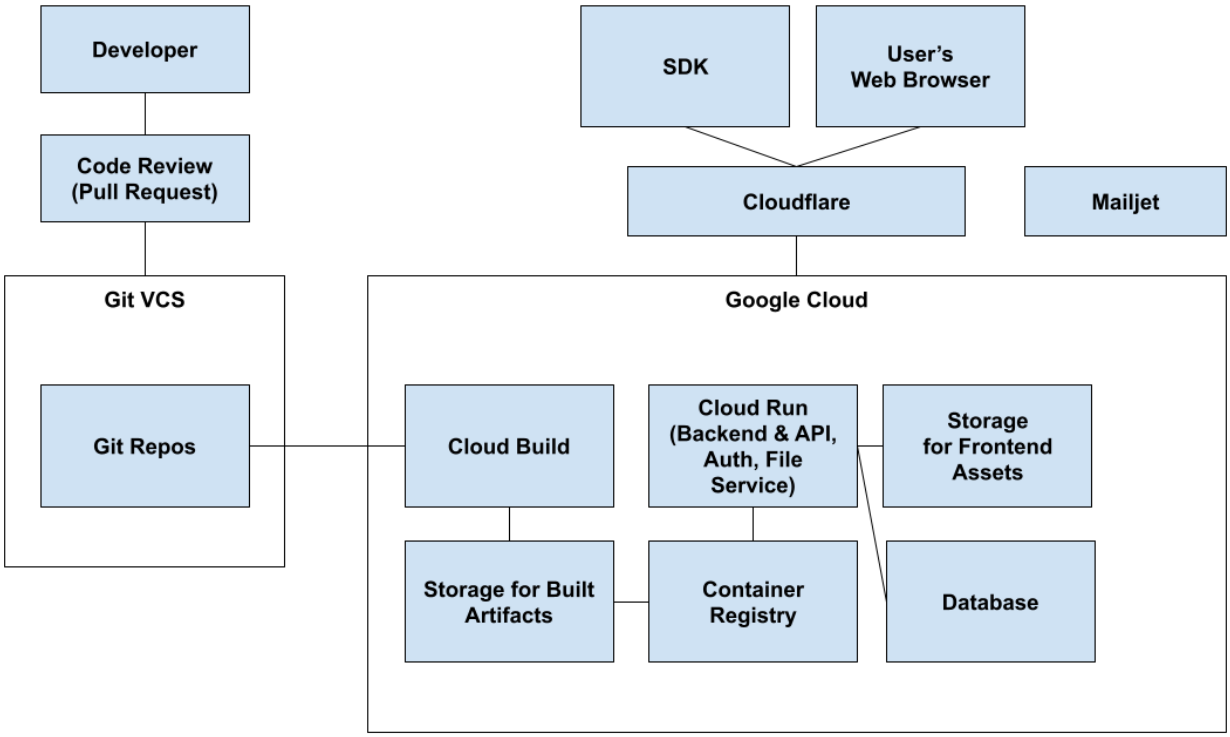

Figure 34 – Deployment process (detail)

6 PERFORMANCE METRICS

WP6 recommended the initial KPIs to be used for games in the VRHAB-IT. The four-performance metrics that will be used to benchmark each game developed during this project will also be used to judge the quality of the games submitted by third-party developers on VR-HABIT platform. These are:

1. **Retention rate** – in this metric we look for how long patients keep coming back to the platform. There could be many reasons why patients walk-away from the platform so this is the top-most important metric to look for on a daily basis. It is defined as a percentage of all patients who come back after Day-1, Day-3, Day-7, Day-14 and Day-30.
2. **Number of daily sessions** – Building on the retention rate, in this metric, we also look for the number of sessions each player played per day. It must be pointed out that the target for this metric might be different for each user for each different game and is hence set by their therapist. This value could also be less than 1 if the prescription defines less than 1 session per day (e.g. 3 sessions per week)
3. **Session duration** - a session is the time from when the player starts playing to the moment when the activity is suspended. This metric will give us a better understanding of how players are doing when compared to traditional therapy methods and if the games are interesting enough to keep the players engaged for a longer period of time.
4. **Engagement score** - the engagement score is another way of measuring the user satisfaction. In this metric we will try and see what the players are doing while playing the game. A number of issues must be addressed; for instance, will the patients be using the most basic feature only, will they be exploring around menus or environments to look for additional content and do they replay the same game over and over again to get a better score? We define this metric in terms of areas traversed with points given to each one according to the difficulty of finding that particular place and summing the values up.

7 VALIDATION CRITERIA

7.1. Outline

The purpose of the validation criteria is to allow the development of appropriate mechanisms to assess the VR Platform and games. These mechanisms ensure that the evolution of the controller designs and gameplay are fulfilling the goals of allowing functional interaction control for the different patient groups. Initially, these were derived from the user requirements defined in deliverables D2.1 (VR Platform Requirements) and D7.1 (Persona Report). In particular, the VR Games section of this D2.1 outlines, with reference to the game environments proposed in Task 6.4, how these requirements can be supported within the constraints of available VR systems and proposed controller designs.

7.2. Re-framing of evaluation to include standard controllers

The process of validation is designed to complement both the activities of Task 7.4 (VR environment testing and feedback), and the development of the experiment protocols which

define how the VR-HABIT system will be tested. Initially this was proposed to proceed iteratively, on an approximate 3-month cycle of feedback, analysis and redesign. However, delays related to COVID-19 have severely curtailed access to patients and thus initial data is lacking. This lack of data has also hampered the development of experiment protocols that are required for some of the project's ethics applications. In particular, testing in the UK involving non-CE marked devices requires additional regulatory approval. To mitigate the added delay of this process, we have re-organised the phases of work to introduce an initial phase using standard VR controllers and games, with subsequent phases incorporating the VRHAB-IT controllers and bespoke games. This separation of ethics approvals has allowed data-gathering activities to proceed as early as COVID-19 restrictions allow. For initial studies, the focus is on patient experience and system usability (in terms of both physical controllers and task mechanics). In addition to user comfort and enjoyment, we are using variants of standard questionnaires to assess each user's sense of presence and embodiment (deliverable D2.1 provides further detail on these phenomena). This data will be fed back into refinement of environment designs and gameplay mechanisms.

7.3. User assistance mechanisms

User assistance mechanisms can be thought of as any method applied to gameplay that helps the user to complete a task that they might otherwise struggle to perform. We have divided these mechanisms into two categories as follows.

7.3.1. Automated through the control interface

A specific mechanism of interest here is manipulation of user embodiment. In this case the game will identify the user's goal and take partial control of the user inputs to facilitate the completion of that goal. This works in VR because the user does not see the real movements of their own hands, but instead sees a virtual proxy (or avatar) within the game, and up to a point will not notice small differences between their proprioceptive sense of how their upper limbs are moving and the visual representation of their limbs in the virtual environment. For example, this mechanism can be applied to filtering out tremors from user movement, or to extend the user's range of motion. Further details of this phenomenon are described in D2.1. While the gameplay design can allow various degrees of such assistance to be applied, the intention currently is that the level of assistance will be set under therapist supervision (i.e. the therapist will monitor task performance and decide what degree of assistance is appropriate). It will not be under the control of machine-learning mechanisms within the game, since the control problem to be resolved here is complex and nuanced, and so requires expert input.

7.3.2. Human-in-the-loop

We have continued to research this type of mechanism through a series of experiments on "co-pilot" user assistance. Experiments have been conducted under an existing blanket ethics case which has allowed experiments involving VR equipment with consenting adults as participants. The initial focus of this work was the development of a prototype to allow split control of arbitrary VR games across multiple input devices. With the developed prototype, a second input device is used to manipulate the player's position in the virtual 3D space. This second device enables motion impaired players to play VR applications that require full body

movement. Additionally, an identified use case is for the second input device to be operated by a second player, turning the game into a cooperative task.

To validate the prototype, we conducted a pilot study with eight participants. For the experiment, the participants played a standard VR game “Space Pirate Trainer” in pairs with the experimenter. In one role the participant simulated the motion impaired player and had to play seated while in the other role they controlled the first player’s movement through a regular Xbox Gamepad. The task in the game was to shoot down flying drones while dodging lasers that the drones fire in the direction of the player. A player without motion impairments can perform this through body movement, but in the study, they rely on the second player to move them out of the way of incoming fire. Participants were interviewed after each experience. Factors that were investigated were fun, engagement, communication, responsibility, and ownership of the final score. In general feedback was positive, while ownership of achievements within the game was not straightforward.

A subsequent experiment explored the co-pilot controlling aspects of the player’s hand movement, for example by allowing the co-pilot to extend the reach of the player beyond their usual range to reach distant or high objects in a game, while the player retained control of their hand and its actions. In this study seated healthy adult players, paired with a co-pilot and playing a simulated office environment game, reported that enjoyment was maintained while assistance was provided, and that they were able to mitigate some of the limitations of playing sitting down.

Different aspects of the co-piloting work have been presented at the XR Accessibility Workshop at ACM IMX 2020; IEEE VR 2021 Workshop on Novel Input Devices and Interaction Techniques (NIDIT); and the Institute of Physics and Engineering in Medicine (IPEM) Emerging Technologies.

7.4. Development of Experimental Protocols

Validation of the PRIME-VR2 games ultimately requires data fed back from user trials. In Section 7.2 we have described how the overarching development of the experimental framework has been redesigned to allow data to be captured as early as possible in the wake of delaying factors. These factors have been due to both COVID-19 restrictions and supply-chain issues in electronics restricting the availability of the selected VR system (Valve Index) throughout Europe. All three living labs now have working systems, and both St James Hospital (Malta) and KNRC (Cyprus) have obtained ethical approval for initial testing. Arrangements are currently underway to send a researcher from UCL to Cyprus and Malta to oversee trials involving co-piloting (see Section 9.3 above).

Detail of the derivation and development of experimental protocols for each of the three living labs is described in deliverable D7.2 (Ethical approval and protocol validation). Here this information is summarised to include only those details relevant to the validation of the Prime-VR2 platform, rather than the motivating factors behind each element. At all 3 Living Labs the experimental protocols have been divided into similar phases:

- Physical and spatial measurements from participants to guide the design of Prime-VR’s bespoke physical controllers.

- Measurement and evaluation of off-the-shelf systems
- Measurements and evaluation of bespoke Prime-VR games

It is the last of these elements that is of particular interest here, since this is what relates to the integrated prototype described in this document. While the patient categories at each of the living labs are very different, the main research question of the study at each lab is similar, in that it involves the evaluation of the feasibility and efficacy of VR as an intervention for each class of patient. Following from this, a larger number of subsidiary questions arise, and the system design and evaluation cycle must address each of these. Examples of metrics that were considered include:

- Accessibility and usability – are patients able to use the platform easily, and perform the tasks within each game effectively?
- Retention rate – how often, and for how long, do patients come back to use the platform?
- Performance - do patients improve their performance of the VR tasks and games over time, and is this associated with increased levels of functioning in real-world tasks?
- Adverse side-effects – are there any negatively associated effects of using the system. This might be dizziness or nausea induced by the VR games, or frustration with the system if it is too hard to use.

A protocol for off-the-shelf systems has been developed (for the UK this is still currently under review). A subsequent protocol for the bespoke system is under development, but subject to modification based on initial data. Data capture within this protocol includes:

- Recruitment rate
- Accessibility
- Length and number of the sessions
- Incidence of adverse events
- Data acquired by the VR system itself (position, orientation and acceleration tracking from the headset and hand controllers)
- Video recording of both participants and a rendering of their view of the VR environment.
- Technical problems arising with the VR technology will be recorded

Specific data captured is dependent upon whether the VR experience is with or without co-piloting. For VR experiences without co-piloting, standardised questionnaires are used to assess the following:

- **Simulator-sickness**: the virtual environments and interaction in this study are designed to avoid the types of visual stimulus that can lead to simulator sickness (e.g., vection, visual-vestibular mismatch). For this reason, we are not applying the full Simulator-sickness questionnaire (SSQ) (Kennedy et al., 1993) that is commonly used, but will instead embed a single question (within the presence questionnaire) to capture simulator sickness symptoms.

- **Presence:** to capture user "presence" (the degree to which a person in VR responds to interactions and events in the simulated VR world as if they are real) we use a questionnaire derived from (Slater, Usoh and Steed, 1994), with additional questions that are specific to the virtual environments and gameplay. It has become standard practice to investigate presence in this way, and several variants of presence questionnaires are commonly used.

- **Embodiment:** related to the notion of presence, is the user's sense of embodiment i.e., their sense of agency and control over their virtual body. Recent research efforts have made progress toward validated questions relating to body ownership, location, agency, and motor control (Peck and Gonzalez-Franco, 2021) and our questionnaire is based on this, though restricting questions to those relevant to the nature of our study.

VR experiences involving co-piloting use the following data for assessment:

- The number of activations of the co-pilot
- Screen-capture and tracking information
- The user's experience with the co-piloting feature and their human co-pilot will be assessed with two short, standardised questionnaires after the session:

    1. The patient's experience with co-piloting will be captured using the UEQ-S (Schrepp, Hinderks and Thomaschewski, 2017). It is a shortened version of the User Experience Questionnaire, a standardised questionnaire to measure user experience.
    2. To capture the patient's experience with the co-pilot, a shortened version of the Networked Minds Measure of Social Presence (NMMoSP) (Harms and Biocca, 2004) will be used. It is a standardised questionnaire designed to capture how a user perceives social presence, i.e., how they feel about another person that shares the virtual environment with them. While the co-pilot does not have a visual representation in the virtual world, they still share it with the patient through their inputs. Due to this lack of representation, not all items of the questionnaire are applicable, and a shortened version will be used.

Following the VR session, a semi-structured interview will be undertaken within one week. This will capture the patient's views on the system: its comfort, accessibility, usefulness, and potential ways for improvement. The semi-structured interview will also be used to follow up on responses and effects that the researchers might have observed during sessions.

7.5. Evaluation of PRIME-VR2 Platform Interface

The PRIME-VR2 platform interface was designed and implemented to support the user requirements captured through project deliverables, focus groups and meetings with living labs project partners. An initial assessment of the platform mostly sought to address technical issues with the platform interface. A second and more detailed assessment was performed as a guided walkthrough with a member of the clinical team from GDIH. This assessment focused on 2 classes of user: therapist and patient, since these are the principal users of the system. The evaluation was based on the cognitive walkthrough method (Polson et al, 1992). This is a technique developed for evaluation of user interfaces, grounded in theories of

exploratory learning. In other words, it assesses how easily new users can intuit the actions to perform desired tasks. In the main, the interface was found to be simple and intuitive to use, following specifications derived in earlier design work. However, several issues were raised and these are currently being addressed. A further iteration will take place once the platform is being used by patient populations. At this stage it is expected that only minor changes and bug fixes will be required.

8 CONCLUSION

The Integrated Prototype Demonstration presents all achievements so far, the regarding details about the Web Portal, PrimeVR2 Unity SDK, and the VR Ecosystem (Rehabilitation Games). It also defined the terminology and coding standards to align the work between WP6 members.

The Web Portal, PrimeVR2 Unity SDK, and the Rehabilitation Games create an integrated environment. Within this ecosystem, the Doctors and their Patients can work seamlessly to achieve their rehabilitation goals. Because every data from the games is uploaded into the Web Portal in real-time, the analysis is much easier even remotely.

The system remains extendable in the future with new games and metrics. Later goals can be to extend the system with multiplayer functions or third-party developers.

Keep in mind in the implementation phase there can be slight changes in the execution. This is mostly related to late information from other Work Packages or findings that were not previously known.

9 REFERENCES

Harms, C. & Biocca, A.F. (2004). Internal consistency and reliability of the networked minds social presence measure. *Seventh Annual International Workshop: Presence 2004*.

Kennedy, R. S., Lane, N. E., Berbaum, K. S., & Lilienthal, M. G. (1993). Simulator Sickness Questionnaire: An Enhanced Method for Quantifying Simulator Sickness. T*he International Journal of Aviation Psychology*, 3(3), 203–220. https://doi.org/10.1207/s15327108ijap0303_3

Peck, T. C., & Gonzalez-Franco, M. (2021). Avatar Embodiment. A Standardized Questionnaire. *Frontiers in Virtual Reality*, 1, 575943. https://doi.org/10.3389/frvir.2020.575943

Schrepp, M., Hinderks, A., & Thomaschewski, J. (2017). Construction of a Benchmark for the User Experience Questionnaire (UEQ). *International Journal of Interactive Multimedia and Artificial Intelligence*, 4(4), 40. https://doi.org/10.9781/ijimai.2017.445

Slater, M., Usoh, M., & Steed, A. (1994). Depth of Presence in Virtual Environments. *Presence: Teleoperators and Virtual Environments*, 3(2), 130–144. https://doi.org/10.1162/pres.1994.3.2.130

10 APPENDIX

10.1. PRIME-VR2 Gameplay API

Version: 1.0.1
This is the API definition for the PRIME-VR2 Web Portal project. For more information please ***READ THE SPECIFICATION***.
Schemes: https

10.2. Summary

10.2.1. Tag: gameplay

Gameplay API

Operation	Description
POST /GAMEPLAY/SESSIONS	Create Game Session for Patient
GET /GAMEPLAY/GAMES	List Games (previously "SDK Details")
GET /GAMEPLAY/GAMES/{GAMEID}/PATIENTS/{PATIENTID}/CONFIGURATION	Get Game Configuration for Patient

10.2.2. Tag: file

FileService API

Operation	Description
GET /FILES/{FILEID}	Get a file from file service

10.2.3. Tag: auth

AuthService API

Operation	Description
POST /LOGIN	Return an bearer token

GET /PROFILE/ME	Return the logged in user profile data (by the given token)

10.3. Security

10.3.1. auth_svc

Type: oauth2
Flow:
implicit
AuthorizationUrl:
HTTP://AUTH-API.PRIMEVR.COM/LOGIN
Scopes:
gameplay.config: Get Game config for User
gameplay.session: Create Gameplay Session for User
games.index: List Games

10.4. Paths

Get a file from file service

10.4.1. GET /files/{fileId}

Tags: ***FILE***

fileId	The ID of the file	path	*string*	

200 OK
Success
404 Not Found
File not found
List Games (previously "SDK Details")

10.4.2. GET /gameplay/games

Tags: ***GAMEPLAY***

200 OK
Success
GAMELISTITEM

auth_svc	games.index

Get Game Configuration for Patient

10.4.3. GET /gameplay/games/{gameId}/patients/{patientId}/configuration

Tags: ***GAMEPLAY***

gameId	The ID of the game	path	*integer*	
patientId	The ID of the patient	path	*string*	

200 OK
Success
USERGAMECONFIGURATION
400 Bad Request

Invalid input or game disabled
404 Not Found
Game or Patient not found

auth_svc	gameplay.config

Create Game Session for Patient

10.4.4. POST /gameplay/sessions

Tags: ***GAMEPLAY***
GAMEPLAYSESSIONINPUTMODEL

200 OK
Success
400 Bad Request
Invalid input or game disabled
404 Not Found
Game or Patient not found

auth_svc	gameplay.session

Return an bearer token

10.4.5. POST /login

Tags: ***AUTH***
format: { data: { email: , password: } }'
AUTHLOGINOBJECT

200 OK
Success
401 Unauthorized
Authentication Failed
Return the logged in user profile data (by the given token)

10.4.6. GET /profile/me

Tags: ***AUTH***

200 OK
Success
401 Unauthorized
UnauthorizedError

10.5. Schema definitions

10.5.1. AuthData: *object*

email: *string*
password: *string*

10.5.2. AuthLoginObject: *object*

data: **AUTHDATA**

10.5.3. GameListItem: *object*

id: *integer*
name: *string*

version: *integer*
isEnabled: *boolean*
iconId: *string*
previewImageId: *string*
gameFileId: *string*
introVideoId: *string*
description: *string*
recommendedUses: *string*
recommendedGoals: GOALS

10.5.4. GameplaySessionInputModel: *object*

patientId: *string*
`"74548492-a159-4bfb-a2cf-43ea51a85dfa"`
gameId: *number*
`12`
startedAt: *string*
An ISO-formatted time string
`"2020-08-11T13:35:49Z"`
duration: *number*
The duration of the gameplay session in minutes
`42`
score: *number*
The score achieved during the session
`10078`
achievements: *string[]*
An array of symbolic achievement names unlocked during this session

```
[
    "100-kills",
    "discovered-hidden-room"
]string
```

variables: *object[]*
An array of variable names and values for this session (these will be used to calculate the leaderboard and display charts).
object
name: *string*
The name of the variable
`"kills"`
value: *integer*
The value of the variable
`48`
state: *object*
Custom state object the game can use to calculate achievements from

10.5.5. Goals: *object*

gameplayTimePerDay: *integer*
sessionsPerWeek: *integer*

10.5.6. UserGameConfiguration: *object*

config: *object*
Game configuration object that conforms to the configuration JSON schema specified when creating the game
state: *object*
Custom state object the game can use to calculate achievements from