RealTES : RealTime Environment Simulator for Work/Learning

CS492 Project

MDL18CS032 19 Arya Jaydev K M
MDL18CS050 28 George M Cherian
MDL18CS066 37 Lijo Zechariah James
MDL18CS114 59 S Sandeep Pillai
B. Tech Computer Science & Engineering

Department of Computer Engineering
Model Engineering College
Thrikkakara, Kochi 682021
Phone: +91.484.2575370
http://www.mec.ac.in
hodcs@mec.ac.in

June 2022

Model Engineering College Thrikkakara
Department of Computer Engineering

C E R T I F I C A T E

This is to certify that, this report titled ***RealTES: RealTime Environment Simulator for Work/Learning*** is a bonafide record of the work done by

MDL18CS032 19 Arya Jaydev K M
MDL18CS050 28 George M Cherian
MDL18CS066 37 Lijo Zechariah James
MDL18CS114 59 S Sandeep Pillai

Eighth Semester B. Tech. Computer Science & Engineering

students, for the course work in **CS492 Project**, which is the second part of the two semester project work, under our guidance and supervision, in partial fulfillment of the requirements for the award of the degree, B. Tech. Computer Science & Engineering of **APJ Abdul Kalam Technological University.**.

Guide

Dr. Sindhu L
Assistant Professor
Computer Engineering

Coordinator

Dr. Sindhu L
Assistant Professor
Computer Engineering

Head of the Department

Dr. Preetha Theresa Joy
Professor
Computer Engineering

June 5, 2022

Acknowledgements

This project would not have been possible without the kind support and help of many individuals. We would like to extend our sincere thanks to all of them.

First of all, We would like to thank our esteemed Principal, Prof. (Dr.) Jacob Thomas V, for his guidance and support in maintaining a calm and refreshing environment to work in and also for providing the facilities that this work demanded.

We are highly indebted to our Project Coordinator, Dr. Sindhu L, Assistant Professor and Head of the Department, Dr. Preetha Theresa Joy, Professor for their guidance, support and constant supervision throughout the duration of the work as well as for providing all the necessary information and facilities that this work demanded.

We would like to thank our Project Guide, Dr. Sindhu L, Assistant Professor for her support and valuable insights and also for helping us out in correcting any mistakes that were made during the course of the work.

We offer our sincere gratitude to all our friends and peers for their support and encouragement that helped us get through the tough phases during the course of this work.

Last but not the least, we thank the Almighty God for guiding us through and enabling us to complete the work within the specified time.

Arya Jaydev K M

George M Cherian

Lijo Zechariah James

S Sandeep Pillai

Abstract

With the advent of the pandemic, remote work/learning has become very monotonous and has taken a toll on the mental well being of employees and students. We believe by allowing a more dynamic environment, we can capture the charm of actually being in an office/classroom while being safe from the potential threat of Covid-19 and any future circumstances. Our goal is to create a virtual office or classroom that people can interact with using their virtual avatars, to create a sense of connection with one's colleagues or classmates, that a typical Zoom Video Call cannot allow.

This project aims to implement a close to life environment that reflects the typical day to day activities of offline offices, classrooms etc. Users can create 2D avatars with which they can walk around a virtual office/classroom and interact with each other and the environment. Each room in the office is a virtual voice channel that cannot be listened to by people outside of the room. Meeting rooms can be locked to allow for privacy, and create a real office space experience. Each user can have their own office space that they can decorate and customise.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Proposed Project

1.1.1 Problem Statement

With the advent of the pandemic, organizations and institutions across the globe were forced to quickly shift to remote work/learning and this lead to the meteoric rise in the usage of platforms like Zoom, Google Meet, Microsoft Teams etc. Over time people from different walks of life incorporated these platforms into their daily workflow to only realize that the continuous usage of these platforms had become monotonous and had taken a toll on one's mental well being as such platforms provided a mere source of communication without factoring in the underlying characteristics of a typical human conversation. Hence, there needs to be a platform that provides the technical functionality of the above mentioned platforms along with certain features that brings in a sense of connection amongst individuals.

1.1.2 Proposed Solution

RealTES (Real Time Environmental simulation) is a virtual office where everyone has their own avatars. This is to mimic the in-office working conditions. The application has private meeting rooms and customised personal work-spaces. Employees can make use of Deep-focus spaces for enhancing their workflow. The application also has recreational space where people can socialise and hangout with each other. Some of the other features provided by our system include tools to enhance productivity without compromising one's health and well being, by maintaining work life balance.

This two dimensional work space will be built with the aid of a Game Engine of our choice. The application aims to be browser first, so that there is no overhead of downloading additional software. Users can view their entire work space from the top down perspective and use either keyboard or mouse inputs to navigate around the office.

Whenever multiple users are in the same room, they seamlessly connect with each other via audio and video channels using the WebRTC technology, which abstracts the underlying UDP networking protocols, while offering high throughput and low latency communication.

In the background, while the virtual office is running, a secondary application system makes use of computer vision to determine the user's alertness levels and workload by making use of facial

cues, and advices the user to take breaks. There are also inbuilt reminders, alerts and timers to allow the user to keep track of deadlines.

RealTES hopes to bring more connectivity to digital interactions over conventional internet calls by emphasising on the social factor. The pictorial representation of a work space allows the human mind to form a sense of physical being within the organisation.

Some potential challenges to our system will be, the inability to convey real physical sensations beyond auditory and visual stimulation. We may also be weighed down by the kinks that any online application faces like network faults.

Chapter 2

System Study Report

2.1 Literature Survey

We analyzed 10 different papers primarily centred around topics like WebRTC, Game Development, VOIP, Network Sockets and Facial expression detection and the following summarizes the papers that we analyzed.

This paper [1] discussed the history of Real Time Communication systems and the features that WebRTC provides such as streaming content, obtaining network information, dealing with NAT and firewall etc, alongside a general high level architecture. The protocols used such as ICE and STUN are also discussed, through its interaction over the O.S.I network layer, and the reason to use UDP instead of TCP. The paper implements a thermal WebRTC application using Scaledrone, which is a push messaging service; WebRTC, for the website server to send the browser IDs; HTML for the frontend, with the core of the system being a JavaScript API server backend.

This paper [2] evaluates the performance of a WebRTC-based video conferencing system. The paper evaluates 6 parameters, namely Baseline Experiments, Cross Traffic, Multi-Party Topology and Video Codecs, Mobile Performance and Real Wireless Networks. The bedrock of WebRTC is the User Datagram Protocol which helps realise the high throughput and low latency needs while congestion control is handled by the GCC algorithm with the usage of adaptive threshold.This paper establishes benchmarks for performance evaluation and the overall performance was as good as a TCP flow. A UDP stream having GCC and adaptive threshold implemented was offered a higher priority than TCP in terms of cross traffic. In terms of video codecs, the WebRTC system utilized the VP8 codec.VP9 is the latest version and is more efficient (lower bitrate because of superior compression) but requires higher computational power. The performance on mobile devices and real wireless networks was deemed adequate except for the lack of WebRTC on iOS devices.

This paper [3] implements a multi user virtual learning environment that simulates a museum of history using Autodesk 3D Max. Students walk around the simulation as virtual avatars with gestures, implemented using Avatar Studio 2.0. An interaction server is built using Java for enabling users to walk through the world and to communicate with the different avatars by using different interactive gestures. The user executes a Client browser application that is fundamentally in charge of the visualization and communicating with the server. The server knows the positions of all avatars

and communicates this information to the other clients. This way students can walk around and share emotions with interactive avatar gestures, making learning more realistic and complete.

Construct 2 [4] is a very simple commercial application for the creation and design of two-dimension games. In Construct 2, the classic coding is replaced with dragging and dropping game objects. Games developed in Construct 2 are implemented in HTML5 and Javascript. In this game engine, every layout has its event sheet in which all the steps related to the addition of the events, addition of actions and other events. This eliminates the need for complicated scripting in programming languages with fixed syntax, which makes it an easy tool to learn. It allows a creator to define objects, its types, give objects behaviour and effects and group them into families. Instant preview and iterative development makes it a fast prototyping tool with flexible behaviours, stunning visual effects and capabilities.

A game engine [5] provides a main framework and common functions for developing games, which is the core of controlling games. Another feature of Unity3D is that game resources and objects can be imported or exported in the form of a package, which can easily make different game projects share development work. Therefore, using packages can greatly improve efficiency in game development. The component model is applied in the Unity3D game development, which provides a scalable programming architecture. Game function modules can be reused conveniently in this component model. The component model is applied in the Unity3D game development, which provides a scalable programming architecture. Game function modules can be reused conveniently in this component model. The game is controlled by programming scripts to access the internal of Unity3D. There are many system classes in Unity 3D,which are useful in scripts. There are two kinds of classes. Runtime classes and editor classes.

I room [6] is an environment for intelligent interaction. It will provide support for formal business meetings, tutorials, project meetings, discussion groups, and ad hoc interactions. which provides human participants with intelligent and intelligible task support, process management, collaborative tools, and planning aids. One key intelligent system used in the I-Room is the I-X Technology process support framework and I-Plan. With the advent of 3D virtual worlds(for example, Second Life and OpenSimulator), I-room was able to link I-X technologies so that they could support a community connected via such a virtual worlds meeting space. I-Plan is an intelligent planning aid that can offer task-support help, generate and refine plans to adapt them to the situation at hand, support the execution of standard operating procedures, support the various stages of conducting a meeting, help handle post-meeting group actions, and so on..

This paper [7] is a general case study of what a video game engine is and the features it provides to the developer such as Scripting, Rendering, Animation, Audio and Network abstractions. It also compares the numerous available game engines for the objective of making “Interactive worlds” for non-entertainment purposes. It maps out the pros and cons of all the different choices for game engines. The comparisons were done on the basis of a multitude of factors, such as basic features, price, customer support, flexibility, usability, functionality, multimedia support, etc.

Real-time walk [8] through with vivid lighting/shadowing effects requires high computation performance. Therefore the author adopts the next-generation 3D game engine called UDK(Unreal Development kit, developed by Epic Games). In this, the system is doing high-quality 3D CG rendering, real-time weather, environment simulation and game-like man-machine interaction. UDK is

a popular 3D game engine with the superior performance of light and shadow, and it also had good compatibility with models created with Maya 2011. The virtual space of "Reizei-ke"is constructed based on the default template map of UDK.

This paper [9] implements a system employed using Computer Vision to analyse blink rate, eye closure, yawning etc to effectively and quickly identify the drowsiness of a driver during driving the vehicle and alert the driver accordingly. In the proposed work, pre-existing features for facial landmark detection is implemented to identify the state of drowsiness and fatigue. These predefined landmarks help in shape prediction to clearly identify the various regions of the face like eyebrows, eye, mouth region etc and changes in parameters report various expressions of the person. Camera monitors and captures to extract frames. Each extracted frame ishttps://www.overleaf.com/project/6299004a4fabe7b483e4f430 analyzed to study the pattern of facial features. There are threshold values for eye aspect ratio and mouth aspect ratio, which upon exceeding, will record as a blink and a yawn. Blink and yawn rates are analyzed to determine whether the driver is drowsy and an alert is triggered.

The authors build a system [10] that tries to add visual cues to a 3D face generated based on the user's facial features. This system makes use of a high quality 3D model provided by Rocket Libraries. A text to speech framework is used whose output is sent to a module that tries to predict the required actions and non-verbal cues to be adopted. There are 2 databases involved, a Facial Mocap Database and a Head and Gaze motion database. Using the data available in these two databases and the user image captured , the system synthesizes an animation for lip synchronization and adds the non verbal cues by selecting the appropriate motion to be exhibited by the face. This is then used to form a 3D mapping of the user's face and the final animation is done with the help of the graphics library (OpenGL ES) to get a 3D avatar as output. This system adopts the eFase framework and uses K-means clustering to normalize all motion nodes. The authors shocased this system to a select audience and concluded that that this system was better suited for informal conversations.

2.2 Proposed System

RealTES (Real Time Environmental simulation) is a virtual office where everyone has their own avatars. This is to mimic the in-office working conditions. The application has private meeting rooms and customised personal work-spaces. Employees can make use of Deep-focus spaces for enhancing their workflow. The application also has recreational space where people can socialise and hangout with each other. Some of the other features provided by our system include tools to enhance productivity without compromising one's health and well being, by maintaining work life balance.

This two dimensional work space will be built with the aid of a Game Engine of our choice. The application aims to be browser first, so that there is no overhead of downloading additional software. Users can view their entire work space from the top down perspective and use either keyboard or mouse inputs to navigate around the office.

Whenever multiple users are in the same room, they seamlessly connect with each other via audio and video channels using the WebRTC technology, which abstracts the underlying UDP networking protocols, while offering high throughput and low latency communication.

In the background, while the virtual office is running, a secondary application system makes use of computer vision to determine the user's alertness levels and workload by making use of facial cues, and advices the user to take breaks. There are also inbuilt reminders, alerts and timers to allow the user to keep track of deadlines.

RealTES hopes to bring more connectivity to digital interactions over conventional internet calls by emphasising on the social factor. The pictorial representation of a work space allows the human mind to form a sense of physical being within the organisation.

Some potential challenges to our system will be, the inability to convey real physical sensations beyond auditory and visual stimulation. We may also be weighed down by the kinks that any online application faces like network faults.

Chapter 3

Software Requirement Specification

3.1 Introduction

3.1.1 Purpose

With the advent of the pandemic, remote work/learning has become very monotonous and has taken a toll on the mental well being of employees and students. By allowing a more dynamic environment, one can capture the charm of actually being in an office/classroom while being safe from the potential threat of Covid-19 and any future circumstances. This project aims to help people create a sense of connection with one's colleagues or classmates, that a typical Zoom Video Call cannot allow.

3.1.2 Document Conventions

This document follows MLA (Modern Language Association) Format. Boldfaced text has been used to emphasize section and sub-section headings. Highlighting is to point out words in the glossary and italicized text is used to label and recognize diagrams.

A/V - Audio/Video
RTC - Real Time Communication

3.1.3 Intended Audience and Reading Suggestions

This intended audience of this document includes all stakeholders of this project who are supposed to review and sign-off this document. The document will prove to be useful for the developers of the project as well as people looking to gain knowledge on game development and real-time communication.

3.1.4 Project Scope

This project aims to design and implement a 2D virtual office or classroom with Unity game engine and real-time communication technologies, for people to interact with their colleagues/classmates as virtual avatars.

3.1.5 Overview of Developer's Responsibilities

The developer is responsible for overseeing the implementation of the project in a manner that the final implementation goes beyond the prototype and can be used by multiple people. The developer should also maintain appropriate coding standards and design and contribute to technical design documentation.

3.2 Overall Description

3.2.1 Product Perspective

During the pandemic, remote/hybrid working has become the new normal. This situation has increased the usage of platforms like Microsoft Teams, Google Meet and Zoom. The over usage of these platforms is affecting the mental well being of workers. This project (Real-TES) is an upgraded version of these already existing video calling/ virtual meeting solutions. The project focuses more on bringing a real-world experience to virtual meeting rooms. The existing virtual meeting solutions fail to effectively mimic human interactions, which is affecting the mental well being of employees/students. This product will be built by keeping in mind the nature of human conversations. The dynamic approach of meeting rooms through the usage of spatial audio and video features will help bring an real world like feeling to the interactions.

3.3 Product Functions

3.3.1 Login/Signup

Here users will create a new account or login into his/her existing account using Google login or email ID.

3.3.2 Create/Customize Avatar

In this window, the user will create/customize their avatar.

3.3.3 Create/Customize the Workspace/Gamespace

Here users will create a new workspace or customize the existing workspace.

3.3.4 Workspace/Gamespace Navigation

User will be able to navigate their avatar through the workspace. The user will be able to enter deep-focus space or meeting rooms as per their wish.

3.3.5 Audio/Video communication via webRTC

Users will be able to interact with each other through the technology of WebRTC. The application will further implement Spatial Audio/Video feature to help emulate a close to real-life communication experience.

3.3.6 Presentation tools/aids

Availability of different presentation tools like screen sharing,file sharing, whiteboards etc. will help users to pitch their ideas effectively just like in real life.

3.3.7 Schedules and reminders

Integrated schedules and reminders will help users to keep track of their workflows and stay on top of deadlines.

3.3.8 Fatigue tracking

This feature will take care of the mental/physical well being of users by avoiding overworking on tasks providing timely reminders based on parameters judged through facial cues.

3.4 User Classes and Characteristics

The different users who will use this product:

3.4.1 Employees

Employees will use this app to do their day to day tasks. Real-TES will help them to attain maximum productivity without compromising their mental/physical well being.

3.4.2 Students

Students will use this app to interact with other students and teachers and gain a pleasant and dynamic learning experience.

3.4.3 Organization

The organizations will be able to create their own workspace and invite the employees for work.

3.5 Operating Environment

This is a system which can operate on any modern web browser and on any operating system. It also requires a good internet connection.

3.5.1 User Classes and Characteristics

The different users who will use this product:

3.5.2 Employees

Employees will use this app to do their day to day tasks. Real-TES will help them to attain maximum productivity without compromising their mental/physical well being.

3.5.3 Students

Students will use this app to interact with other students and teachers and gain a pleasant and dynamic learning experience.

3.5.4 Organization

The organizations will be able to create their own workspace and invite the employees for work.

3.5.5 Operating Environment

This is a system which can operate on any modern web browser and on any operating system. It also requires a good internet connection.

3.5.6 Design and Implementation Constraints

1. Security: Real-TES should encrypt the video/audio communications to ensure the safety of the organization and it's important information.
2. Reliability: Real-TES should be reliable enough to work in different network conditions and doesn't fail during high traffic periods (Peak Hours).
3. Usage of Data: Real-TES should use the least amount of data/bandwidth. In order to work in different conditions.
4. Regulatory Policies: NA
5. Hardware Limitations: Real-TES requires a system with camera and microphone.
6. Interfaces to other application: NA
7. Parallel operations: NA
8. Audit Functions: NA
9. Control Functions: NA
10. Safety and Security Considerations: The system adheres to safety and security policies.
11. Reliability Requirements: The total number of bugs in the system shall not exceed 1% of the total line number of code, except connection reliability, which is out of range.
12. Criticality of the Application: NA

3.5.7 User Documentation

All documentation will be made in accordance with requirements pertaining to open source software under the GNU General Purpose License.

3.5.8 General Constraints

- Overall quality of the system depends on the strength of the network.
- The system fails to fully capture the facial cues.
- Doesn't work on iOS device browsers.
- Striking a fine balance between customisation and scalability.

3.5.9 Assumptions and Dependencies

The following assumptions are made with regard to this project:

- Every employee in the organization has an account
- Every organization knows meeting room requirements in advance.
- Every device the employees are using contains a Webcam and microphone.

The following are the Dependencies:

- A modern browser
- Unity
- WebRTC

3.6 External Interface Requirements

3.6.1 User Interfaces

The user is offered a 2-Dimensional top view of the work space after successful login and Avatar creation. In this view, each user can see the entire space containing the various rooms, the users and their locations and other interactable objects. Each user's avatar will be displayed along with his/her name. The user can interact with another user by either walking to or double clicking on his/her avatar. Interaction between users is possible at all regions except the Deep Focus areas. Once the user is within a particular distance of another, a tile pops up with the video of the other user and they can chat. The gamespace contains multiple rooms ranging from cubicles and meeting rooms to cafeterias and recreational rooms. All rooms will have interactable objects which are appropriate to the function and purpose of each room. The cubicle assigned to each user can be further personalized by him/her with objects of their choice. Each user has the freedom to move about the workspace and this motion is governed through keyboard and mouse inputs.

3.6.2 Hardware Interfaces

Any device that has an inbuilt/external microphone and camera which is capable of accessing the internet and offers support for modern web browsers is capable of running the platform.

3.6.3 Software Interfaces

- Firebase Auth: Authenticates users
- Photon: Unity package that manages remote server events
- Netcode: Unity package that manages server concurrency among peers.

3.6.4 Communication Interfaces

The platform requires E-mail for account setup and authentication.The very heart of this platform is WebRTC which helps establish communication channels that offer low latency and high throughput. WebRTC helps abstract the underlying UDP mechanism while also handling congestion control using the Google Congestion Control (GCC) algorithm.

3.7 Hardware and Software Requirements

3.7.1 Hardware Requirements

1. An Intel Pentium 4 processor or later that's SSE3 capable
2. A Web Cam to capture video data.
3. A Microphone to capture speech.

3.7.2 Software Requirements

1. **Operating System:** Any OS that can run a browser with WebGL support.
2. **C#:** is a general-purpose, multi-paradigm programming language. C encompasses static typing, strong typing, lexically scoped, imperative, declarative, functional, generic, object-oriented, and component-oriented programming disciplines.
3. **Unity:** is a cross-platform game engine with a built-in IDE developed by Unity Technologies. It is used to develop video games for web plugins, desktop platforms, consoles and mobile devices.
4. **WebRTC:** is a free and open-source project providing web browsers and mobile applications with real-time communication via application programming interfaces.
5. **SQLite:** is a C-language library that implements a small, fast, self-contained, high-reliability, full-featured, SQL database engine. SQLite is the most used database engine in the world.
6. **Photon Unity Networking (PUN):** is a Unity package for multiplayer games. Flexible matchmaking gets your players into rooms where objects can be synced over the network. Remote Procedure Calls, Custom Properties or "low level" Photon events are just some of the features of this package.
7. **Netcode for GameObjects:** is a Unity package that provides networking capabilities to GameObject and MonoBehaviour workflows. The framework is interoperable with many low-level transports, including the official Unity Transport Package.
8. **Web Browser:** Any popular web browser such as Mozilla Firefox, Google Chrome or Microsoft Edge can be used to access the GUI provided by this project.
9. **OpenCV:** OpenCV (Open source computer vision) is a library of programming functions mainly aimed at real-time computer vision. The library is cross-platform and free for use under the open-source BSD license. OpenCV supports the deep learning frameworks TensorFlow, Torch/PyTorch and Caffe.

3.8 Functional Requirements

3.8.1 Login/Registration

The user will have to login to access the platform's functionality. Users can login/sign up by either using their email or Google account. Users who sign up for the first time will need to verify their accounts through a link sent to their provided email and will need to provide additional details like display name, phone number, designation, etc..

3.8.2 CRUD on Avatar

Every user who logs into the platform for the first time will need to set up an avatar for him/her. The avatar details will be saved and will be automatically retrieved the next time the user logs in. Registered users can make modifications to the avatar at any point of time within the game space.

3.8.3 CRUD on Game Space.

The user will have an option to either join a work space created by someone or create a new one by selecting from a choice of already existing templates available on the platform. Regardless of the choice, each user will have the ability to personalize sections of the work space.

3.8.4 Game Space Navigation

Each user will have unrestricted access to all areas within the work space except for the designated Deep Focus regions.An internal lock mechanism will also restrict access to meeting rooms if they're occupied. Avatar navigation is governed through keyboard and mouse inputs.

3.8.5 Audio/Video communication via webRTC

When a user comes within a particular distance of another user or if the user is in a meeting room, a communication channel is established among all users in proximity via webRTC. In addition to this the platform will implement spatial audio and video features that make use of distance vectoring to offer characteristics of real human conversation based one's physical proximity to the user.

3.8.6 Integration of presentation tools/aids

To further enhance communication, all users will have the option to use presentation tool/aids including but not limited to screen sharing, file sharing, whiteboard,etc.

3.8.7 Recreational Spaces and Deep Focus Spaces

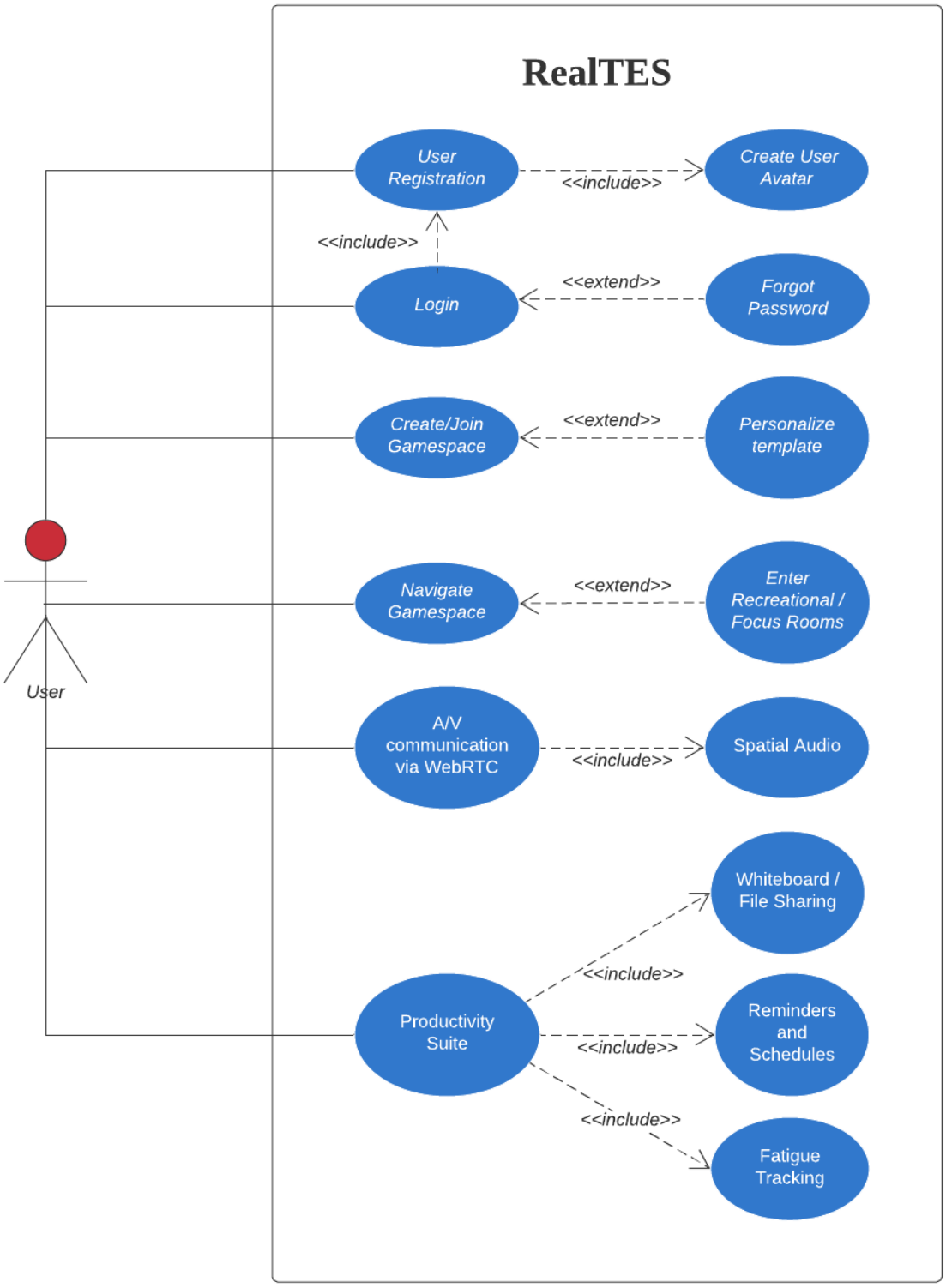
RealTES will have designated Recreational Spaces where employees can hang out and interact with each other as they would do in a traditional office set up.These spaces include the Cafeteria, Game Room, etc. To further enhance productivity, the platform will also offer Deep Focus spaces where the users will be able to temporarily isolate themselves.

3.8.8 Schedules and Reminders

RealTES will offer timely reminders and updates regarding deadlines, upcoming meetings, etc so that users can always stay on top of the work assigned to them.

3.8.9 Fatigue tracking.

While working from home, people often become deeply involved in their work that they lose track of their external environment and time which leads to issues that affect one's physical and mental well-being. In RealTES, users will have the option of enabling a fatigue tracker that uses Computer Vision to detect signs of tiredness and occasionally nudges the user to take a break.


RealTES
User Registration
Create User Avatar
<<include>>
<<include>>
Login
<<extend>>
Forgot Password
Create/Join Gamespace
<<extend>>
Personalize template
User
Navigate Gamespace
<<extend>>
Enter Recreational / Focus Rooms
A/V communication via WebRTC
<<include>>
Spatial Audio
Whiteboard / File Sharing
<<include>>
Productivity Suite
<<include>>
Reminders and Schedules
<<include>>
Fatigue Tracking

3.9 Non-functional Requirements

3.9.1 Performance Requirements

- RealTES should offer low latency, reliable audio/video communication and multiplayer environment to it's users at all times.
- Performance requirement by the user side should be minimal. The browser application must be light-weight enough to run on most modern web browsers even with slower internet connections.
- Server machines should have powerful CPU and high bandwidth internet access so that it can handle multiple users at the same time.

3.9.2 Security Requirements

- Data Protection: Collected data must be stored on a secure server which cannot be accessed by an unauthorised personnel.
- User Authentication: Only registered users shall be allowed to use the platform.
- Life Time Of Data: Data should be backed-up or cleared periodically depending on the user requirements.
- Encryption: All communication via WebRTC and HTTP must be secured and encrypted.

3.9.3 Software Quality Attributes

- Robustness: The software should be able to handle a lot of data, store them and perform aggregate operations on them without error and failure.
- Correctness: The software must correctly get distance vectoring information for proper spatial audio/video.
- Reliability: The system should be able to work satisfactorily even in slow network conditions.
- Usability: The software must be easy to use and convenient for all users.
- Learnability: The software should be self explanatory and provide graphical cues for usage wherever necessary.
- Extensibility: Required modification at specific locations can be made without the software failing.
- Scalability: The software must be able to handle multiple concurrent requests from large number of users without failing.
- Testability:The software must be modularized and have a proper structure in order to conduct tests.

3.10 Other Requirements

- Licenses: The application must adhere to licenses and regulations set forth by all platforms which are supported.

Chapter 4

System Design

4.1 System Architecture

The system architecture shows the various modules of the system and the interaction between them in the RealTES System. A brief description of the working of these modules is also given below.

4.1.1 Unity Game Engine

The Unity Game engine is the heart of RealTES application, as the entire ecosystem interacts with this single module. It helps run the game instance through WebGL, abstracting away the inner workings of the application, like the player movement, animations, sounds and physics interactions. It communicates with the Photon Multiplayer server to establish concurrency among every client.

4.1.2 Photon Multiplayer Server

This module is the main server of the application, that manages the Rooms and server node instances. It ensures that all clients connected to the same room are in sync with each other. It seamlessly abstracts away the creation of rooms and the joining of rooms by users. Essentially, its the environment on top of which the game instance runs on in the network.

4.1.3 Communication Module

Audio/Video communication between users is handled over WebRTC. WebRTC helps abstract the underlying UDP mechanism while also handling congestion control using the Google Congestion Control (GCC) algorithm. Agora is a Communications as a Service platform that provides WebRTC technology. It provides APIs for establishing RTC connections and control communication and live broadcast services. Users communicate with each other using Agora API which will be integrated into RealTES.

4.1.4 Presentation Aids/Tools

Presentation aids and tools are used to help the users to communicate effectively. Presentations are one of the most unavoidable parts of an organization setup. It will make the process of idea pitching way easier. The aids offered by the platform include whiteboard and file sharing features in

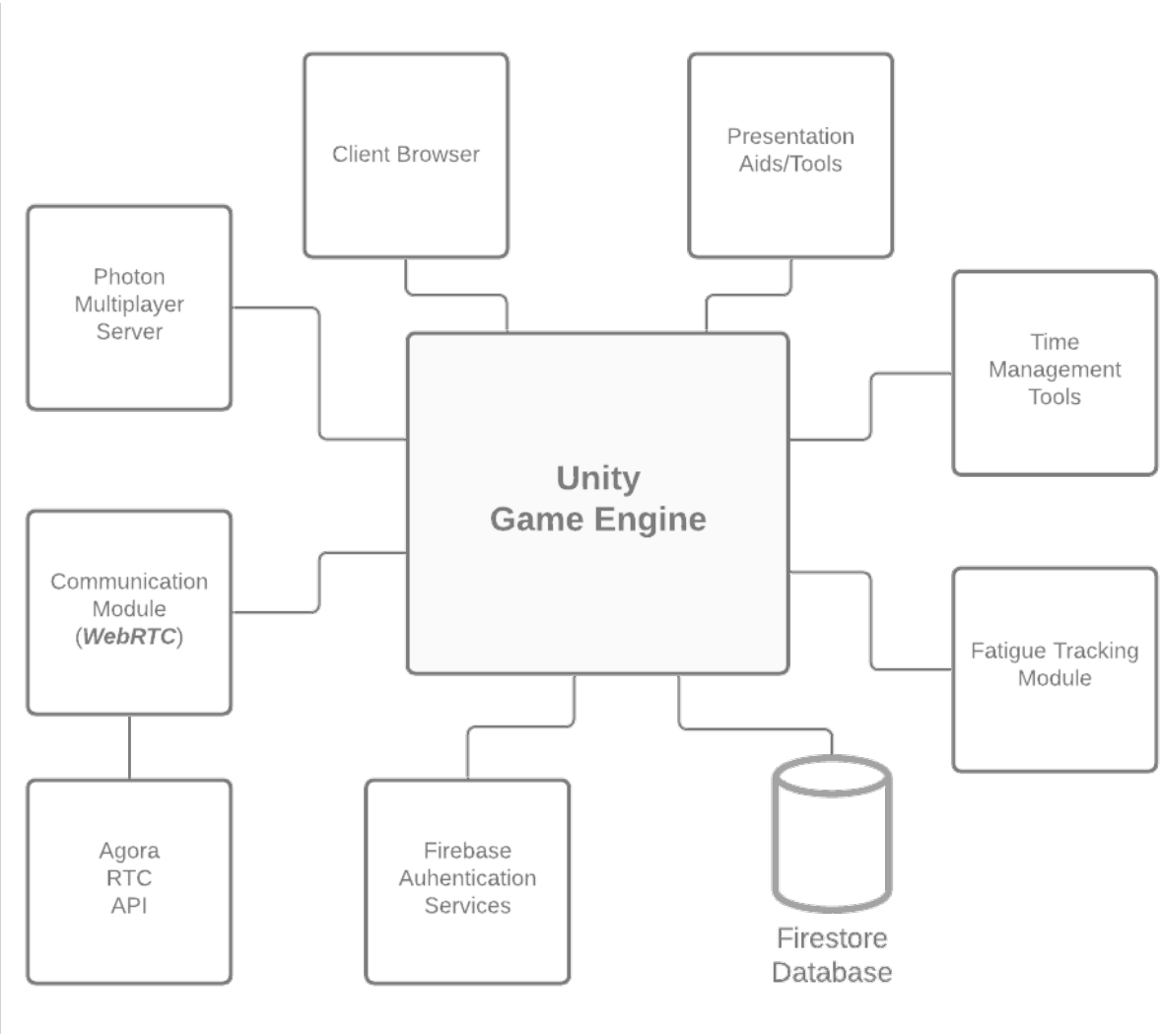


Figure 4.1: RealTES "Virtual Office" System Architecture.

addition to screen sharing via RTC. There will be an option to customize the features to incorporate different organizational needs.

4.1.5 Fatigue Tracking Module

This module exists independently and is tasked with looking after the well-being of the users by periodically monitoring the user's facial cues for noticeable signs of tiredness. This module makes use of Computer Vision to analyze the user's face for such signs and feeds the input to a trained model for evaluation. If the model determines that the user is exhibiting signs of tiredness, the user will be offered suggestions to stay fresh and focused by sending system wide alerts.

4.1.6 Time Management Tools

This module abstracts the set of services and tools on the platform that are managed by the Calendar API. These tools include meeting scheduling, deadline reminders and new assignment alerts. Such tools help utilize the available time in the best possible way and reduce clashes which in turn boost productivity.

4.2 Input Design

4.3 Database Design

Given below is the database design of RealTES for virtual office/classroom. RealTES uses Google Cloud Firestore for storing persistent data. It is a NoSQL database, that stores data as documents inside collections representing tables. The two tables required by RealTES are, Users collection and Rooms collection.

The Users collection stores information such as id, username, email, avatar name, contact information and workspaces to which the user belongs. The Rooms collection stores information about offices/workspaces, such as their id, name, creator, unity scene template and the name and position of custom objects to be placed in the room.

Users	
id	string
username	string
email	string
avatar	string
mobile	string
designation	string
rooms	array

Table 4.1: Users Collection

Rooms	
id	string
name	string
owner	string
template	string
roomObjects	array of map

Table 4.2: Rooms Collection

4.4 Libraries and Packages Used

- Photon for Unity - a framework for building game server logic on top of its existing robust backbone.
- JSON.Net for Unity - Serializes Firestore query parameters and deserializes JSON response.
- Assets and Texture packs - For designing gamespace.
  - 2D Mega Texture Pack
  - Stylized PBR Texture Pack
  - Serene Village Revamped Pack
  - RPG World Pack
- Firebase JavaScript SDK - Available via CDN for accessing Cloud Firestore.
- Agora-Web-SDK-NG (v4.X) - Available via CDN, abstracts WebRTC.
- Python modules for implementing Fatigue Tracking
  - scipy - For calculating spatial distance between landmarks.
  - imutils - Utilities needed for obtaining landmark indices.
  - dlib - For the facial landmark detection model.
  - opencv-python (cv2) - Abstracts Computer Vision functionality for Python
  - gtts - Google Text-to-Speech for generating audio alerts.
  - numpy - Numpy arrays (nd.array) used for handling landmark indices.

4.5 Module Description

4.5.1 Unity Game Engine

The Unity Game engine is the heart of RealTES application, as the entire ecosystem interacts with this single module. It helps run the game instance through WebGL, abstracting away the inner workings of the application, like the player movement, animations, sounds and physics interactions. It communicates with the Photon Multiplayer server to establish concurrency among every client.

The Unity Game Engine abstracts the following:

1. The Graphical rendering pipeline of the gamespace.
2. The Audio System
3. Player Input handler
4. Physics Engine.
5. Event Listeners
6. Script Execution System.

4.5.2 Photon Multiplayer Server

This module is the main server of the application, that manages the Rooms and server node instances. It ensures that all clients connected to the same room are in sync with each other. It seamlessly abstracts away the creation of rooms and the joining of rooms by users. Essentially, its the environment on top of which the game instance runs on in the network.

The Photon Server handles the following:

1. Joining and creating room instances
2. Load Balancing over network.
3. Nearest server finder.
4. Game Object position synchronisation.
5. Buffering level state in memory.
6. Remote Procedure Calls Wrapper.

4.5.3 Communication Module

Audio/Video communication between users is handled over WebRTC. WebRTC helps abstract the underlying UDP mechanism while also handling congestion control using the Google Congestion Control (GCC) algorithm. Agora is a Communications as a Service platform that provides WebRTC technology. It provides APIs for establishing RTC connections and control communication and live broadcast services. Users communicate with each other using Agora API which will be integrated into RealTES.

The Agora API helps abstract the following RTC Functionalities:

1. Client creation and initialization
2. Track creation and initialization
3. Track publish and subscribe operations
4. Remote User Handling
5. Screen Sharing
6. Event Listeners

4.5.4 Presentation Aids/Tools

Presentation aids and tools are used to help the users to communicate effectively. Presentations are one of the most unavoidable parts of an organization setup. It will make the process of idea pitching way easier. The aids offered by the platform include whiteboard and file sharing features in addition to screen sharing via RTC. There will be an option to customize the features to incorporate different organizational needs.

4.5.5 Fatigue Tracking Module

This module exists independently and is tasked with looking after the well-being of the users by periodically monitoring the user's facial cues for noticeable signs of tiredness. This module makes use of Computer Vision to analyze the user's face for such signs and feeds the input to a trained model for evaluation. If the model determines that the user is exhibiting signs of tiredness, the user will be offered suggestions to stay fresh and focused by sending system wide alerts.

4.5.6 Time Management Tools

This module abstracts the set of services and tools on the platform that are managed by the Calendar API. These tools include meeting scheduling, deadline reminders and new assignment alerts. Such tools help utilize the available time in the best possible way and reduce clashes which in turn boost productivity.

Chapter 5

Data Flow Diagram

A Data Flow Diagram(DFD) known as Bubble chart, is a graphical representation of the "Flow" of data through an information system, modelling it's process aspects. A DFD is often used as a preliminary step to create an overview of the system without going into great detail, which can later be elaborated. A DFD shows what kind of information will be input to and output from the system, how the data will advance through the system etc. DFD can also be used for the visualization of data processing(Structured Design).

5.1 Level 0 DFD

Highest abstraction level DFD is known as level 0 DFD, which depicts the entire information system as one diagram concealing all the underlying details. This diagram offers a bird's eye view on how multiple users interact with the RealTES platform and also showcases how the Photon Multiplayer Server manages such interactions and responds to them.

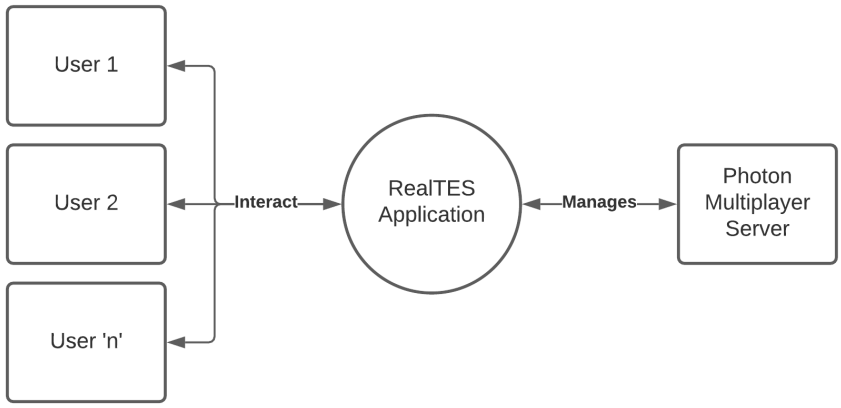


Figure 5.1: RealTES Level 0 DFD

5.2 Level 1 DFD

Level 1 DFD provides a more detailed breakout of pieces of the context diagram. The main functions carried out by the system is considered as we break down the high level process of the context diagram into it's sub-processes. This diagram offers a detailed view on the key components of the RealTES system and abstracts the core functionalities offered by it. The diagram further elaborates the interactions specified in Level 0 DFD by showcasing how the Unity App which acts as the bedrock to the platform encapsulates the flow of data between the platform and the other integral modules.

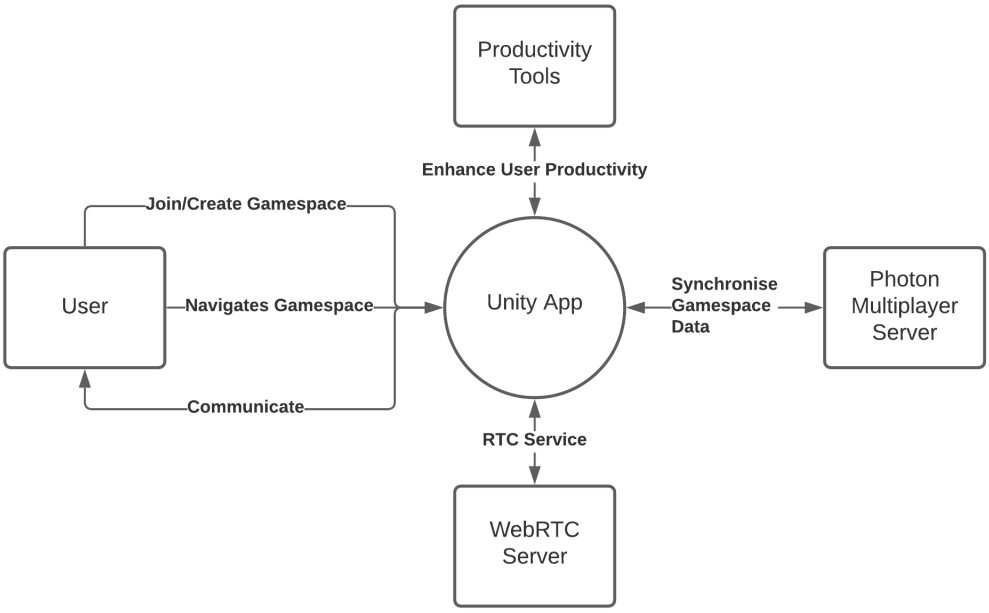


Figure 5.2: RealTES Level 1 DFD

5.3 Level 2 DFD

A Level 2 DFD provides a more detailed look at the processes that make up the system than the Level 1 DFD. This diagram explains in detail how the flow of data is carried out in the system.

The user joins the RealTES-lobby and logs in via Firebase authentication. The logged-in user will be able to create a workspace and customize it, or join an existing workspace and will also have the choice to make changes to their personal avatars if required. Users can navigate through the workspace and the interactions between the avatars and workspace objects are synchronized in real-time by the Photon server. The Agora API handles audio/video communication between the users by abstracting the WebRTC communication channel. RealTES also offers a suite of productivity

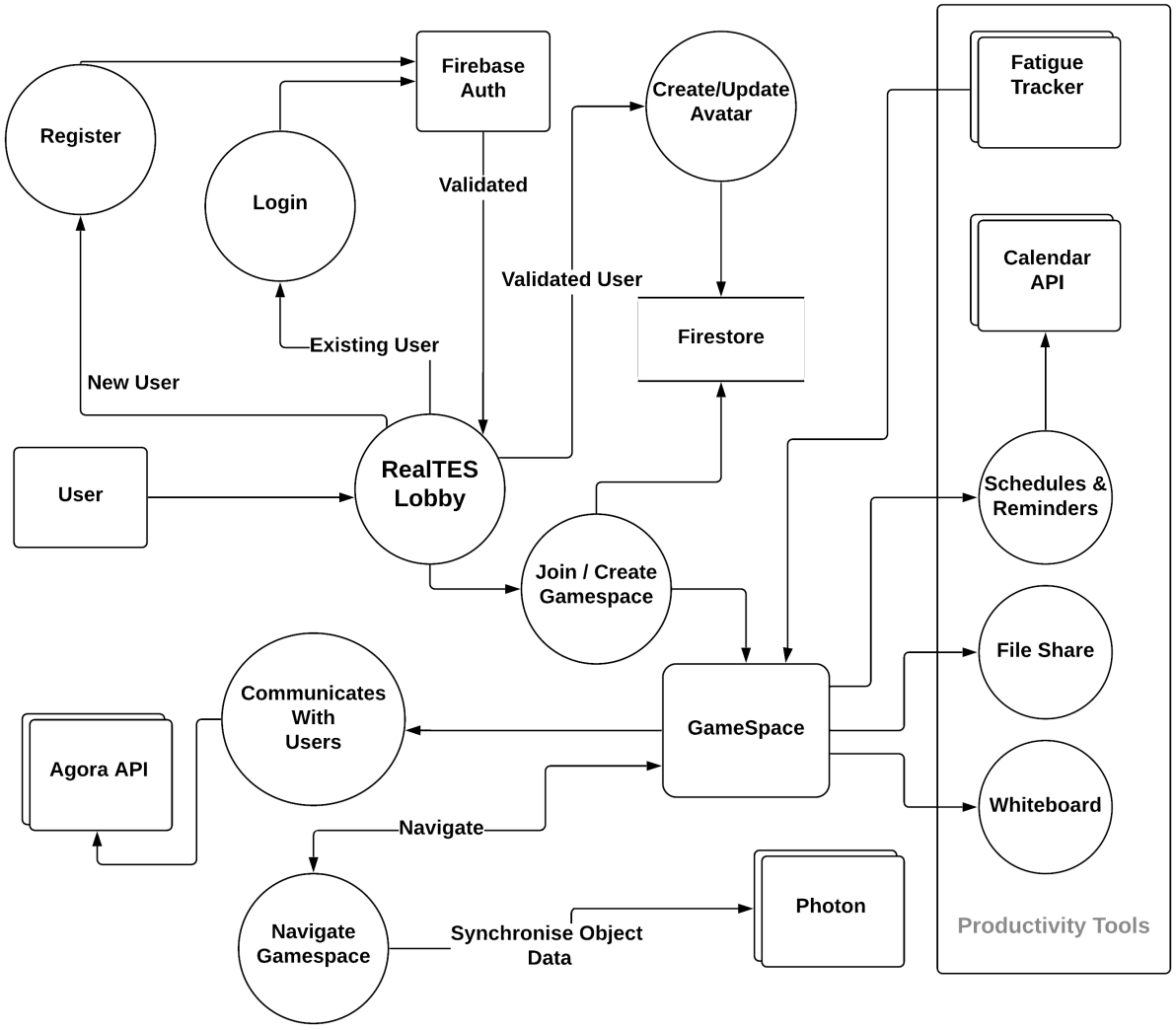


Figure 5.3: RealTES Level 2 DFD

tools for fatigue tracking, scheduling meetings, setting reminders and deadlines, whiteboard, file sharing etc.

Chapter 6

Implementation

6.1 Algorithms

6.1.1 Game space Initialisation

We store fixed templates in the game server, however each user is able to modify the office to their taste. Therefore we implement this by adding the new objects on top of the default template.

1. User chooses room
2. Photon server creates photon room with corresponding ID and loads template scene.
3. Agora creates new channel with roomID and adds new user.
4. Unity fetches the list of custom props stored in Firestore by users.
5. Iterate over this list and create game objects with their corresponding prefabs.
6. Initialize the game objects in Photon server at their respective positions specified.

6.1.2 Room Entry Exit Algorithm

Each interior section is actually an area within the gamespace placed away from the exterior. When the player touched the door, they get teleported towards this interior space, creating an illusion of entering a building or room.

However, this simple implementation has its issues. Upon being teleported to the interior, the teleport trigger of the interior triggers and teleports the player back outside. This again triggers the exterior teleporter and teleports them back inside. This results in an infinite loop. Therefore, we implemented a blocking system:

1. We first place two teleporter objects at the door exterior and interior.
2. We attach them to a parent door controller. The door controller has a list of teleport system customers.
3. When an object collides with any one of these teleporters, they are entered to this customer array.

4. If they were not part of the customer array, they are teleported to the respective sister teleporter coordinate.
5. If they are already in the customer array, do not teleport them.
6. When a object stops colliding with any one of the teleporters, they get removed from the customer array.

6.1.3 Spatial Communication Algorithm

Each user can communicate with other users within a limited distance from them, to establish a sense of immersion.

1. Upon joining the room, the user joins the Agora WebRTC session server with the same name as the room id.
2. Instead of subscribing to all users in the call as Agora usually does, we instead prevent this and block the template function attempting to connect to all user's tracks.
3. Instead, we implement our own logic for when a user can subscribe to another user's track.
4. Within Unity, when a different player enters the threshold distance area of our user, we will attempt to subscribe to this other player.
5. We call the JSLIB "SubscribeWhenNear" plugin method.
6. This method translates the call from C to Javascript and calls the browser's twin method.
7. The browser checks if the user is already been subscribed to and if not, subscribes to the other user's track.
8. This results in their audio and video stream being added to the RTC window within the web page.
9. When the other user leaves the threshold vicinity area of our user, we do the reverse, calling the "UnsubscribeWhenFar" plugin method.
10. This method also translates the call from C to Javascript and calls the browser's twin method.
11. The browser receives this request and removes the other user from our subscribed users array, and removes their video player and audio stream from the RTC window.

6.1.4 Path Finding Algorithm

This is the algorithm used to traverse the gamespace using the mouse input, automatically guiding the player character through the game space, avoiding impassable terrain and finding the shortest path to the destination.

1. Initially, we create a matrix of the map, in the form of cubes that represent the map as a graph.

2. Each node is assigned a weight to be used for cost calculations. Objects and tiles tagged as "Obstacles" are given a cost of infinity.

3. When the user clicks on the map, we call the A* shortest path algorithm to calculate the shortest path from the player's current coordinate node in the graph, to the node where the mouse clicked.

4. It returns a vector list of path coordinates that start at the current location and ends at the destination.

5. This vector "stack" is fed to our custom Player Movement Controller that decodes it and moves the player character through this path.

6. Upon reaching the end of the vector stack, we will have arrived at the destination.

6.1.5 World Decoration System

This system allows you to place objects within the gamespace, and this object will be replicated on all user clients in the room. The items persist through sessions as well.

1. The user can click on the Place Item Button to open up the Decoration Window.

2. The decoration window is populated with all the placable Game Objects.

3. When a player selects an object, the System enters the "Placement Mode" state loop:

   (a) Create a duplicate "Ghost" object from the prefab ONCE and set its opacity to 0.5f.

   (b) Update this Ghost Object's position to that of Mouse cursor world coordinates.

   (c) If mouse is further than "Maximum Placement Distance" from the player OR the placement coordinate is not over a "Placable" tilemap layer, set the Ghost Object's sprite to RGB(255, 100, 100) [Red Tint] and set canBePlaced to false.

   (d) If mouse is closer than "Maximum Placement Distance" AND the placement coordinate is over a "Placable" tilemap layer, reset the Ghost sprite to default RGB(255, 255, 255) and set canBePlaced to true.

   (e) If user clicks the left mouse button, and the canBePlaced flag is true, do the Place Object routine:

      i. Create the object from the prefab and set its coordinate to mouse world position.

      ii. Call Photon's InstantiateWorldObject function to replicate this object on all other clients.

      iii. Reset the Ghost object being rendered over the mouse to null

      iv. Store the coordinate and metadata with the Firestore handler.

   (f) If user clicks the left mouse button and the canBePlaced flag is false, alert the user that the item cannot be placed here.

6.1.6 Text Chat System

Players can communicate with each other regardless of where they are within the room. They can use the chat window to send messages.

1. Read the value within the chat input field.
2. Package and send this string as an RPC call with the destination being all users in the room.
3. Users recieve the RPC call and trigger their local recieveMessage Function. They unpack the message, add a timestamp and formatting, and append the text to the chat body.

6.1.7 Resubscription Logic

We encountered an issue of users not resubscribing to the user that unmuted their microphone. To solve this issue, we developed an additional Re subscription method:

1. When a user clicks on the unmute button or leaves a private space, they call an additional "TriggerSubscribeForOtherPlayers" remote procedure call, which is sent to ALL users in the server.
2. Each user calls their local Proximity Chat module subscription method, which scans their surrounding for users.
3. This RPC calls subscribe on all the users regardless of whether they are already subscribed or not.
4. The browser checks if it is already subscribed to each user, and if not, subscribes to the user.

6.1.8 Track Swapping for Screen Sharing

A key limitation of Agora is that a client can only publish one video track at a time , either the user's video or his/her screen. So some logic should be in place to handle the overall publish - unpublish sequence that needs to be followed to handle the events from the beginning to the end of a screen share operation.

1. Stop playing the user-camera track and close it.
2. Unpublish the user-camera track.
3. Initialize the user-screen track.
4. Play the screen track on the appropriate container.
5. Publish the screen track so that other users can subscribe to it.
6. Check if the user has stopped screen share using the 'on-track-ended' event listener.
7. Stop playing the user-screen track and close it.
8. Unpublish the user-screen track.
9. Initialize the user-camera track.
10. Play the camera track on the appropriate container.

6.1.9 Fatigue Tracking

The fatigue tracking feature is based on facial feature extraction using Computer Vision, where behaviors such as eye closure and yawning duration are analyzed to determine whether a person is indicating signs of tiredness and take the appropriate action.

Inputs : User Video recorded via WebCam
Output : Window Alert and Audio Alert depending on threshold
Modules : OpenCV, dlib, imutils

1. Extract the frames from the video snippet captured by the webcam.
2. Extract HOG (Histogram of Oriented Gradients) descriptors from each frame to train the model using a linear SVM (Support Vector Machine) to detect and obtain the facial region.
3. Localize the face first and then detect the facial features on the region of interest by using the facial landmark detector available in the dlib module.
4. Pass the pre-processed grayscale image and number of pyramid layers as input to the facial landmark detector.
5. Extract left eye, right eye and mouth features using the indices.
6. Observe Eye Aspect Ratio (EAR) and Mouth Opening Ratio (MOR) to check for signs of fatigue with regard to the acceptable thresholds.
7. Compute the weighted average of EAR and MOR and alert the user by issuing a system wide alert when this average exceeds the acceptable threshold.

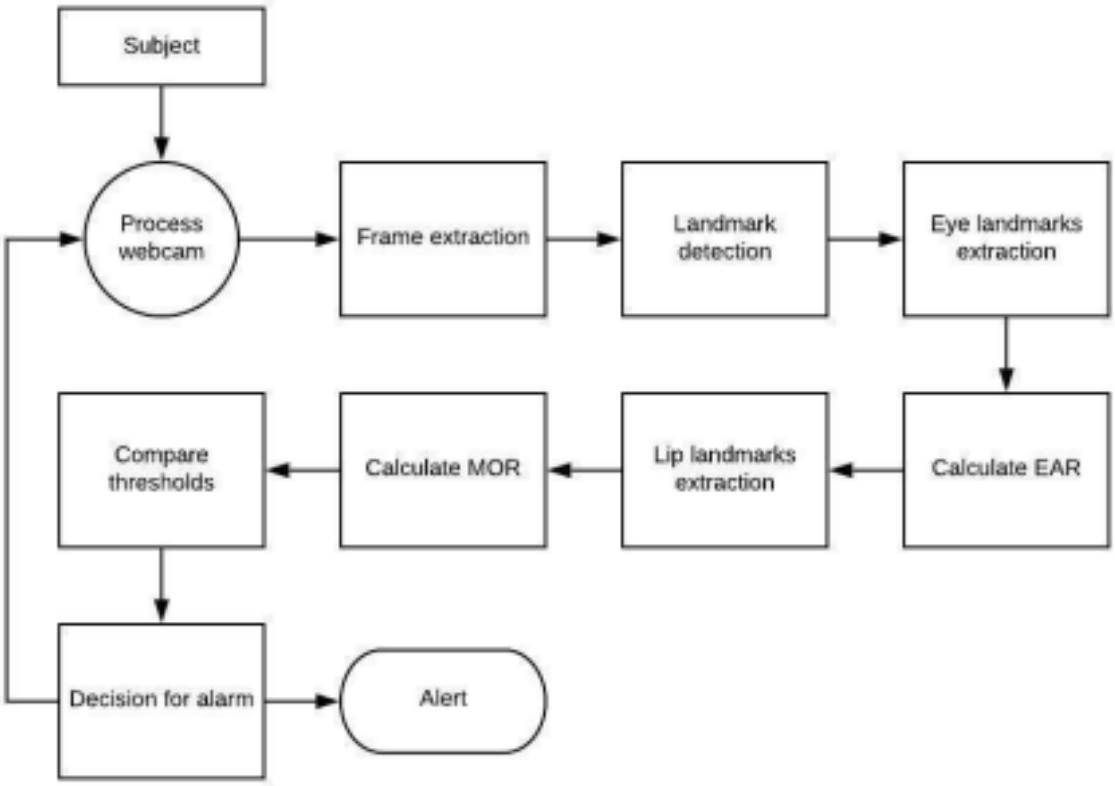


Figure 6.1: Fatigue Detection System

6.1.10 Reminder Algorithm

A reminder and clock will allow the user to set up a reminder. It will notify the user when the time comes.

Inputs : Time entered by the user.
Output : Window alert.

1. User input the time.
2. Every second it compares the current time with the reminder time.
3. If the current time equals the reminder time then the window alert will popup.

6.2 Development Tools

1. **IDE: VSCode -** Visual Studio Code is a free source-code editor made by Microsoft for Windows, Linux and macOS. Features include support for debugging, syntax highlighting,intelligent code completion, snippets, code refactoring, and embedded Git.

2. **Version Control: GitHub -** A distributed version-control system for tracking changes insource code during software development.

3. **Testing: live-server -** A npm (Node Package Manager) based little development server for testing out web apps on local host.

4. **Testing: ngrok -** A cross-platform application that exposes local server ports to the Internet.

5. **Testing: Web server for Chrome -** A simple chrome extension that allows you to create a development server for testing out web apps on local host

6. **Unity**: The game creation software that provides a suite of products to aid in game development, including the robust cross platform Unity game engine.

7. **Piskell**: It is a free online editor for animated sprites pixel art.

Chapter 7

Testing

For testing this project, we primarily looked at how the various components in each module behaved for multiple scenarios and checked if the overall behaviour was consistent with our expectations during the various phases of development.

7.1 Testing Methodologies

Software testing methodology is for making sure that software products/systems developed have been successfully tested to meet their specified requirements and can successfully operate in all the anticipated environments with required usability and security. Software testing methods are traditionally divided into white and black box testing. These two approaches are used to describe the point of view that a test engineer takes when designing test cases. White box testing by seeing the source code tests internal structures or workings of a program, as exposed to the functionality exposed to the end-user. In white-box testing an internal perspective of the system, as well as programming skills, are used to design test cases. The tester chooses inputs to exercise paths through the code and determine the appropriate outputs. This is analogous to testing nodes in a circuit. While white-box testing can be applied at the unit, integration and system levels of the software testing process, it is usually done at the unit level. Black box testing treats the software as a black-box, examining functionality without any knowledge of internal implementation, without seeing the source code. The testers are only aware of what the software is supposed to do, not how it does it. The testing methods applied were:

1. Unit Testing
2. Integration Testing
3. System Testing

7.2 Unit Testing

Unit testing involves designing test cases to validate that the internal program logic is functioning properly, and the inputs provided produces valid output. All decision branches and internal code flow are validated. All individual modules are tested separately. It ensures that each module performs accurately to the specification and has well defined inputs and outputs.

7.2.1 Gamespace

Test 1.1

Description: Tried Moving player through gamespace with input keys.
Result: Pass

Test 1.2

Description: Tried connecting to Photon room instance.
Result: Pass

Test 1.3

Description: Tried synchronising player names across all clients
Result: Fail

Test 1.4

Description: Tried synchronising player names across all clients
Result: Pass

Test 1.5

Description: Tried implementing text chat in game.
Result: Pass

Test 1.6

Description: Tried placing custom decoration in the office and synchronise with others.
Result: Fail

Test 1.7

Description: Tried placing custom decoration in the office and synchronise with others.
Result: Pass

Test 1.8

Description: Tested proximity chat logic with multiple players.
Result: Pass

Test 1.9

Description: Tested exiting private space and checking if others resubscribe
Result: Fail

7.2.2 Communication Module

Test 2.1

Description: Tried to start a video call locally (Agora v3.6)
Result: Pass

Test 2.2

Description: Tried out mute and unmute operations locally (Agora v3.6)
Result: Pass

Test 2.3

Description: Tried out screen-share operations locally.(Agora v3.6)
Result: Fail

Test 2.4

Description: Tried out screen-share operations locally.(Agora v3.6)
Result: Pass

Test 2.5

Description: Tried out a group video call via Ngrok.(Agora v3.6)
Result: Fail

Test 2.6

Description: Tried out a group video call via Ngrok after modifying subscription logic.(Agora v3.6)
Result: Fail

Test 2.7

Description: Tried out a group video call via Ngrok after trying out a different initialization logic.(Agora v3.6)
Result: Fail

Test 2.8

Description: Migration to Agora-NG (v4.X)
Result: Pass

Test 2.9

Description: Tried out a group video call via Ngrok.(Agora v4.X)
Result: Pass

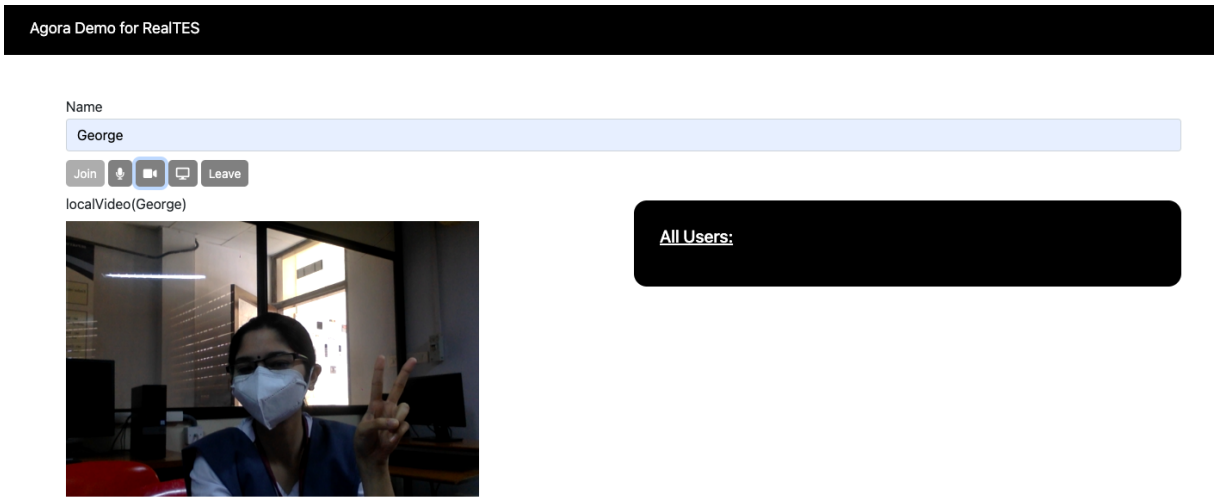


Figure 7.1: Agora Video Call Testbed

Test 2.10

Description: Tried out a delayed-subscription logic for Spatial Audio.(Agora v4.X)
Result: Fail

Test 2.11

Description: Trying out a delayed-subscription logic for Spatial Audio by modifying event-listener actions.(Agora v4.X)
Result: Pass , Re-subscription to be initiated through Unity.

7.2.3 Productivity Suite

Test 3.1

Module: Fatigue Tracking
Description: Tried out the Fatigue Tracking system on multiple subjects for estimating accuracy.
Result: Pass

Test 3.2

Module: Fatigue Tracking
Description: Introduced Audio alerts to the Fatigue Tracking System via gtts (Google Text-to-Speech)
Result: Pass

Test 3.3

Module: Fatigue Tracking
Description: Tried creating a single executable file that encompasses the entire Fatigue Tracking

System using py2exe.
Result: Fail

Test 3.4

Module: Fatigue Tracking
Description: Tried creating a single executable file that encompasses the entire Fatigue Tracking System using auto-py-to-exe.
Result: Pass

Test 3.5

Module: Time Management Tools
Description: Trying to set up the reminder and clock
Result: Fail

Figure 7.2: Clock Image

Test 3.6

Module: Time Management Tools
Description: Trying to set up the Clock
Result: Pass

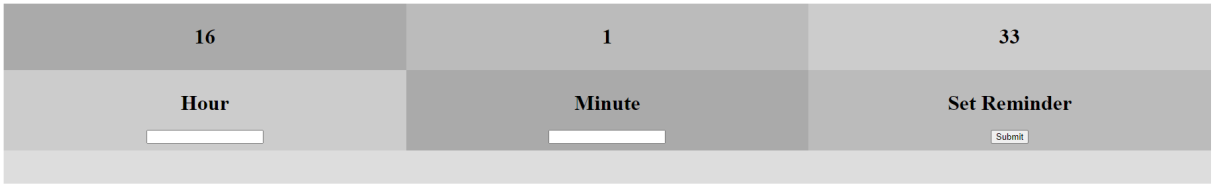


Figure 7.3: Clock image

Test 3.7

Module: Time Management Tools
Description: Trying to set up the reminder.
Result: Pass

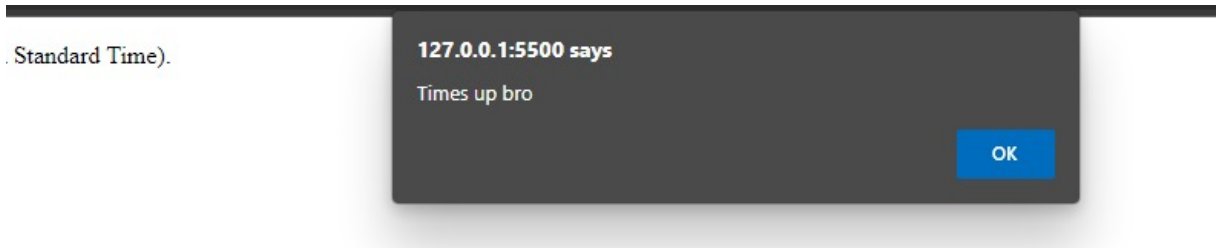


Figure 7.4: Reminder image

7.2.4 JSLib Module

Test 1.1

Description: Checks whether JSLib function GetDocument fetches document of given ID from the specified collection, for all collections in Firestore.
Result: Pass

Test 1.1

Description: Checks whether JSLib function SetDocument that updates document of given ID from the specified collection, for all collections in Firestore.
Result: Pass

Test 1.1

Description: Checks whether JSLib function AddElementInArrayField adds element to a particular field of specified document in a collection.
Result: Pass

Test 1.1

Description: Checks whether JSLib function AddElementInArrayField adds element to a particular field of specified document in a collection.
Result: Pass

Test 1.1

Description: Checks whether JSLib function RemoveElementInArrayField removes an element from a particular field of specified document in a collection.
Result: Pass

Test 1.1

Description: Checks whether JSLib function UpdateDocument updates the specified document in a collection.
Result: Pass

Test 1.1

Description: Checks whether JSLib function DeleteDocument deletes the specified document in a collection.
Result: Pass

7.3 Integration Testing

In Integration Testing, different modules were combined and tested to see if the modules interact properly and produce the correct output. Tests were performed after the integration of the following modules.

1. Unity - Photon
2. Unity - Firestore (via jslib)
3. Unity - Agora (via jslib)

7.3.1 Unity - Photon

Integrating the Photon Module into Unity was mostly straightforward, with the Photon Unity Networking (PUN) in the asset store. This opened up special scripts that could be used with the game objects. For instance, the Photon Transform view would help synchronise the position of the player object in the network. The Photon RB script will help synchronise the physics engine aspect of the game engine, when attached to it as a component.

7.3.2 Unity - Firestore

Initially, the plan was to use Firestire Unity SDK for accessing Firestore service from Unity. Later it was realized that it Firestore Unity SDK does not work with WebGL builds, hence requiring a custom interface between Unity and Firestore SDK for web. This interface is a plugin written in JSLib native Javascript library functions that unity can call. Unity-Firestore tests involved writing JSLib and firing queries for all CRUD operations from unity via JSLib and verifying the responses.

7.3.3 Unity - Agora

The Game Engine module (Unity) and the Communication Module (Agora) were developed independently at the beginning. During the requirements gathering and design specification process, the idea was to use the Agora SDK specifically suited towards building Unity apps,however, owing to RealTES' browser first approach, it was realized that the Agora components would be incompatible with the WebGL based rendering of the Unity app, hence necessitating a switch to the Agora SDK for Web. This change required a bridge so that the Agora based methods (JavaScript) could be called from the Unity app (C#). Enter jslib which facilitated communication between the Unity and Agora based modules.

This involved tests for the following functionalities.

1. Mapping of Unity (Photon) player to the equivalent Agora Player.

2. jslib - Agora interactions.
3. Unity - jslib interactions.
4. Synchronization of player states in Unity and Agora to implement spatial audio.
5. Position based subscription and re-subscription. (Collider logic in Unity, event-listener logic in Agora.)
6. Compatibility and responsiveness of web based components corresponding to the Unity and Agora modules.

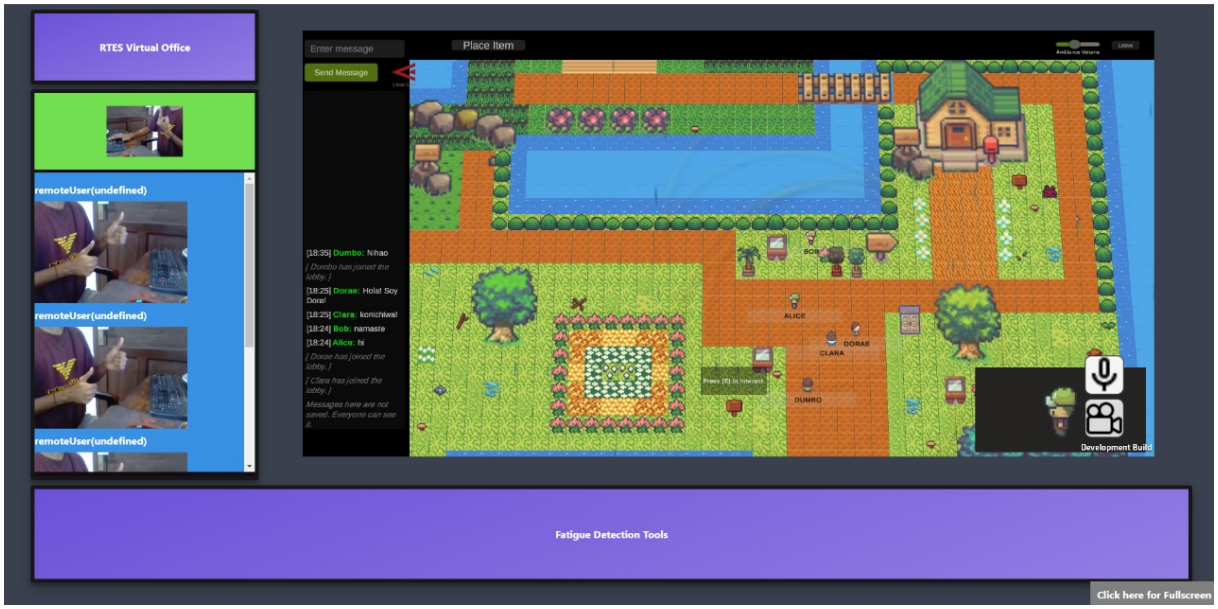


Figure 7.5: Testing out the Integration of the Unity and Agora based modules.
(Early Beta Version)

7.4 System Testing

In system testing all the modules are integrated in order of execution. Testing is carried out over the whole system. All the events starting with joining a room, walking in the gamespace, audio and video communication etc till the user leaving the gamespace was examined.

A few kinks due to network related issues like ghost players, track publishing delay etc were noticed and sufficient guards were implemented to compensate for these out of hand issues in the best way possible.

The overall event flow was re-examined and the results were deemed satisfactory.

Figure 7.6: Fully integrated system

Chapter 8

Graphical User Interface

The graphical user interface is a user interface that allows users to interact with electronic devices through graphical icons and visual indicators such as secondary notations, instead of text-based user interfaces, typed command labels or text navigation.GUIs were introduced in reaction to the perceived steep learning curve of command-interfaces which require a command to be typed on a computer keyboard. Designing the visual composition and temporal behaviour of a GUI is an important part of software application programming in the area of human-computer interaction. Its goal is to enhance the efficiency and use case for the underlying logical design of a stored program, a designed discipline name usability. Methods for user-centred design are used to ensure that the visual language introduced in the design is well-tailored to the tasks. For several decades GUIs were controlled exclusively by a mouse and keyboard. While these types of input devices are sufficient for desktop computers, they do not work as well for mobile devices, such as smartphones and tablets. Therefore, mobile operating systems are designed to use a touch screen. Many mobile devices can now be controlled by spoken commands as well. Because there are now many types of digital devices available, GUIs must be designed for the appropriate type of input. For example, a desktop operating system, such as OS X, includes a menu bar and windows with small icons that can be easily navigated using a mouse. A mobile OS, like iOS, includes large icons and supports touch commands like swiping and pinching to zoom in or zoom out. Automotive interfaces are often designed to be controlled with knobs and buttons and TV interfaces are built to work with remote control. Regardless of the type of input each of these interfaces are considered GUIs since they include graphical elements.

8.1 GUI Overview

RealTES GUI can be divided into 3 sections:

- Agora container - A side panel containing video containers for local and remote video stream as well as screen share functionality
- Unity container - The main container encapsulating Unity gamespace and chat functionality
- Productivity tools - Bottom panel containing buttons for enabling fatigue detection and setting alerts and reminders

Figure 8.1: Webpage

8.2 Main GUI Components

The main unity container is the entry point to RealTES. A user joins an office/game space as in Figure 8.2. Figure 8.3 shows the gamespace with chat functionality and audio/video controls. The side panel to the left contains remote and local video streams as shown in Figure 8.4, that are selectively added added and removed based on distance vectoring. The bottom panel contains buttons for fatigue tracking, alerts, reminders and clock.

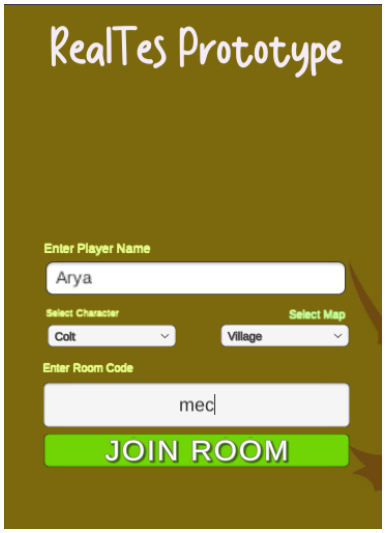


Figure 8.2: Join office/game space

Figure 8.3: Unity container with gamespace and chat functionality

Figure 8.4: Agora video panels

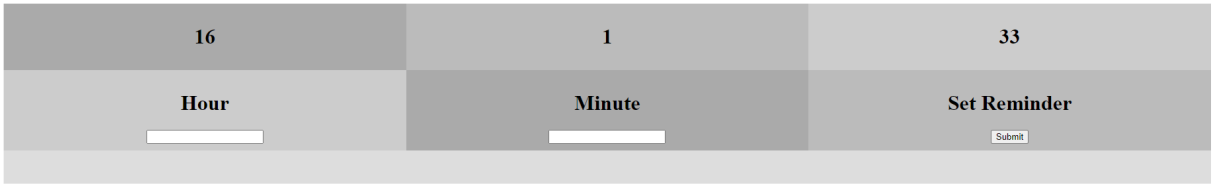


Figure 8.5: Clock

Chapter 9

Results

We were able to implement a real-time multi user chat and video calling web application with a gamified representation of a real world environment in a 2-D space which includes the following features:

1. Private Spaces
2. Path Finding Algorithm
3. Multiple Gamespaces
4. Gamespace customization
5. Spatial Audio/Video
6. Screen Sharing
7. Fatigue Tracking
8. Clock, reminders and alerts

9.1 Gamespace

9.1.1 Village GameSpace

Figure 9.1: Village GameSpace

9.1.2 Village GameSpace/Japanese House

Figure 9.2: Village GameSpace/Japanese House

9.1.3 Village GameSpace/Internal Auditorium

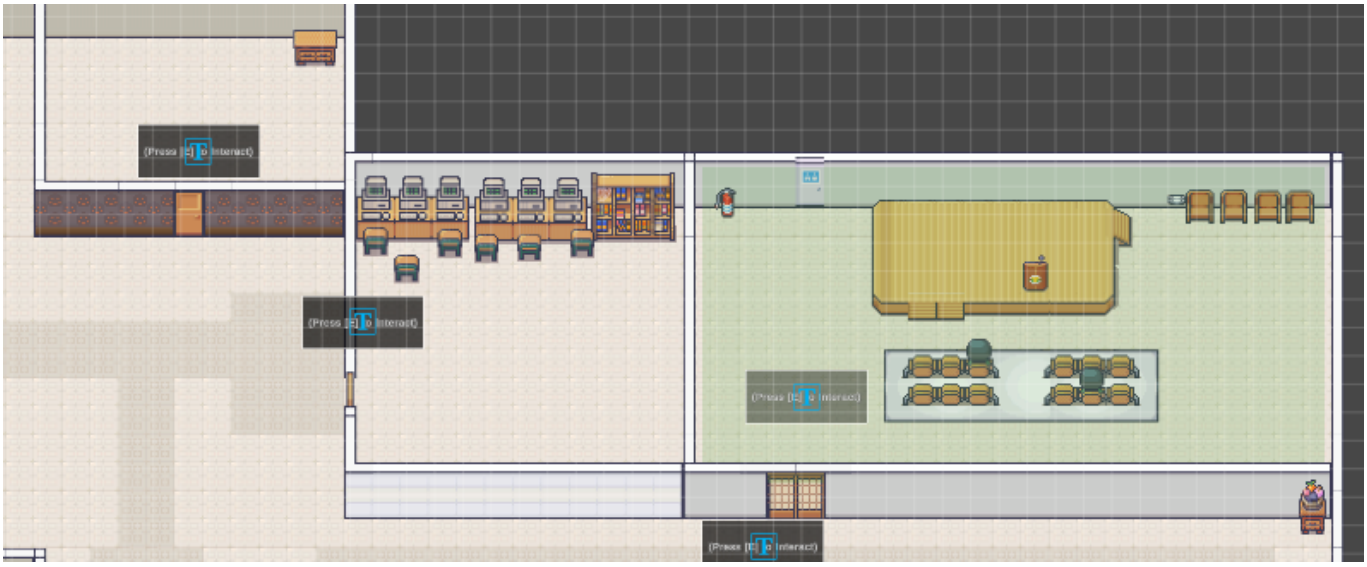


Figure 9.3: Village GameSpace/Internal Auditorium

9.1.4 Village GameSpace/Library

Figure 9.4: Village GameSpace/Library

9.1.5 Village GameSpace/Character

Figure 9.5: Village GameSpace/Character

9.1.6 MEC GameSpace/Exterior

Figure 9.6: MEC GameSpace/Exterior

9.1.7 MEC GameSpace/Interior

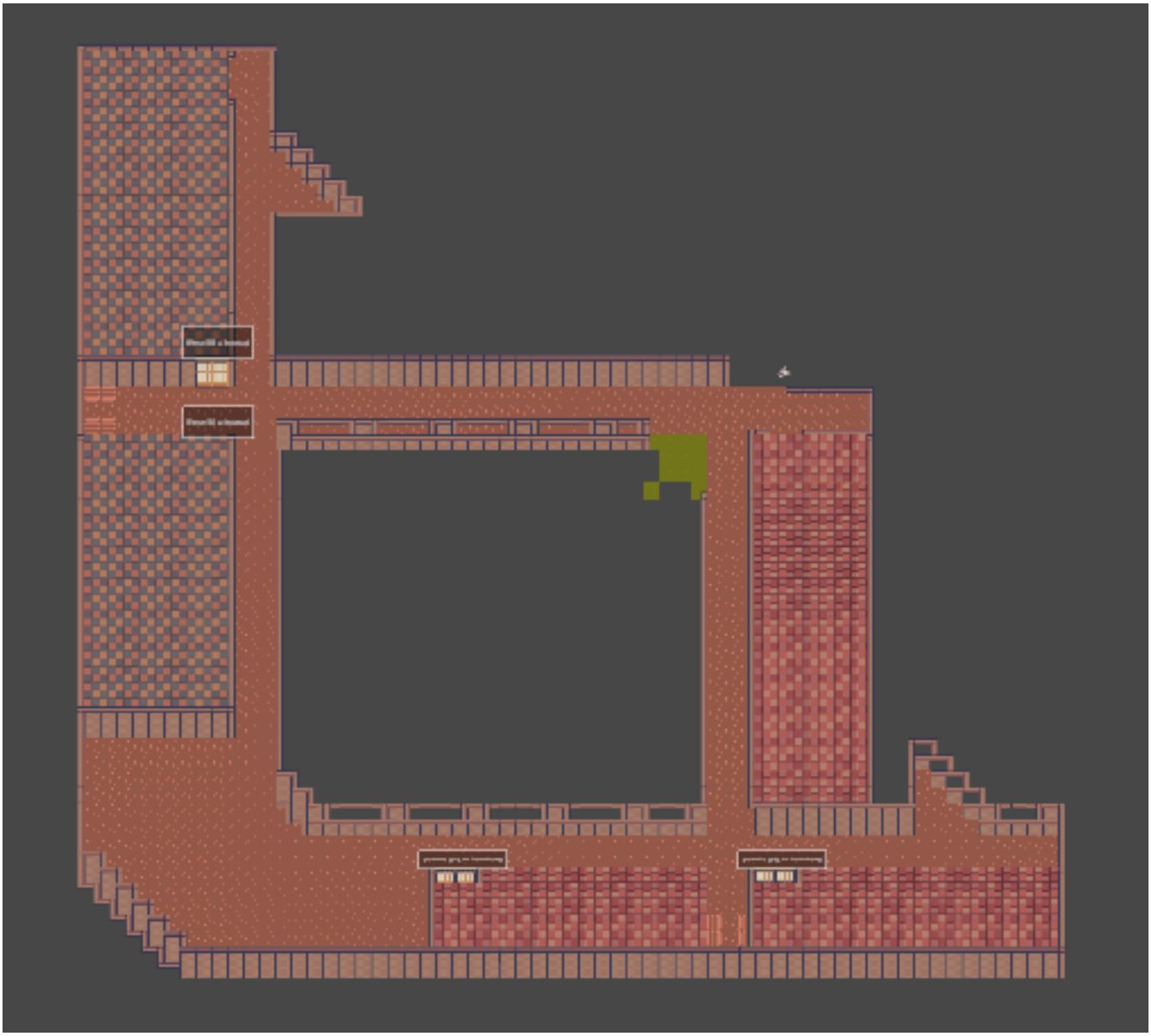


Figure 9.7: MEC GameSpace/Interior

9.1.8 MEC GameSpace/Classroom

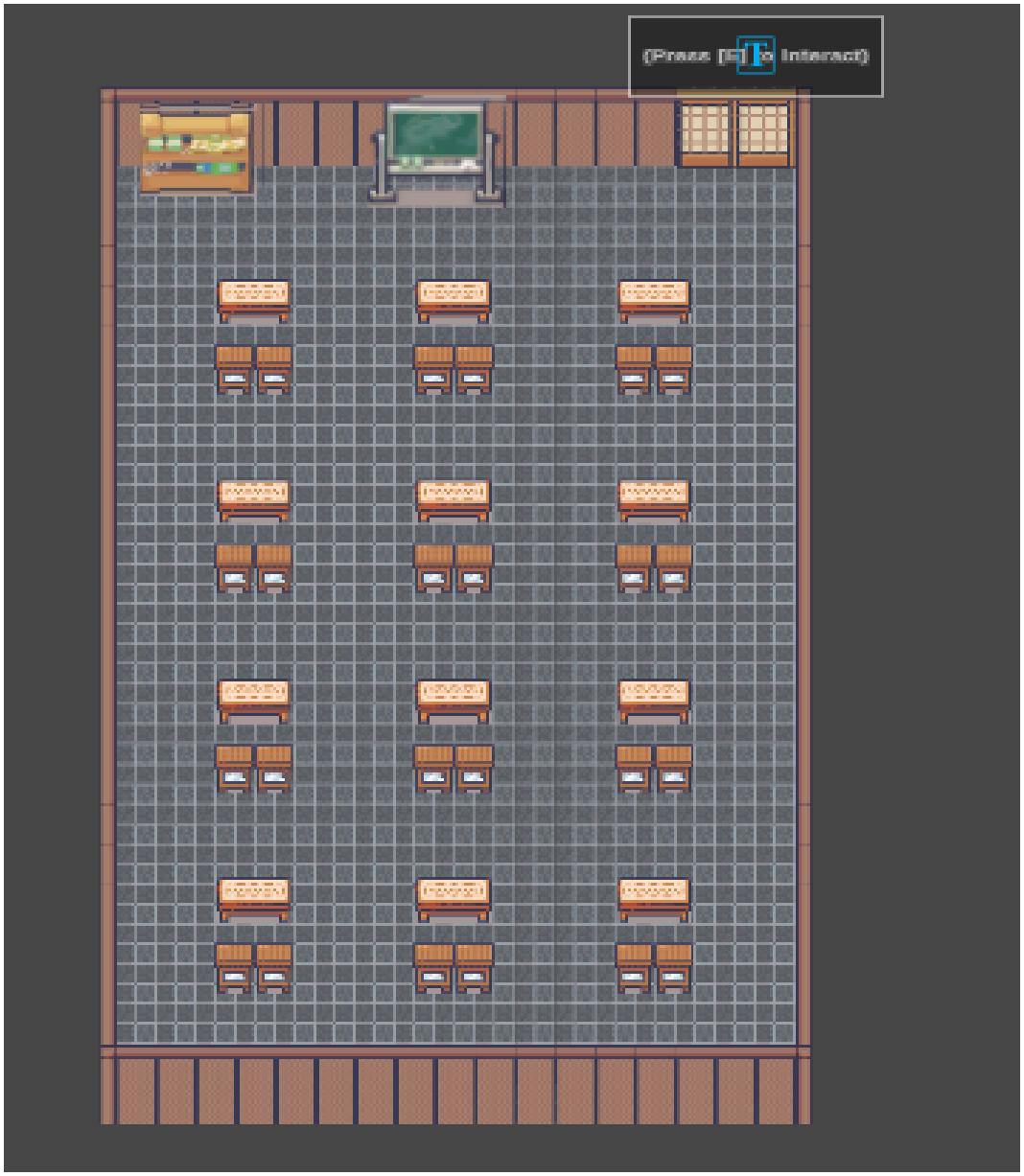


Figure 9.8: MEC GameSpace/Classroom

9.1.9 Welcome Message

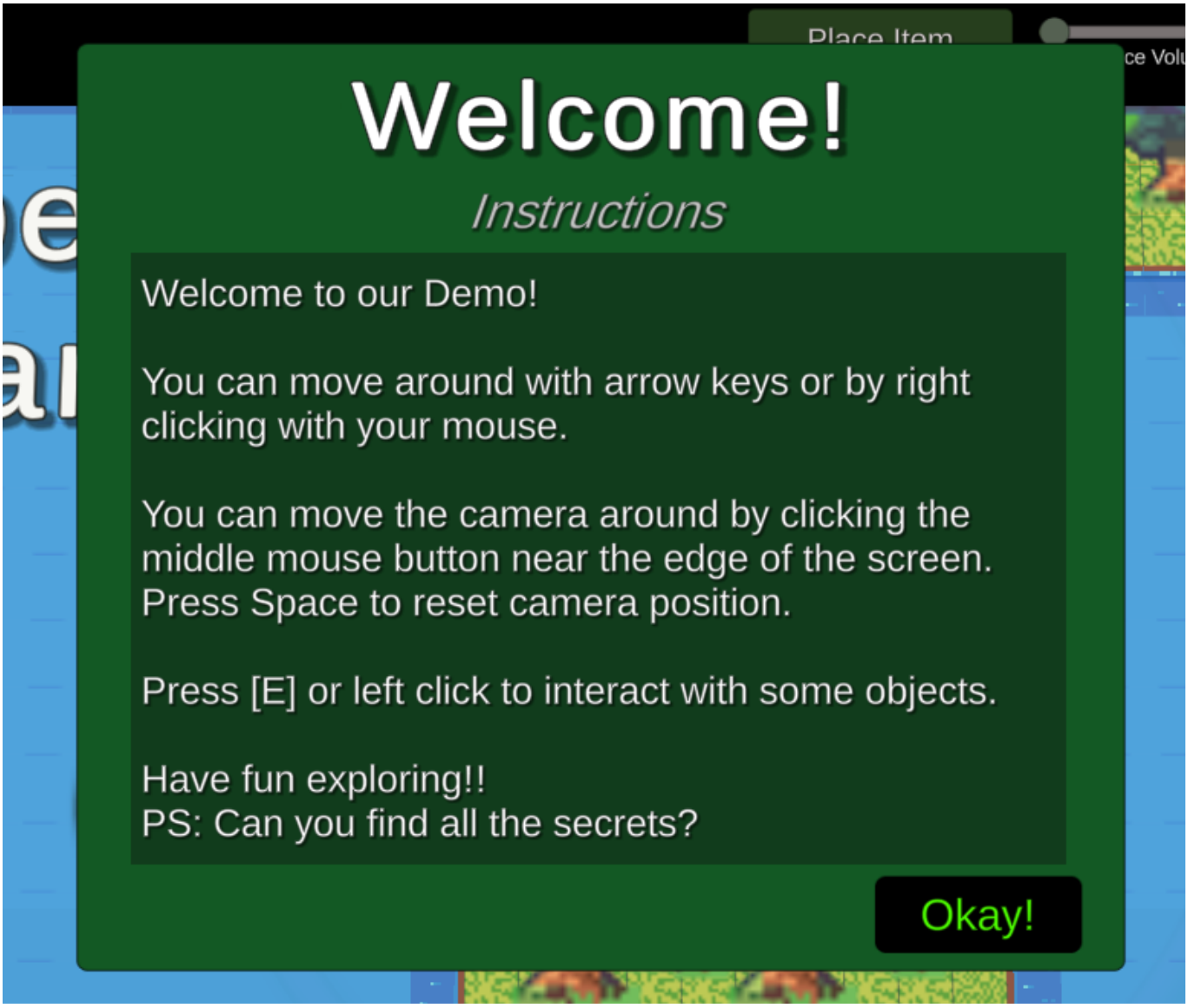


Figure 9.9: Welcome Message

9.1.10 Video and Chat

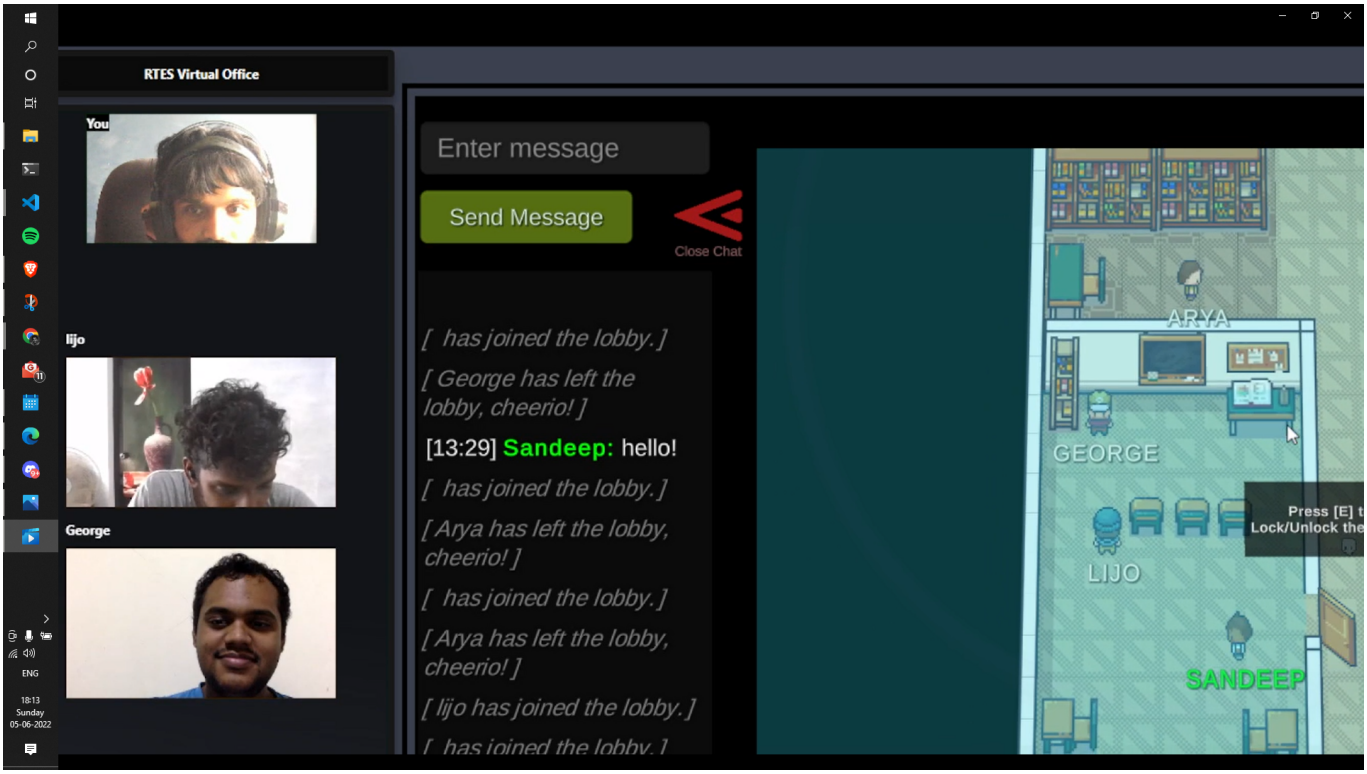


Figure 9.10: Video and Chat

9.1.11 Screen Sharing

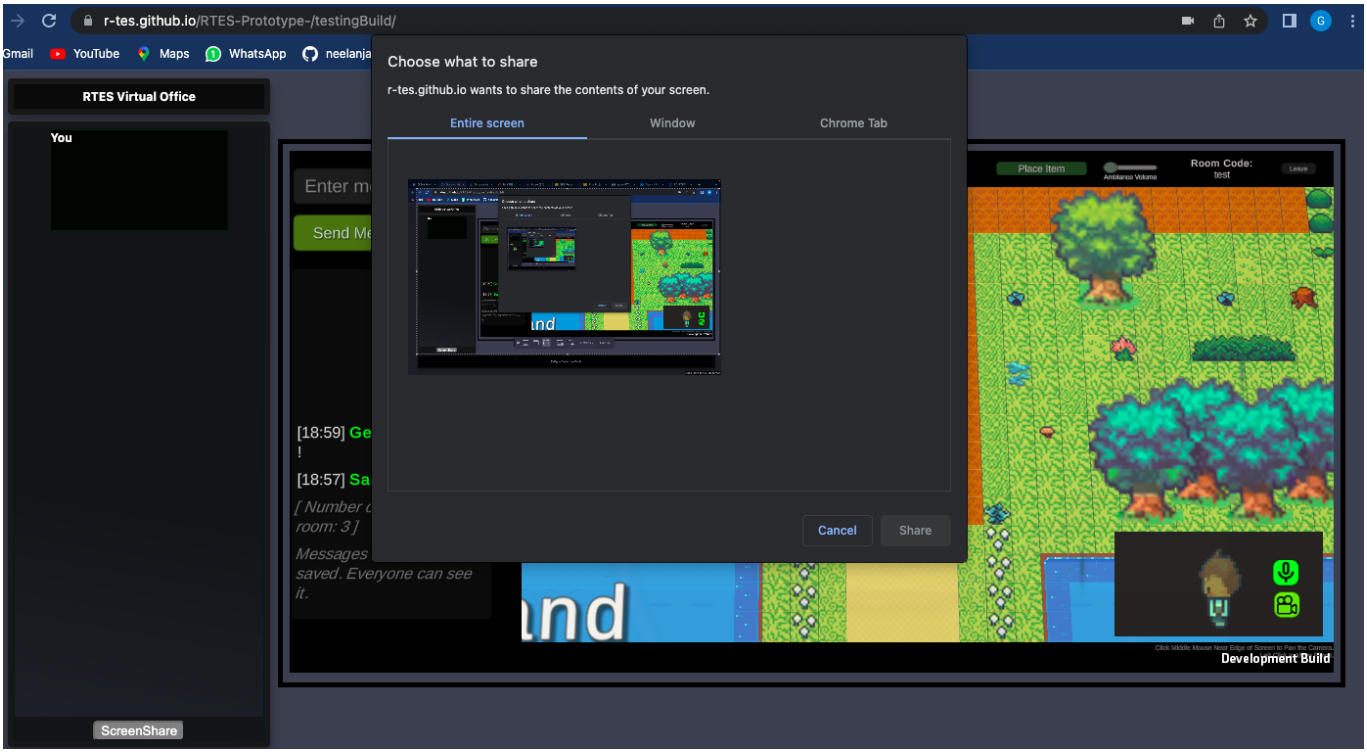


Figure 9.11: Screen Sharing

9.1.12 Object Placer

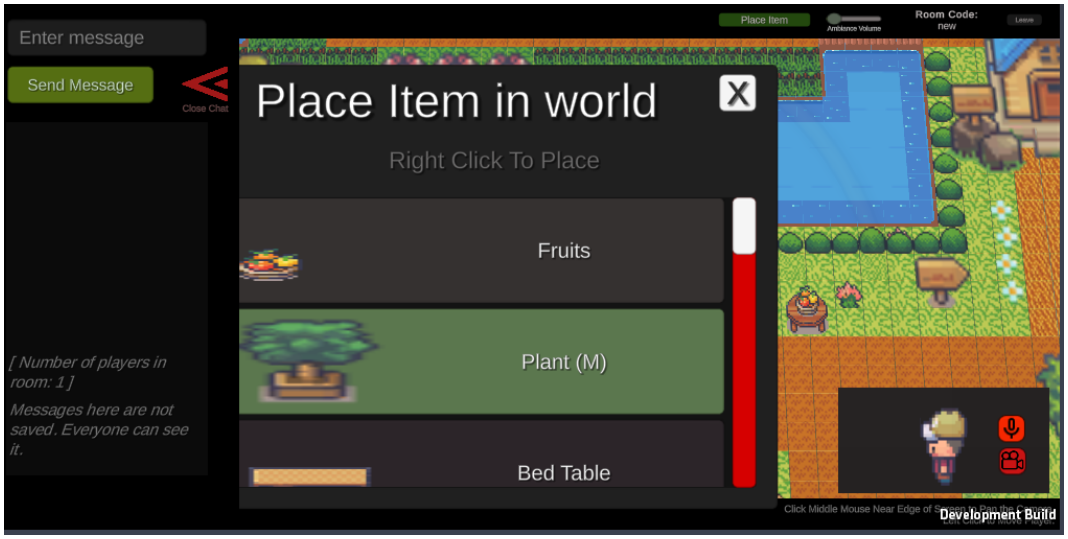


Figure 9.12: Object Placer

Figure 9.13: Placed Object

9.1.13 Object Tracker

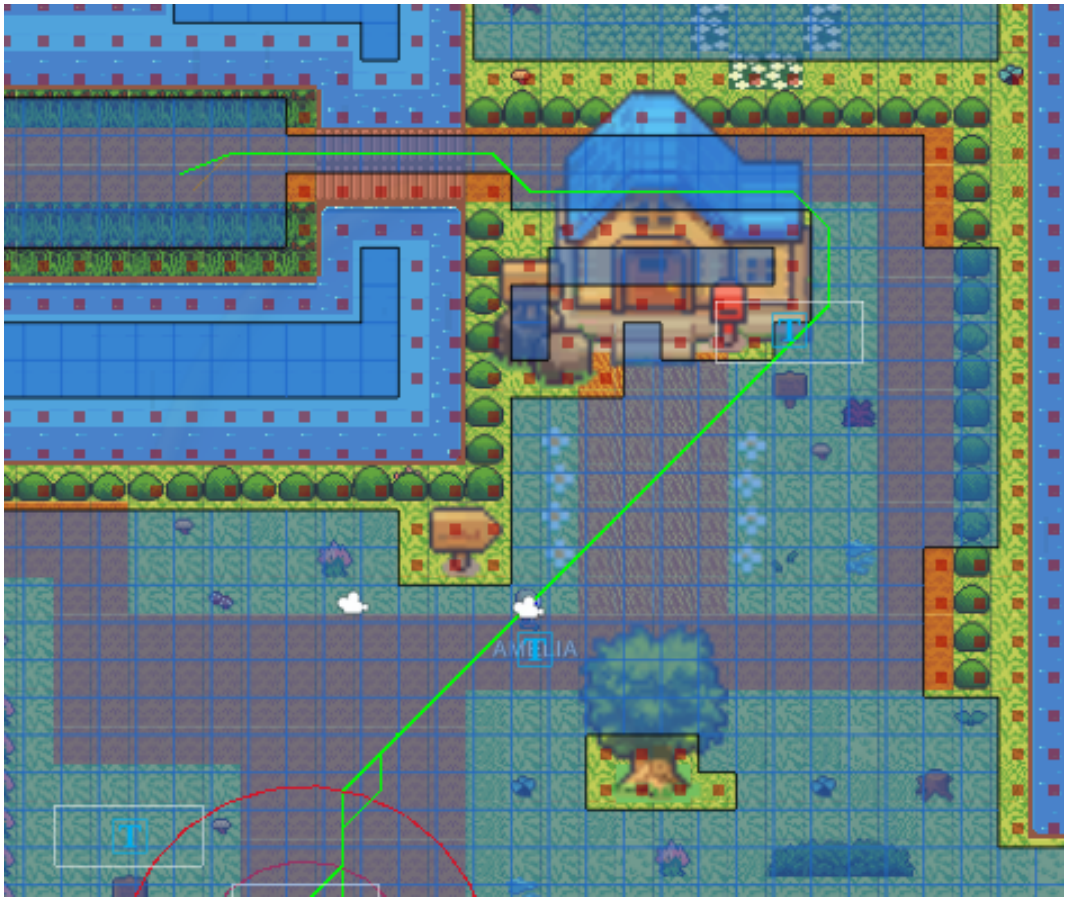


Figure 9.14: Object Tracker

9.1.14 Private Spaces

Figure 9.15: Out Side Of Private Space

Figure 9.16: Inside Side Of Private Space

Chapter 10

Conclusion

Should there ever be another situation where people would once again be confined to working/learning virtually , we hope RealTES as a platform would be the go to choice for many because :

1. It best represents the essence of a real world environment and comes really close to mimicking the fundamental characteristics of an actual human conversation.
2. It helps users improve their productivity through the usage of time management and fatigue tracking tools.

Chapter 11

Future Scope

We believe that we have a solid offering in RealTES that can stand toe-to-toe with platforms like Google Meet, Gather.town etc as RealTES offers a breath of fresh air in terms of bringing out what we all love about face to face conversations. The productivity suite helps RealTES go one step further when compared to similar offerings as it helps address falling productivity and emphasizes on keeping users healthy and happy.

We strongly feel that RealTES has the capability to be the preferred choice for remote work or learning and believe that it's true potential lies beyond the scope we've currently defined for it.

The following represents some of the areas that are worth exploring that could aid in the commercial use of RealTES.

1. Allow users to create their own custom avatars instead of having to choose from the presets.
2. Allow for the gamespace to be modified, with additional rooms and areas as per the user's choosing.
3. Adding more features like post-it-notes, whiteboards etc within the gamespace.
4. Setting up a token server for Agora so that features like host and co-host can be implemented.
5. Giving the user the option to blur their background or change the background during a video call.
6. Creating an all in one executable that can run the Fatigue Tracking system on Mac/Linux devices.
7. Allow users to schedule multiple reminders.

References

[1] Suciu, George; Stefanescu, Stefan; Beceanu, Cristian; Ceaparu, Marian (2020). [IEEE 2020 Global Internet of Things Summit (GIoTS) - Dublin, Ireland (2020.6.3-2020.6.3)] 2020 Global Internet of Things Summit (GIoTS) - WebRTC role in real-time communication and video conferencing. , (), 1–6. doi:10.1109/GIOTS49054.2020.9119656

[2] Jansen, Bart; Goodwin, Timothy; Gupta, Varun; Kuipers, Fernando; Zussman, Gil (2018). Performance Evaluation of WebRTC-based Video Conferencing. ACM SIGMETRICS Performance Evaluation Review, 45(2), 56–68.

[3] Mu, Bo; Yang, YuHui; Zhang, JianPing (2009). [IEEE 2009 International Conference on Information Technology and Computer Science (ITCS 2009) - Kiev, Ukraine (2009.07.25-2009.07.26)] 2009 International Conference on Information Technology and Computer Science - Implementation of the Interactive Gestures of Virtual Avatar Based on a Multi-user Virtual Learning Environment. , (), 613–617.doi:10.1109/itcs.2009.134

[4] Sag, Antonio; Orehovacki, Tihomir (2019). [IEEE 2019 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO) - Opatija, Croatia (2019.5.20-2019.5.24)] 2019 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO) - Development of 2D Game with Construct 2. , (), 1647–1652.doi:10.23919/mipro.2019.8756702

[5] Xie, Jingming (2012). [IEEE 2012 7th International Conference on Computer Science Education (ICCSE 2012) - Melbourne, Australia (2012.07.14-2012.07.17)] 2012 7th International Conference on Computer Science Education (ICCSE) - Research on key technologies base Unity3D game engine. , (), 695–699. doi:10.1109/iccse.2012.6295169

[6] Sheng, Wang; Nakata, Susumu; Tanaka, Satoshi; Tanaka, Hiromi H.; Tsukamoto, Akihiro (2013). [IEEE 2013 International Conference on Culture and Computing (Culture Computing) - Kyoto, Japan (2013.09.16-2013.09.18)] 2013 International Conference on Culture and Computing - Modeling High-Quality and Game-Like Virtual Space of a Court Noble House by Using 3D Game Engine. , (), 212–213. doi:10.1109/culturecomputing.2013.70

[7] Cowan, Brent; Kapralos, Bill (2014). [IEEE 2014 IEEE 14th International Conference on Advanced Learning Technologies (ICALT) - Athens, Greece (2014.7.7-2014.7.10)] 2014 IEEE 14th International Conference on Advanced Learning Technologies - A Survey of Frameworks and Game Engines for Serious Game Development. , (), 662–664. doi:10.1109/icalt.2014.194

[8] Tate, A. (2011). I-Room: Augmenting Virtual Worlds with Intelligent Systems. , 15(5), 56–61. doi:10.1109/mic.2011.71

[9] Garg, Hitendra (2020). [IEEE 2020 International Conference on Power Electronics IoT Applications in Renewable Energy and its Control (PARC) - Mathura, Uttar Pradesh, India (2020.2.28-2020.2.29)] 2020 International Conference on Power Electronics IoT Applications in Renewable Energy and its Control (PARC) - Drowsiness Detection of a Driver using Conventional Computer Vision Application. , (), 50–53. doi:10.1109/PARC49193.2020.236556

[10] Rincon-Nigro, M.; Zhigang Deng, (2013). A Text-Driven Conversational Avatar Interface for Instant Messaging on Mobile Devices. IEEE Transactions on Human-Machine Systems, 43(3), 328–332.

[11] Unity - Manual : https://docs.unity3d.com/Manual/index.html

[12] Photon Unity Networking : https://doc.photonengine.com/en-us/pun/current/getting-started/pun-intro

[13] Agora Web SDK API Reference : https://docs.agora.io/en/video/API/20Reference/web-ng/index.html.

[14] Ramic-Brkic, Belma (2018). [IEEE 2018 10th International Conference on Virtual Worlds and Games for Serious Applications (VS-Games) - Würzburg, Germany (2018.9.5-2018.9.7)] 2018 10th International Conference on Virtual Worlds and Games for Serious Applications (VS-Games) - Enhancing Progressive Education through the Use of Serious Games. , (), 1–4. doi:10.1109/VS-Games.2018.8493422

[15] Voulgari, Iro; Komis, Vassilis (2011). [IEEE 2011 3rd International Conference on Games and Virtual Worlds for Serious Applications (VS-GAMES 2011) - Athens, Greece (2011.05.4-2011.05.6)] 2011 Third International Conference on Games and Virtual Worlds for Serious Applications - On Studying Collaborative Learning Interactions in Massively Multiplayer Online Games., (), 182–183. doi:10.1109/vs-games.2011.36

[16] Tomala-Gonzales, Jennifer; Guaman-Quinche, Jose; Guaman-Quinche, Edwin; Chamba-Zaragocin, Wilman; Mendoza-Betancourt, Silvana (2020). [IEEE 2020 15th Iberian Conference on Information Systems and Technologies (CISTI) - Sevilla, Spain (2020.6.24-2020.6.27)] 2020 15th Iberian Conference on Information Systems and Technologies (CISTI) - Serious Games: Review of methodologies and Games engines for their development., (), 1–6. doi:10.23919/CISTI49556.2020.9140827

[17] Sri Mounika, T.V.N.S.R., Phanindra, P.H., Sai Charan, N.V.V.N., Kranthi Kumar Reddy, Y., Govindu, S. (2022). Driver Drowsiness Detection Using Eye Aspect Ratio (EAR), Mouth Aspect Ratio (MAR), and Driver Distraction Using Head Pose Estimation. In: Tuba, M., Akashe, S., Joshi, A. (eds) ICT Systems and Sustainability. Lecture Notes in Networks and Systems, vol 321. Springer, Singapore. https://doi.org/10.1007/978-981-16-5987-4$_6$3