**The Time Machine Project Report**



**An Interactive History Learning Experience**

**Prepared by**

**Jairaj Shaktwat, Eric Heilman,**

**Mitul Patel, and Zachary Jones**

**University of Illinois Chicago**

**Spring 2017How to Use This Document**

This document is intended as a sample template that can be copied and edited to suit a particular software engineering project. It was assembled from a combination of documents [1], [2], and [3].

**Styles**

This document was written in Microsoft Word, and makes heavy use of styles. The styles dialog is initially located on the menu bar under the “Home” tab in MS Word. It is recommended that the styles dialog be pulled off into a separate window when working on formatting of the document. If each paragraph is assigned a style, then modifying that particular style will affect all paragraphs in the document having the same style.

The table of contents uses the document headings and sub-headings to automatically generate table of contents information.

**Tracking Changes and Multiple Authors**

The “Review” tab in MS Word contains several tools that are of particular use when editing large documents, particularly when multiple authors are involved:

The “Tracking” section allows you to track the ( proposed ) changes to a document, and to step through each proposed change to either accept or reject the proposed changes.

The “Compare” section allows you to merge changes proposed by different authors, ( which will be marked in separate colors for identification ), and then to use the change tracking tools described above to accept or deny each change.

The recommended procedure is to start with each author having a copy of a base document, ( possibly this template. ) Then each author changes the section(s) they are responsible for, and submits their changed version to one person who acts as the overall document editor. This author merges the changes, selectively accepts or rejects each change, and then distributes a new base document to all authors for the next round of changes. It is also possible to merge the changes and then distribute the document, so that all authors can review the proposed changes. ( The latter approach may be appropriate for documents such as bylaws, in which the changes must be approved by a committee or a vote before they can be accepted. )

**Dealing With Material that is Unwanted ( Right Now )**

Much of this document includes material that is not needed for every project, and/or which may not have been written yet, and so should be removed before printing or distributing the document. There are several ways to do this, however each has their drawbacks:

1. Delete the material completely. The drawback is that now it is completely gone, and the only way to get it back is to copy it from some other document, if that is even available.
2. Change the font to "Hidden". The drawback is that this does not affect the numbering of sections, either in the text or in the table of contents. However the original style information is retained, so when the text is unhidden, individual paragraphs do not need to be restyled.
3. Change the style to "Hidden". This does cause the document to renumber properly, ( because the paragraphs are no longer numbered paragraphs ), but all the original style information is gone, so if it is unhidden later, then all the styles of all the unhidden paragraphs will have to be restored manually.
4. Move the text, say to the end of the document, and then do not print or distribute the material at the end of the document. If this is all that is done, then this material will continue to appear in the table of contents. However if the font is also changed to "hidden", then the material will not appear in the TOC, but all the original style information will be retained. The drawback to moving the text is that now you need to keep track of where the information was moved from, in case you ever want to move it back.

Hidden text can be made visible by selecting the backwards "P" paragraph mark on the "Home" tab.

**Table of Contents**

The table of contents lists the sections of the document and the page upon which each section starts. The table of contents may or may not include subsections, etc.

Microsoft Word ( and many similar programs ) has the ability to generate a table of contents automatically from section headings, and to update it when the document changes. The table of contents included in this template is automatically generated by MS Word, and can be updated by selecting the table and clicking on the menu that appears above the table at that time.

**Note: Remove all instructional materials before finalizing and submitting this document, including this entire page and the one before it.**

**Table of Contents**

[How to Use This Document 2](#_Toc461441751)

[List of Figures 9](#_Toc461441752)

[List of Tables 10](#_Toc461441753)

[I Project Description 12](#_Toc461441754)

[1 Project Overview 12](#_Toc461441755)

[2 The Purpose of the Project 12](#_Toc461441756)

[2a The User Business or Background of the Project Effort 12](#_Toc461441757)

[2b Goals of the Project 12](#_Toc461441758)

[2c Measurement 13](#_Toc461441759)

[3 The Scope of the Work 13](#_Toc461441760)

[3a The Current Situation 13](#_Toc461441761)

[3b The Context of the Work 14](#_Toc461441762)

[3c Work Partitioning 16](#_Toc461441763)

[3d Competing Products 18](#_Toc461441764)

[4 The Scope of the Product 18](#_Toc461441765)

[4a Scenario Diagram(s) 18](#_Toc461441766)

[4b Product Scenario List 19](#_Toc461441767)

[4c Individual Product Scenarios 19](#_Toc461441768)

[5 Stakeholders 19](#_Toc461441769)

[5a The Client 19](#_Toc461441770)

[5b The Customer 20](#_Toc461441771)

[5c Hands-On Users of the Product 20](#_Toc461441772)

[5d Priorities Assigned to Users 21](#_Toc461441773)

[5e User Participation 22](#_Toc461441774)

[5f Maintenance Users and Service Technicians 22](#_Toc461441775)

[5g Other Stakeholders 22](#_Toc461441776)

[6 Mandated Constraints 23](#_Toc461441777)

[6a Solution Constraints 24](#_Toc461441778)

[6b Implementation Environment of the Current System 25](#_Toc461441779)

[6c Partner or Collaborative Applications 25](#_Toc461441780)

[6d Off-the-Shelf Software 26](#_Toc461441781)

[6e Anticipated Workplace Environment 27](#_Toc461441782)

[6f Schedule Constraints 28](#_Toc461441783)

[6g Budget Constraints 28](#_Toc461441784)

[7 Naming Conventions and Definitions 29](#_Toc461441785)

[7a Definitions of Key Terms 29](#_Toc461441786)

[7b UML and Other Notation Used in This Document 30](#_Toc461441787)

[7c Data Dictionary for Any Included Models 30](#_Toc461441788)

[8 Relevant Facts and Assumptions 31](#_Toc461441789)

[8a Facts 31](#_Toc461441790)

[8b Assumptions 32](#_Toc461441791)

[II Requirements 33](#_Toc461441792)

[9 Product Use Cases 33](#_Toc461441793)

[9a Use Case Diagrams 33](#_Toc461441794)

[9b Product Use Case List 34](#_Toc461441795)

[9c Individual Product Use Cases 35](#_Toc461441796)

[10 Functional Requirements 35](#_Toc461441797)

[11 Data Requirements 36](#_Toc461441798)

[12 Performance Requirements 38](#_Toc461441799)

[12a Speed and Latency Requirements 38](#_Toc461441800)

[12b Precision or Accuracy Requirements 39](#_Toc461441801)

[12c Capacity Requirements 39](#_Toc461441802)

[13 Dependability Requirements 40](#_Toc461441803)

[13a Reliability Requirements 40](#_Toc461441804)

[13b Availability Requirements 40](#_Toc461441805)

[13c Robustness or Fault-Tolerance Requirements 41](#_Toc461441806)

[13d Safety-Critical Requirements 42](#_Toc461441807)

[14 Maintainability and Supportability Requirements 43](#_Toc461441808)

[14a Maintenance Requirements 43](#_Toc461441809)

[14b Supportability Requirements 43](#_Toc461441810)

[14c Adaptability Requirements 44](#_Toc461441811)

[14d Scalability or Extensibility Requirements 44](#_Toc461441812)

[14e Longevity Requirements 45](#_Toc461441813)

[15 Security Requirements 45](#_Toc461441814)

[15a Access Requirements 45](#_Toc461441815)

[15b Integrity Requirements 46](#_Toc461441816)

[15c Privacy Requirements 47](#_Toc461441817)

[15d Audit Requirements 48](#_Toc461441818)

[15e Immunity Requirements 48](#_Toc461441819)

[16 Usability and Humanity Requirements 48](#_Toc461441820)

[16a Ease of Use Requirements 48](#_Toc461441821)

[16b Personalization and Internationalization Requirements 50](#_Toc461441822)

[16c Learning Requirements 51](#_Toc461441823)

[16d Understandability and Politeness Requirements 52](#_Toc461441824)

[16e Accessibility Requirements 52](#_Toc461441825)

[16f User Documentation Requirements 53](#_Toc461441826)

[16g Training Requirements 54](#_Toc461441827)

[17 Look and Feel Requirements 54](#_Toc461441828)

[17a Appearance Requirements 54](#_Toc461441829)

[17b Style Requirements 55](#_Toc461441830)

[18 Operational and Environmental Requirements 56](#_Toc461441831)

[18a Expected Physical Environment 56](#_Toc461441832)

[18b Requirements for Interfacing with Adjacent Systems 56](#_Toc461441833)

[18c Productization Requirements 57](#_Toc461441834)

[18d Release Requirements 58](#_Toc461441835)

[19 Cultural and Political Requirements 58](#_Toc461441836)

[19a Cultural Requirements 58](#_Toc461441837)

[19b Political Requirements 59](#_Toc461441838)

[20 Legal Requirements 60](#_Toc461441839)

[20a Compliance Requirements 60](#_Toc461441840)

[20b Standards Requirements 61](#_Toc461441841)

[III Design 61](#_Toc461441842)

[21 System Design 61](#_Toc461441843)

[21a Design goals 61](#_Toc461441844)

[22 Current Software Architecture 63](#_Toc461441845)

[23 Proposed Software Architecture 63](#_Toc461441846)

[23a Overview 63](#_Toc461441847)

[23b Class Diagrams 63](#_Toc461441848)

[23c Dynamic Model 63](#_Toc461441849)

[23d Subsystem Decomposition 64](#_Toc461441850)

[23e Hardware / software mapping 64](#_Toc461441851)

[23f Data Dictionary 64](#_Toc461441852)

[23g Persistent Data management 64](#_Toc461441853)

[23h Access control and security 64](#_Toc461441854)

[23i Global software control 65](#_Toc461441855)

[23j Boundary conditions 65](#_Toc461441856)

[24 Subsystem services 65](#_Toc461441857)

[25 User Interface 65](#_Toc461441858)

[26 Object Design 65](#_Toc461441859)

[26a Object Design trade-offs 65](#_Toc461441860)

[26b Interface Documentation guidelines 66](#_Toc461441861)

[26c Packages 66](#_Toc461441862)

[26d Class Interfaces 66](#_Toc461441863)

[IV Test Plans 66](#_Toc461441864)

[27 Features to be tested / not to be tested 66](#_Toc461441865)

[28 Pass/Fail Criteria 66](#_Toc461441866)

[29 Approach 67](#_Toc461441867)

[30 Suspension and resumption 67](#_Toc461441868)

[31 Testing materials ( hardware / software requirements ) 67](#_Toc461441869)

[32 Test cases 67](#_Toc461441870)

[33 Testing schedule 67](#_Toc461441871)

[V Project Issues 68](#_Toc461441872)

[34 Open Issues 68](#_Toc461441873)

[35 Off-the-Shelf Solutions 68](#_Toc461441874)

[35a Ready-Made Products 68](#_Toc461441875)

[35b Reusable Components 69](#_Toc461441876)

[35c Products That Can Be Copied 69](#_Toc461441877)

[36 New Problems 70](#_Toc461441878)

[36a Effects on the Current Environment 70](#_Toc461441879)

[36b Effects on the Installed Systems 70](#_Toc461441880)

[36c Potential User Problems 70](#_Toc461441881)

[36d Limitations in the Anticipated Implementation Environment That May Inhibit the New Product 71](#_Toc461441882)

[36e Follow-Up Problems 71](#_Toc461441883)

[37 Tasks 72](#_Toc461441884)

[37a Project Planning 72](#_Toc461441885)

[37b Planning of the Development Phases 72](#_Toc461441886)

[38 Migration to the New Product 73](#_Toc461441887)

[38a Requirements for Migration to the New Product 73](#_Toc461441888)

[38b Data That Has to Be Modified or Translated for the New System 73](#_Toc461441889)

[39 Risks 74](#_Toc461441890)

[40 Costs 75](#_Toc461441891)

[41 Waiting Room 76](#_Toc461441892)

[42 Ideas for Solutions 77](#_Toc461441893)

[43 Project Retrospective 77](#_Toc461441894)

[VI Glossary 78](#_Toc461441895)

[VII References / Bibliography 78](#_Toc461441896)

[VIII Index 78](#_Toc461441897)

### ****List of Figures****

[Figure 1 – Context Diagram That Represents the System 11](#_Toc461441898)

Figure 2 - Timeline of Possible Scenarios……………………………………………………….14

Figure 3 – Use-Case Diagram……………………………………………………………………25

Figure 4 – Data Requirements……………………………………………………………………42

Figure 5 – Class Diagram………………………………………………………………………...54

Figure 6 – Sequence Diagram…………………………………………………………………….55

Figure 7 – Puzzles and Tasks…………………………………………………………………….56

Figure 8 – Important Items and Objects………………………………………………………….57

Figure 9 – Controllers…………………………………………………………………………….57

Figure 10 – Hardware/Software……………………………………………………………….…59

Figure 11 – Work Breakdown Structure………………………………………………………….79

### ****List of Tables****

**The following sample table and figure are only here to initialize the list of figures and list of tables above. They should be removed when real ones are included in the document. ( I.e. delete this entire page when you can. )**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Participants** | **Activities** | **Notes** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table 1- Sample Table of Survey Dive Activity



Figure 1 - Sample Image of a Survey Dive Boat

( photo by Tony Kiefer )

# Project Description

## Project Overview

The Time Machine is a virtual reality program that will be able to transport you back in time to a three-dimensional landscape based upon an important historical event or culture. The program inserts the user into the experience that they select, where they are given a character who represents a citizen or important figure at the time. The user is assigned tasks related to the selected story line immersing them into the role. By letting the user explore their environment and solve puzzles along the way, they will learn valuable information about historical events while also being able to explore what life was like in the past. Using a controller in each hand, the user navigates through the environment allowing the user to interact with the world around them. The Time Machine will display heads up information about what the user is looking at letting them know who or what they are looking at. The Time Machine server on which the scenarios will be saved on, will be loaded with different scenarios to learn, and explore. The Time Machine servers will be loaded with new content regularly, opening the entire past to the users to explore.

## The Purpose of the Project

### The User Business or Background of the Project Effort

The Time Machine is being developed as a replacement for history textbooks. The material being taught through the software will provide an unforgettable learning experience by having the user interact with history. This interaction with the virtual landscape will make the user better retain the lesson by giving them an experience tied to the material that they learn. Puzzles that need to be solved within the game will make the user more invested in what is going on thereby forcing them to the learn the material. How exciting is it to sit down and read chapters from a history book that just seems to go on and on with details, facts, and dates? Not really, but that doesn’t mean that the material itself isn’t based upon exciting historical events that shaped the world. You could sit down and learn all about ancient Rome by reading a textbook, but wouldn’t it be more interesting to be able to walk around ancient Rome and see what it looked like? Wouldn’t you want to explore what the earth was like when dinosaurs ruled the world? This product will help to make history fun again. One thing to consider is trying to make the content interesting and not boring. This is serious because the whole point of the project is to give the user an entertaining experience and not to bore the user. Another thing to consider being able to make the user can feel like they are in another world, which requires a lot of research to give the environment the correct look and feel the developers were going for. This is important because if the environment doesn’t give off the right atmosphere, it will not give the user the full effect of the project.

### Goals of the Project

We want to give the user accurate information about the time and location they are in. The whole point of The Time Machine is to be a replacement for a textbook, so the project needs to stay on track with providing the same level of quality information. We also want to give the user an interactive experience in which they can interact with the world. The Time Machine shouldn’t just feel like a tour of an environment. Next, we want the user to be engaged with their surroundings and feel immersed in the world. We want to make sure that the user can retain information that they learn inside of the simulation. Lastly, we want to make sure the system runs smoothly so that the user doesn’t experience lag or faulty visuals.

### Measurement

To ensure that all the information that we use is correct it may be smart to test the product on experts in the field who will give feedback on whether the information is correct and not construed. Different sources for information may have contradicting facts and it should be made sure that the information our product gives is relevant to the historical period the user is in.

To make sure that the users feels like they are being an integral part of the learning experience, we must make sure that there is a good balance of information being given to the user versus letting the user interact with their environment. We don’t want there to be too much lecturing and listening but we also don’t want the user to feel like they must constantly be interacting with the environment.

Since the user is supposed to be using our product for educational purposes, they should be able to remember the information that they are given. To test the user’s retention rate, we can test them right after they use the product, a day after, a week and so on. By taking data at these points we can see how well the users retained what they learned.

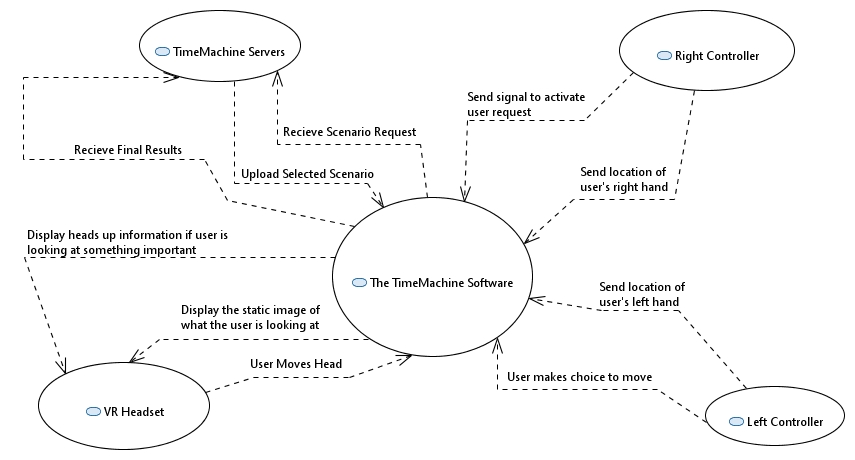
It’s very important that we make the system on which The Time Machine runs on is extremely smooth. Since the user will have their vision completely covered by the headset, we don’t want poor graphics or lagging to give them the full experience without being distracted. We can measure this by testing the system extensively to see if there are any bugs that need to be worked out or performance issues.

## The Scope of the Work.

### The Current Situation

The Time Machine will combine the use of a Virtual Reality headset and two controllers that the user holds in each hand. The left controller will contain a joy stick for movement while the right joystick will have a button on the bottom of it for making selections within The Time Machine. Each controller and the headset will contain accelerometers to track the users head movement and where the user’s hands are located, making the user feel like they are inside the three-dimensional world. The system will track these movements and react to what the user is doing based on user selections and user location. For example, if the user selects to option to talk to a non-playable character then the system should react by the system initiating correct response. The system also builds boundaries so that players cannot walk out of the environment or stop users from walking through people or walls. The system also must display a heads-up information box that is dependent on where the user is looking. The system will connect to The Time Machine servers which contain downloads for each scenario. The server will then upload the experience for the user and when the user finishes their experience the system must send the results of their experience back to the servers to be stored.

### The Context of the Work

**Figure 1- Context Diagram that Represents the System**

### Work Partitioning

Business Event List

Event Name Input and Output Summary

|  |  |  |
| --- | --- | --- |
| 1. System sends image to headset | Two-Dimensional image (output) | System combines head location, hand location, movement, and interaction. then displays images of what the user looking at on the screen. |
| 2. Display heads up information | Text box with fact (output) | If what the user is looking at is flagged as important, an text box displaying information will appear in their vision. |
| 3. User selects direction to move | Intended direction signal via joystick (input) | System receives if the user wants to move forward, backward, left, right not move. |
| 4. User moves left hand | Change in Left Location (input) | System records where the user moved their left hand in 3-D space. |
| 5. User moves right hand | Changed in Right Location (input) | System records where the user moved their right hand in 3-D space. |
| 6. User moves head | User head position (input) | The accelerometer in the headset tracks where the user is looking in 3-D space. |
| 7. User selected an intractable object | Send signal to system that event has been triggered (input) | Record the event that the users selects then send back the correct response or animation. |
| 8. User selects a scenario from the menu | User sends scenario ID to server (input) | After the user selects a level, the program must take that selection and send the request to the server. |
|  |  |  |
| 9.Server uploads level | Environment uploaded to the system (out) | After the server finds the selected scenario, it is then uploaded to the system so the user can start the experience. |
| 10. User completes the scenario | Final report sent to server (out) | Once the user finishes, all information collected about the users experience is then saved on the server |
|  |  |  |

### Competing Products

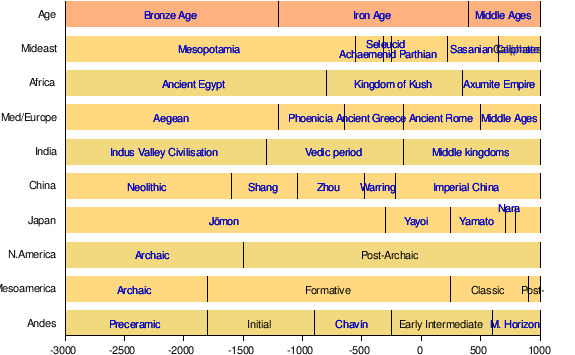
There are some similar products that do exist already. Lecture VR is an app from Immersive VR Education, in which the user sits in a virtual lecture hall and is taught by an instructor with visual aids. While Lecture VR places the person a virtual classroom, the product fails to engage the user with activities and the classroom isn’t very visually appealing. Alchemy VR is another product that lets users go on educational adventure with a narrative like a virtual reality documentary. The Alchemy VR immerses the user into a story and three-dimensional world but still fails to engage the user as they are just watching and not really interacting with what is going on. There are also various travel virtual reality programs that can show people many locations around the world but provide little to no educational material.

Our product is different from these existing products because in The Time Machine, the user is the one in control with the freedom to move throughout the world. Most of all these existing products are more like a guided tour or an instructional video that lets the user just look around and not be able to interact with the environment that they are in. We believe that by taking this next step in making a more hands on experience that it will further the learning of the users of the product.

## The Scope of the Product

This product will be created for use with a virtual reality headset that will be connected to two controllers. Vision will be dictated on an axis based on the orientation of the headset, and the controllers will dictate movement as well as interaction with the virtual world. The educational system will transport the user to whichever historical period they select, followed by the completion of various educational quests and puzzles that will seek to educate the user as necessary. Once the scenario is over, the program returns the user to the main menu and prompts them to select another scenario.

### Scenario Diagram(s)

There are various periods within human history that are interest to the educational field. For example, the following diagram illustrates human history in a broad manner

**Figure 2- Timeline of Possible Scenarios**

This illustrates the many different periods that a user could potentially choose when using our virtual reality simulation. For example, a user could choose “Iron Age”, followed by either “Ancient Greece” or “Ancient Rome”. The user would then enter the simulation, which would allow him / her to effectively travel back in time. The simulation would be an immersive experience that simulates time travel for an incredibly valuable, unique, once in a lifetime educational experience.

### Product Scenario List

This product will initially cover major time-periods within ancient history. Once these eight scenarios are released, developers can start to work on new scenarios to be released on a regular basis. The major time-periods within ancient history that have been chosen for the initial scope of this product are as follows.

Ancient Greece (0 AD)

Ancient Rome (300 AD)

Middle Ages (1100 AD)

The Renaissance (1600 AD)

The American Revolution (1765 AD)

The Industrial Revolution (1820 AD)

WWII (1940 AD)

Vietnam War (1965)

### Individual Product Scenarios

Each historical moment will have a specific story line that will play out for the user within the virtual reality environment. These story lines will be tailored to a specific time and place, and will highlight notable aspects of the era’s nature. The initial story lines to be released are as follows:

**Ancient Greece:** Spend a day as a merchant citizen within one of Ancient Greece’s many city-states.

**Ancient Rome:** Observe a day of games at the famous Coliseum within the confines of the Ancient Roman empire.

**Middle Ages:** Live out a day as a medieval knight within the middle ages, and participate in an ancient festival.

**The Renaissance:** Experience firsthand what it was like as a spice trader that lived within Italy in the 1600’s, and try to trade your way to as much wealth as possible.

**The American Revolution:** Witness the Boston Tea Party first hand, as well as a day on the battlefield.

**The Industrial Revolution:** Work as a miner in a rural American city, and participate in a strike as your workforce attempts to unionize.

**WWII:** Spend a day as a German citizen and witness firsthand the realities of living within a war-torn country.

**Vietnam War:** Spend a day as a politician within U.S. congress, and see how the decision to go to war with Vietnam was decided behind closed doors.

## Stakeholders

### The Client

The client who is funding The Time Machine project is Cengage Learning. Cengage is a long time textbook publisher who wants to tap into the educational virtual reality market by making an exciting product to bring to schools across the world. We will be working closely with Jim Donohue who is Cengage’s Chief Product Officer. Jim’s role at Cengage is to make sure that all of their products meet their high expectations.

### The Customer

The main customer for this product will be elementary and middle school districts that want to bring this learning experience to their students and teachers. Schools budget each year for technology needs, some with a larger budget than others, but at some point, the schools will have to upgrade their current systems so The Time Machine is a viable option to schools interested in what VR can bring to the table from an educational perspective.

Another set of customers for The Time Machine would be libraries and museum’s. Libraries could invest in this product to get more people in the door and excited about learning. Similarly, museums could invest in a Time Machine system to show patrons a historical VR simulation that correlates to an exhibit to further enhance the users experience and knowledge. By giving the general population access to VR devices, even if they don’t own the device, it may help increase sales of devices and Time Machine systems by giving users a preview to VR experiences.

### Hands-On Users of the Product

The Time Machine’s first focus groups are children and young adults ages 10 to 15. Users on the younger side of this group typically are in their last couple of years of elementary school where teachers can start to get into deeper detail when talking to students than say a first grader. The users at the top of this age range are usually finishing their freshman year of high school and may be less interested in the product due to their age.

The next group of users that The Time Machine will attract are older scholars and college students. These users are most likely already educated about world history by this point in their lives but in doing research, The Time Machine gives them a first person learning experience that may help them to better understand what they have learned from a textbook already.

Another group of users that will find The Time Machine appealing are people who already own and use VR equipment regularly. These people may try the product to explore the worlds and do the activities but not necessarily focus on the educational part. While they may not be focused on it though, The Time Machine will still provide these users with all kinds of information.

### Priorities Assigned to Users

The key users for The Time Machine are the group of students ages 10-15. These users will be learning about these concepts for the first time so it is showing them something that they have never heard of before from an extremely hands on way. Users within this age range also have a higher capacity for learning than older users.

The secondary users would be the older students. The Time Machine is a tool for learning and giving this learning experience to older students will change the way they look at research. These students already have a solid base knowledge on the historical events thereby making it a secondary learning tool for them and not really a first option.

The unimportant users are the users who are only interested in The Time Machine for its virtual reality capabilities. While we are trying to provide a visually stunning experience for users, our main priority is education and not tourism.

### User Participation

After each run of the simulation, a report will be sent to the server which tracks what the user did throughout their experience. This report includes information such as time taken to complete, attempts made versus correct answers given, and the path which the user took within the simulation. After collecting this data from the users, the data can be used to tell if the program is giving the user a smooth run. The data could also tell the developers that the users are regularly getting lost within the environment, or the material is too hard for the user. If the game is confusing for the user, the project will be dull and seem like a waste of time for the user. It’s important that the user feedback is taken seriously so that the developers know how to correct issues before the release of the program.

### Maintenance Users and Service Technicians

The Time Machine developers will be responsible for maintaining the system once the product is released. Developers will test and check the system regularly to ensure that there are no bugs in the system. The Developers will make sure that the system is running cohesively and make sure that all devices can connect to the servers properly. Hardware problems must be dealt with by the manufacturer of whichever VR device the user is using.

### Other Stakeholders

To make sure that all the information used in the project is factually correct, we will need to consult with expert historians from each field to provide the users with the most accurate information. These experts will be crucial to the project since the developers will not have the same level of knowledge on the topic as the historian. If there was some case in which two experts don’t agree on a topic and our team can’t pick a side, a third expert may be needed to give a third professional opinion on the topic.

Cengage Learning’s marketing experts will help to get the product out there and make sure that there are enough people willing to buy The Time Machine by the release date and then continue to advertise it using videos and images from upcoming releases. The marketing division can also help us to come up with a logo to brand ourselves. Without their help, we wouldn’t be able to be recognized against competing products.

We will also need a large quantity of testers to test the project before it is released. These testers will run through the different scenarios and give us feedback on any bugs or performance issues. Testers will also be given quizzes after testing to see if the software helps them learn important information about their tested scenario. These testers are important because they will give us a good idea of how the general population will react to the product and provide constructive criticism.

## Mandated Constraints

### Solution Constraints

Description: The Time Machine software must run on all available VR headsets on the market. This means that the following headsets should be fully compatible: Oculus Rift, Samsung Gear, HTC Vive, and Google Cardboard. This will enable the simulation to satisfy the client as much as it possibly can. If the product does not run on all available VR headsets on the market, then this may cause the client to be dissatisfied with the simulation as a product, and attempt to renegotiate the contract.

Rationale: Most students or schools will not be able to afford a high-end VR headsets so making it available on cheaper headsets (such as the Google Cardboard) will make the product more accessible. The product will not specialize in high performance VR headsets, instead opting to specialize in low to medium performance VR headsets to ensure compatibility.

Fit criterion: All videos and the interactive features of the product shall be available to the users via their headsets. This will ensure that there is not an excessive need to purchase further hardware to access the simulation, which would actively prevent the simulation from reaching as many people as it possibly could. It would also make the simulator bulkier and less attractive.

Description: The product shall operate on all VR operating systems. This will enable to product to work for the client no matter what, and will also ensure that the product has cross functionality if it is used in unforeseen circumstances. If the product were incompatible with several VR operating systems, this would result in an incredible risk that could result in the product being received negatively.

Rationale: The client does not wish to change the operating system of the headset. Therefore, we will ensure that the client will be able to use the simulation without needing to purchase a VR headset with a different operating system.

Fit criterion: The product shall be approved on all OS by the testing group. This will be accomplished by exhaustively ensuring that each time-period can load and run smoothly on every VR headset / operating system combination on the market.

### Implementation Environment of the Current System

The “Time Machine” will be installed onto a VR headset operating system for the client, with a connected controller that will allow the user to interact with the simulated environment. This interaction will allow the user to navigate and interact with the simulated environment. Every installation will include every educational period created for the system.

The product must operate on all existing VR operating systems. Therefore, the designers must ensure that the created system is cross compatible across every different system on the marketplace. This is made simple by using development software such as Unity Pro, which supports cross compatibility natively. Also, it is imperative that the included controllers work with the same functionality regardless of which operating system is installed on the VR headset being used.

The entire “Time Machine” software suite will include the following: One “Time Machine” software installation, which will be compatible with any of the following VR headsets: Oculus Rift, Samsung Gear, HTC Vive, and Google Cardboard. The suite itself will include several specific time periods, which are listed as follows: Ancient Greece (0 AD), Ancient Rome (300 AD), Middle Ages (1100 AD), the Renaissance (1600 AD), the American Revolution (1765 AD), the Industrial Revolution (1820 AD), WWII (1940 AD), and the Vietnam War (1965). The product also must include a controller that allows the user to interact with the environment as necessary, i.e. solving puzzles.

### Partner or Collaborative Applications

To maximize the number of customers that our product reaches, we should make the product compatible with all the VR headset platforms available on the market that can meet the system requirements of The Time Machine software. Some of these include the Oculus Rift, Samsung Gear, HTC Vive, and Google Cardboard.

### Off-the-Shelf Software

The first piece of OTS software that will be needed to create The Time Machine is Unity 3D Pro edition. This software lets you develop your entire project in one location and then deploys to all the different VR operating systems. The Unity software also sends developers analytics about how the program is being used. Unity also contains performance reporting that collects errors and works to debug it. The software is 125$ a month per seat to develop on but there is no revenue cap for projects developed on their pro level license.

### Anticipated Workplace Environment

The Time Machine is mainly developed to be used in a classroom or museum so the amount of movement by the user should be limited. This is so that people are not bumping into each other or walking into walls since their vision and hearing will be impaired. Another environmental constraint is that The Time Machine should not rely on voice input. The product may be being used by multiple people in the same room such as a classroom so it would be very difficult to register voices when everyone is talking. Another reason The Time Machine should not require voice input is because it maybe being used in an area where a user should be quiet such as a library or museum.

### Schedule Constraints

The Time Machine must be completed in a 12-week period. This is due to two main constricting factors. The first constraint is the budget for the project. The budget only can afford to keep the project alive for 12 weeks. The second constraint is that we want to deploy the product before the beginning of the new school year so more schools can start using it in the classroom. Schools will be more inclined to use our system if they don’t have to transfer systems mid-year3.

### Budget Constraints

The budget to create The Time Machine system is $150,000 US dollars. The first $120,000 will be used to create the system, contract experts, and to be used on any other needed resources before the product is launched. The remaining $30,000 is to maintain and update the system for Cengage. Cengage has made it clear that we cannot go over budget on this project.

## Naming Conventions and Definitions

### Definitions of Key Terms

Virtual Reality (VR) – A program in which a user’s entire vision is covered by a screen giving the effect that they are in another location or world. Virtual Reality takes place in a three-dimensional environment

Three-Dimensional (3D) – A space that is defined by having a height, width and depth.

Two-Dimensional(2D) – A space that is defined by having a height and width but no depth or flat.

Off-The-Shelf Software (OTS) – Software that you can purchase from a store or online.

Headset – The piece of hardware that the user puts on their head that contains a screen and/or speakers to view the virtual reality program

### UML and Other Notation Used in This Document

To build UML diagrams for this document, we will be using Eclipse Papyrus. Papyrus uses UML 2.5.0 which is the most up to date version of UML. The arrows in a UML diagram tell how things relate to each other. A dotted arrow with a closed head is a dependency which means that the arrow shows that the first element is dependent on the second element. An open arrow is an inheritance so the first element is a child of the second element. A closed arrow represents an association between two things but they are each independent of each other. Things placed in circles are different classes or states.

### Data Dictionary for Any Included Models

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Description** | **Type** | **Date** |
| 1. System sends image to headset | System combines head location, hand location, movement, and interaction to make a 2D image that is sent to the headset | Relation | 2/17/2017 |
| 2. Display heads up information | Information is shown in the text box | Relation | 2/17/2017 |
| 3. User selects direction to move | Joystick of the left controller sends the intended direction of motion. | Relation | 2/17/2017 |
| 4. User moves left hand | Location of left hand in 3D space is recorded. | Relation | 2/17/2017 |
| 5. User moves right hand | Location of right hand in 3D space is recorded. | Relation | 2/17/2017 |
| 6. User moves head | Location and direction of head is recorded to track its motions in a 3D space. | Relation | 2/17/2017 |
| 7. User selected an intractable object | User tries to access item that is out of the boundary of the environment. The appropriate response is sent by the system. | Relation | 2/17/2017 |
| 8. User selects a scenario from the menu | Scenario ID of the selected scenario is sent to the server | Relation | 2/17/2017 |
|  |  |  |  |
| 9.Server uploads level | Server uploads the environment for the user to experience. | Relation | 2/17/2017 |
| 10. User completes the scenario | User usage report and statistics are calculated and stored on the server. | Relation | 2/17/2017 |
| 11. The Time Machine Software | The product which is developed in collaboration with Cengage Learning | Entity | 2/17/2017 |
| 12. Time-Machine Servers | Servers which will be used to maintain the various historical scenarios of the product. They will also store user data and usage statistics. | Entity | 2/17/2017 |
| 13. VR Headset | Hardware that displays visual and audio output to the user. It also provides head location and direction information. | Entity | 2/17/2017 |
| 14. Right Controller | Input device for movement of right hand and recording user selections. | Entity | 2/17/2017 |
| 15. Left Controller | Input device for movement of left hand and selecting direction. | Entity | 2/17/2017 |
|  |  |  |  |

## Relevant Facts and Assumptions

### Facts

Location and Puzzle relation are displayed in the game.

Puzzle difficulty increases as the game progresses.

Puzzle will be fetched from a dedicated database.

Includes facts about important locations.

Includes facts about important people.

The game is played from a first-person perspective.

### Assumptions

It is assumed that the younger group of users will be more interested in the product than the older users. This may be wrong and we find out that it is better suited for a higher level of education.

It is also assumed that the users will retain the information for an extended period. By reexamining the testers after one month of the experience and again after 2 months, we can hopefully see that the users will retain the information.

We are also assuming that all of the platforms will be able to handle the program requirements.

The game is cross-platform supported. So, it can be played in android, iOS, windows, Mac OS X, and console systems.

Database gets updated frequently, usually on a 24-hour basis.

Game data is transferable from device to device since user data is stored on the servers.

# Requirements

## Product Use Cases

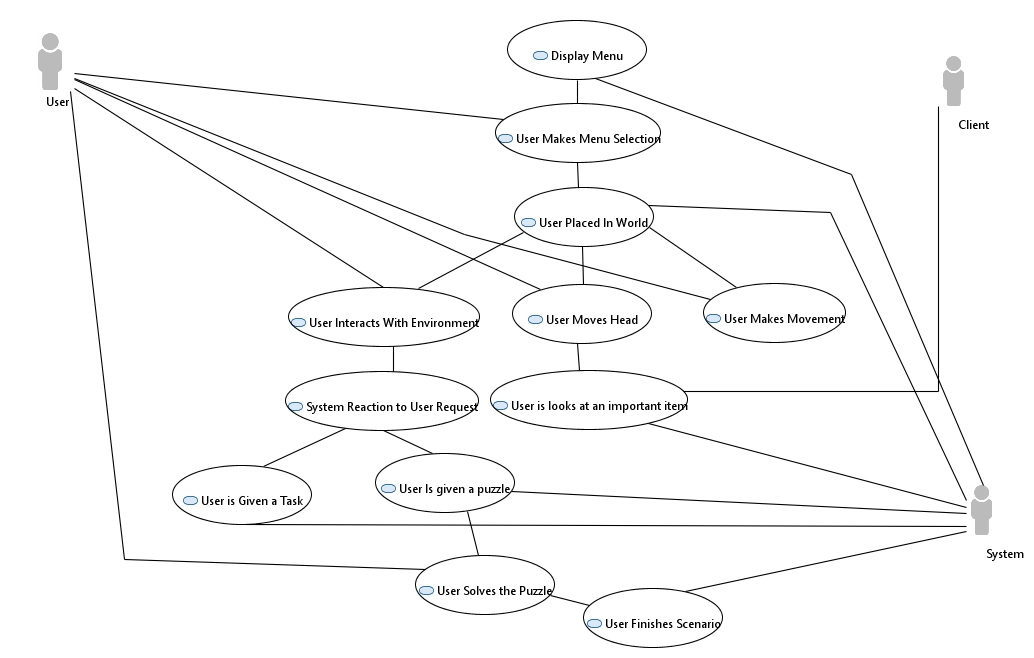
### Use Case Diagrams

Use Case diagrams serve two purposes: As a form of graphical table of contents listing the individual use-cases, and also to define the boundary of what is included as part of the proposed system and what is not included.

A use case diagram identifies the boundaries between the users (actors) and the product. You arrive at the product boundary by inspecting each business use case and determining, in conjunction with the appropriate stakeholders, which part of the business use case should be automated (or satisfied by some sort of product) and what part should be done by the user. This task must take into account the abilities of the actors (section 3), the constraints (section 4), the goals of the project (section 1), and your knowledge of both the work and the technology that can make the best contribution to the work.

The use case diagram shows the actors outside the product boundary (the rectangle). The product use cases are the ellipses inside the boundary. The lines denote usage. Note that actors can be either automated or human.

Depending on the complexity of the product it may be necessary to use more than one diagram to list all of the use cases. When more than one diagram is required the use-cases can be divided up several ways: Normal operations versus exceptional cases, or daily tasks versus monthly tasks, or user tasks versus administration tasks, etc.



### Individual Product Use Cases

|  |
| --- |
| Use case ID: 1 Name: Main Menu is Displayed  pre-conditions: User is logged into their account  post-conditions: Menu Displayed on headset  Initiated by: System  Triggering Event: Initial Startup or Scenario finished  Additional Actors: User |
| Sequence of Events:  Machine Is Started or Scenario is finished   1. If scenario is finished, upload user data   System displays the main menu to the user’s headset |
| Alternatives: User ends the program before menu is displayed  Exceptions: |

|  |
| --- |
| Use case ID: 2 Name: Menu Item Selections  pre-conditions: Main Menu is being displayed  post-conditions:  Initiated by: User  Triggering Event: User  Additional Actors: |
| Sequence of Events:   1. System displays main menu to user 2. User uses joystick on controller to navigate the menu items   User uses button on the controller to select menu item   1. The selection is sent to the system   System sends level data to the user |
| Alternatives: User selects settings  Exceptions: |

|  |
| --- |
| Use case ID: 3 Name: User is placed in Environment  pre-conditions: User selects a level from main menu  post-conditions: Selected scenario is Initiated  Initiated by: System  Triggering Event: User makes selection  Additional Actors: |
| Sequence of Events:   1. Selected scenario is uploaded 2. System finds user starting location   System places user in starting location   1. Scenario is initiated |
| Alternatives: User is displayed settings instead of scenario  Exceptions: User tries to access scenario not yet available |

|  |
| --- |
| Use case ID: 4 Name: User Moves Head  pre-conditions: User is inside scenario  post-conditions: Headset display is updated  Initiated by: User  Triggering Event: User looks up/down or left/right  Additional Actors: System |
| Sequence of Events:   1. User looks up, down, left or right 2. Signal is sent to system   System sends back new image of what is displayed   1. Image is then displayed for user |
| Alternatives: position locked when inside of puzzles so head cannot move laterally  Exceptions: |

|  |
| --- |
| Use case ID: 5 Name: User makes movement  pre-conditions: User is in game, User is not inside puzzle  post-conditions: User is in new location  Initiated by: User  Triggering Event: Joystick is moved  Additional Actors: System |
| Sequence of Events:   1. User moves joystick up, down, left, or right 2. System receives movement request   System decides if user can make that move   1. System sends new location of user   User moves to the new location |
| Alternatives: If joystick is used in a puzzle it will not change location  Exceptions: User tries to walk into a boundary |

|  |
| --- |
| Use case ID: 6 Name: User looks at important item  pre-conditions: users display is up to date  post-conditions: heads up fact box displayed  Initiated by: system  Triggering Event: User moves head  Additional Actors: user, client |
| Sequence of Events:   1. User looks at an important item 2. System receives signal that user is looking a specific item   System retrieves information about said item   1. System displays fact on user display via a pop-up window   User manually closes window or disappears 30 seconds after user looks away from the item |
| Alternatives:  Exceptions: |

|  |
| --- |
| Use case ID: 7 Name: User makes interaction with environment  pre-conditions: User is in a scenario  post-conditions: Request is sent to system  Initiated by: User  Triggering Event: User presses button  Additional Actors: System |
| Sequence of Events:   1. User views interactable object on display 2. User aims controller at said object   User presses the button to select the object   1. Signal is sent to system to request reaction |
| Alternatives: No request sent if object is not interactible  Exceptions: |

|  |
| --- |
| Use case ID: 8 Name: System reaction to request  pre-conditions: user has pressed a button  post-conditions: correct response is given  Initiated by: User  Triggering Event: User selects intractable object  Additional Actors: |
| Sequence of Events:   1. Interaction request is received from the user 2. System decides if user is initiating a Task or Puzzle   User is given a text box and a voice over.   1. Task is given to user or Puzzle is given to user |
| Alternatives: If already in puzzle, puzzle is updated  Exceptions: User interacts with something that is not a Task or Puzzle such as a random civilian. |

|  |
| --- |
| Use case ID: 9 Name: User is given a task  pre-conditions: User has not finished level  post-conditions: Task list is updated  Initiated by: System  Triggering Event: System decides that user requested a task  Additional Actors: |
| Sequence of Events:   1. System decides that the user requested item is a task 2. System checks to see if task has already been completed   If not, the system updates the users task list giving them a new item |
| Alternatives: User is prompted with just a response and no task  Exceptions: |

|  |
| --- |
| Use case ID: 10 Name: User is given a puzzle  pre-conditions: User is not inside of a puzzle  post-conditions: User is displayed puzzle  Initiated by: System  Triggering Event: System decides user request is a puzzle  Additional Actors: |
| Sequence of Events:   1. System decides user request is for a puzzle 2. Information on how to interact with puzzle is displayed   Users display is updated with puzzle |
| Alternatives: If user is already inside of puzzle, update existing puzzle  Exceptions: |

|  |
| --- |
| Use case ID: 11 Name: User Solves puzzle  pre-conditions: User is inside of puzzle  post-conditions: User exits the puzzle  Initiated by: User  Triggering Event: Puzzle Move  Additional Actors: System |
| Sequence of Events:   1. User moves puzzle pieces 2. This then sends a signal to the system with the updated puzzle   The system sees that the puzzle matches the solution   1. The user is congratulated via a text window and voice over   Puzzle is exited |
| Alternatives:  Exceptions: Glitch such that there is no solution for puzzle |

|  |
| --- |
| Use case ID: 12 Name: Scenario Finished  pre-conditions: User has completed all tasks  post-conditions: Main menu is displayed  Initiated by: System  Triggering Event: User completes final task  Additional Actors: |
| Sequence of Events:   1. User completes final task 2. Signal is sent to system notifying that the user completed all tasks   System gives user final cut scene   1. After cut scene user is returned to main menu |
| Alternatives: User exits manually  Exceptions: Unexpected System shutdown |

## Functional Requirements

Requirement #: 1 Requirement Type: 1 Use Case #: 4

Description: The product shall always record users head location

Rationale: To be able to display the user with the correct image

Originator: Zachary Jones

Fit Criterion: The System should update users head location every 10milliseconds to ensure fluid display to user

Customer Satisfaction: 5 Customer Dissatisfaction: 4

Dependencies: Location of headset based on user Conflicts:

Supporting Materials:

History: 3/14/17

Requirement #: 2 Requirement Type: 1 Use Case #: 7

Description: The product should always record user hand locations

Rationale: To be able to tell where the user is pointing in the environment

Originator: Zachary Jones

Fit Criterion: The System should update users hand location every 10 milliseconds to ensure fluid movement

Customer Satisfaction:4 Customer Dissatisfaction:3

Dependencies: Location of controller Conflicts:

Supporting Materials:

History: 3/14/17

Requirement #: 3 Requirement Type: 1 Use Case #: 5

Description: The product will consistently record joystick position

Rationale: To be able to tell if user is making movement request

Originator: Zachary Jones

Fit Criterion: The joystick location will be recorded every 10 milliseconds to tell the system if the user is stationary or trying to move in any direction

Customer Satisfaction:5 Customer Dissatisfaction:5

Dependencies: Joystick position on controller Conflicts:

Supporting Materials:

History: 3/14/2017

Requirement #: 4 Requirement Type: 1 Use Case #:7

Description: The product should record button position on controller

Rationale: To determine if user is making an interaction request

Originator: Zachary Jones

Fit Criterion: To record if the button is pressed or not signaling that the user is making a request. Button position will be updated every 10 milliseconds.

Customer Satisfaction: Customer Dissatisfaction:

Dependencies: Button position on controller Conflicts:

Supporting Materials:

History: 3/14/17

Requirement #: 5 Requirement Type: 2 Use Case #: 2

Description: The product shall keep record of how much time has passed

Rationale: To be able to correctly update user requests, timeout periods, and time taken in scenario

Originator: Zachary Jones

Fit Criterion: The product will start a timer once the system is booted up then can update user requests every 10 milliseconds and be able to tell how long user is in simulation by recording start and finish times

Customer Satisfaction:3 Customer Dissatisfaction: 2

Dependencies: how long the system has been initiated Conflicts:

Supporting Materials:

History: 3/14/17

Requirement #: 6 Requirement Type: 2 Use Case #: 5

Description: The product will record user position inside scenario

Rationale: To determine what the user will be able to look and interact with

Originator: Zachary Jones

Fit Criterion: By knowing the location of the user, the system can tell the user what they can look at and interact with

Customer Satisfaction: 5 Customer Dissatisfaction: 5

Dependencies: Users movement inside simulation Conflicts:

Supporting Materials:

History: 3/14/17

Requirement #: 7 Requirement Type: 2 Use Case #: 5

Description: The system shall not let users move through designated boundries in scenario

Rationale: To make sure user is not walking through walls or objects

Originator: Zachary Jones

Fit Criterion: The system should not allow users to walk through walls, objects, or other people in scenario to ensure realistic gameplay

Customer Satisfaction: 4 Customer Dissatisfaction: 4

Dependencies: Users position in scenario Conflicts:

Supporting Materials:

History: 3/14/17

Requirement #: 9 Requirement Type: 2 Use Case #:7

Description: system shall allow users to interact with certain items in the scenario

Rationale: To make sure user feels like an integral part of the simulation

Originator: Zachary Jones

Fit Criterion: Intractable items will be labeled or highlighted in order to allow users to know what they can and cannot interact with

Customer Satisfaction: 5 Customer Dissatisfaction:3

Dependencies: user location in scenario, controller button position Conflicts:

Supporting Materials:

History: 3/14/17

Requirement #: 9 Requirement Type: 2 Use Case #: 6

Description: Important items inside simulation will give heads up window

Rationale: To provide user with specific facts about an object

Originator: Zachary Jones

Fit Criterion: If the user is looking at an important item, then facts about the item should be displayed to user until the user exits the window or it times out

Customer Satisfaction: 2 Customer Dissatisfaction:2

Dependencies: User head position, time passed Conflicts:

Supporting Materials:

History:3/14/17

Requirement #: 10 Requirement Type: 3 Use Case #:3

Description: The system shall be able to upload correct scenario

Rationale: So the user is not given the wrong scenario or no scenario at all

Originator: Zachary Jones

Fit Criterion: The user sends a request to the system via the main menu so each menu item must produce the correct scenario for the user.

Customer Satisfaction:5 Customer Dissatisfaction:5

Dependencies: User controller position, controller button position Conflicts:

Supporting Materials:

History:3/14/17

Requirement #: 11 Requirement Type: 3 Use Case #:12

Description: The system shall return the user to main menu when scenario is finished

Rationale: The user has completed the scenario

Originator: Zachary Jones

Fit Criterion: If all tasks and puzzles inside of scenario are finished, the user will be returned to the main menu

Customer Satisfaction: Customer Dissatisfaction:

Dependencies: User completes all tasks Conflicts:

Supporting Materials:

History: 3/14/17

Requirement #: 12 Requirement Type: 3 Use Case #:11

Description: The system will exit puzzle upon completion

Rationale: The puzzle is solved so there is nothing else to complete

Originator: Zachary Jones

Fit Criterion: The system will check the position of the puzzle to see if it is in its final position after each user move

Customer Satisfaction: Customer Dissatisfaction:

Dependencies: Puzzle piece positions Conflicts:

Supporting Materials:

History: 3/14/17

Requirement #: 13 Requirement Type: 3 Use Case #: 8

Description: The System must save user data from scenario

Rationale: In order to keep record of users progress

Originator: Zachary Jones

Fit Criterion: after the scenario is finished a detailed report shall be sent to the servers so that it can be analyzed

Customer Satisfaction: 5 Customer Dissatisfaction: 5

Dependencies: User finishes scenario Conflicts:

Supporting Materials:

History: 3/14/17

Requirement #: 14 Requirement Type: 3 Use Case #:9

Description: When assigned a task the task list is updated

Rationale: To keep an updated log of what the user should be doing

Originator: Zachary Jones

Fit Criterion: After task is assigned, the system should create a new task that will appear in the users task list

Customer Satisfaction: Customer Dissatisfaction:

Dependencies: User interaction Conflicts:

Supporting Materials:

History: 3/14/17

Requirement #: 15 Requirement Type: 3 Use Case #: 9

Description: System shall remove task once completed

Rationale: The user has finished so needs to be removed

Originator: Zachary Jones

Fit Criterion: After the user finishes the task, the task is removed from the task list

Customer Satisfaction: 5 Customer Dissatisfaction: 3

Dependencies: User interaction Conflicts:

Supporting Materials:

History: 3/15/17

Requirement #: 16 Requirement Type: 3 Use Case #:1

Description: Upon startup, the main menu is displayed

Rationale: the user has not selected a scenario yet

Originator: Zachary Jones

Fit Criterion: When the user starts the software, the system will display the main menu with different scenario options

Customer Satisfaction: 4 Customer Dissatisfaction: 5

Dependencies: User has just started software Conflicts:

Supporting Materials:

History: 3/15/17

Requirement #: 17 Requirement Type:3 Use Case #: 12

Description: User is returned to main menu after scenario finishes

Rationale: The user has completed the scenario

Originator: Zachary Jones

Fit Criterion: After all tasks have been completed the user has no more tasks to do so they must be exited from the scenario. They then are displayed the main menu to choose a new scenario from

Customer Satisfaction: Customer Dissatisfaction:

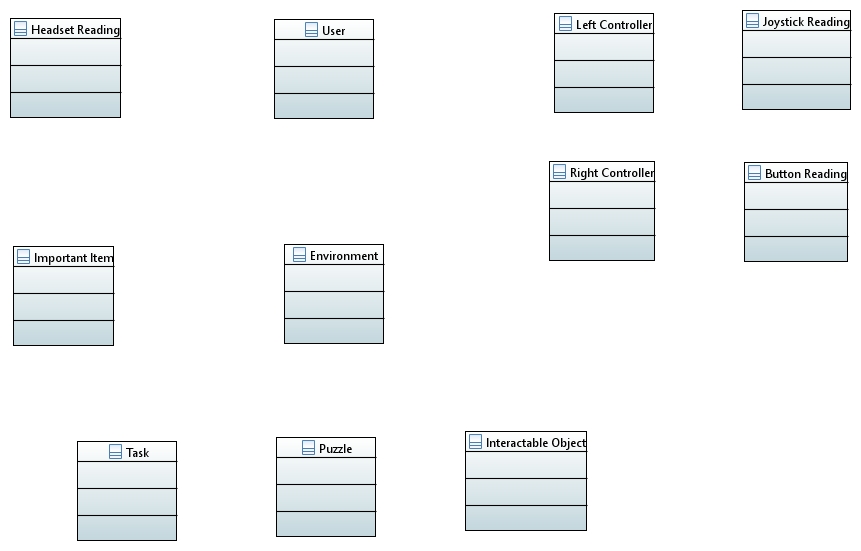
Dependencies: User has completed all tasks Conflicts:

Supporting Materials:

History: 3/15/17

.

## Data Requirements

 1 1 1 1

3 1 1

1 1 1 1

1

0…\* 1

1 1 1

1…\* 1…\* 1…\*

**Figure 4 – Data Requirements**

In the figure above, we can see that each user class has three different subclasses. These classes show that for each user there will be a headset reading and readings from both controllers. Each environment can hold one user at a time. Within the environment there are objects that are labeled as important items, tasks, puzzles and intractable objects. Each important item holds information about item coordinates and the stored fact. Tasks are uncompleted and each have requirements that need to be met before they can be set to completed. Puzzles have pieces and will not be set to correct until final position is equal to current position. Finally, intractable objects will store a cause and an effect so that when a user clicks one the system knows the correct response

## Performance Requirements

### Speed and Latency Requirements

Updating the locations of the external sensors should occur at least 10 times per second to make gameplay fluid. When the user initiates a new puzzle, the new puzzle should appear in under 5 seconds. The newly created puzzles must be stored into database within few seconds so that the user can make the next move. When the user updates the puzzle, the system must recognize the change in 1 second or less. The puzzles should start appearing in new locations within 5 seconds of being assigned as a task. The task list must be updated in less than 2 seconds of the user being assigned the task. When the user starts a new scenario it must not take more than 15 seconds for the level to upload to the device. An increase in users on the system should not slow down the latency or network load which would cause delay user’s games.

### Precision or Accuracy Requirements

Puzzles must be shown depending on the scenario the player is currently playing. After solving a puzzle, user’s information must be updated instantly. difficulty level has to be set for the puzzles before the user begins as users of different ages should be more or less challenged. The accuracy of the headset and controller readings should be within 5 degrees as to not throw off the placement of the user’s head or hands inside The Time Machine. The information the system teaches the user shall be accurate as to not give the user false information about the simulation they are experiencing.

### Capacity Requirements

Connection can be 3G or have at least 5 Mbps speed.  GPS must be enabled in the headset when possible so data can be collected on where users are located when they use the device. Console should have at least 4 GB of RAM in order to run the graphics and physics of The Time Machine. The Console should have 8GB of hard disk available to store the application on the device. The Console must also have a 3-D graphics card to render the three-dimensional environments.

## Dependability Requirements

### Reliability Requirements

Data should be restored to the device if game is to be uninstalled. The system should start returning users at the last place they left off when they exited the scenario. The system should not crash under any circumstances since it is to be expected to be used by many people simultaneously. In the event of a crash, all user data should not be lost since it will be stored on separate servers.

### Availability Requirements

The game should be available to play for 99.99% of the time. The system needs to be updated to the latest version to get new features and scenarios. The system should update if reliable internet connection is found and the servers sense that the device is running on an older version of the software. The only time the system should be down is when the new updates are being published to the servers in order to be pushed out to all devices.

### Robustness or Fault-Tolerance Requirements

The game should store all user data to separate servers than the servers that the game is being run on. That way if the game servers were to go bad or corrupt, user data would not be lost. Once the system comes back on, the system will be able to access user data from these separate server. This is so user’s progress, error rate, time spent in games, and other information will not be lost.

### Safety-Critical Requirements

Follow the console guidelines for age restriction. Children below a certain age should not play the game as it might damage their sight. This is because virtual reality displays are very close to the user’s eyes. User’s should not play the game for longer than 2 hours straight without taking a break as to not tire out and damage the user’s sight. Stop playing immediately if overheating occurs in either the headset, controllers, or console.

## Maintainability and Supportability Requirements

### Maintenance Requirements

Information about total puzzles in a particular area should be visible to all users. Feedback of users should be reviewed and implement most requested features. Feedback from user will also help find bugs such as boundary issues and glitches that testers may not have found previously. Patches should be done in a weekly basis to make sure the system is cohesive. If there is a known time at which the system will be down for a major update, users should be notified at least 1 week in advance.

### Supportability Requirements

Testing phase should start as soon as new code is written. Testers should be actively looking for all the bugs in each current update. Descriptive error messages should be displayed at a time of fatal error as to help the tester understand what went wrong. An in game “Help” section should be created to let user know about components of the game if they have any confusion about the controls or settings.

### Adaptability Requirements

Game must be playable on all popular VR consoles on the market. The software should run smoothly on all the different platforms. The graphics should be almost identical on every platform depending on console graphics abilities. New language support will be added in every release so that the product can be used all over the world.

### Scalability or Extensibility Requirements

Servers should handle a significantly large server load. Thousands of people could possibly be trying to access the system at the same time so the system must be able to keep track of each user’s progress simultaneously. The database should be able to handle at least 50,000 concurrent users upon launch but will be able to handle at least 250,000 concurrent users after 3 years. Each server that user data is stored on will be able to hold data for 50,000 users. Initially we will invest in 3 servers being able to hoId 150,000 user profiles. in 3 years, we expect that we will need 20 servers holding 1 million user profiles.

### Longevity Requirements

Users will be prompted to provide a rating for the application 1 week after they download the software. The Time Machine will be maintained for at least 5 years after release. After the first 3 years, if the user reviews are very high, resources will be allocated to create a newer version of the game to be released for newer platforms while ending the old game servers after the first 5 year. If the game is marginally successful or not successful, the servers will be maintained until user count drops significantly enough that it is not worth enough to keep maintaining the system.

## Security Requirements

### Access Requirements

Users will be prompted to create a user specific profile that will be able to save their progress within their various time periods. Each user specific profile will have a unique username and password for account security. This combination prevents users from accidentally logging into other user’s accounts and progressing them without their knowledge. Users will therefore be able to use The Time Machine software by logging into their account whenever it is accessible to them. This will ensure that each user can save and track their progress on their own time. The user profiles will also allow users to access their accounts wherever they are whether it be from home or at school.

An example of the user creation process is as follows. A new user creates a profile with the username “Smith1” and the password “learning”. The user is then prompted for the username and password again via a login screen. Once the user logs in, he selects the ancient Greece time period. The user advances within the ancient Greece quest half way before having to leave to perform some task in the real world. The user will be able to save his or her progress and return to it at their convenience.

By giving each user a unique username, this prevents users from accessing each other’s private data since there is a one to one correspondence for usernames and passwords. Public accounts will be able to be created for educational purposes upon request. For example, if a teacher wants to create accounts for their entire class so that any student can pick up any device and not have to log in. These public accounts would be beneficial to younger users who might have a harder time with account creation and log in. This will be possible by creating accounts that are not password protected. Another possible way to create public accounts is to develop an educator’s version which does not require log in to use.

### Integrity Requirements

The database that is used for The Time Machine simulation must be sufficiently large enough to store every simulation that the user wants to participate in. This database must not only be sufficiently large, but it also must be able to maintain the saved information within each saved user profile. This is important because multiple users must be able to confidently save their progress within a scenario and the only way that is possible is if that saved data not only has integrity but also sufficient storage space. If users lose confidence in their ability to store their progress within quests, they will become less engaged with the product and their educational experience will suffer.

It is worth noting that the database for The Time Machine must be potentially very large to meet these requirements. This is something that the software engineers should take into consideration while they design the system. The burden of space could be potentially alleviated in part by saving only compressed versions of each user profile.

### Privacy Requirements

Privacy within The Time Machine will be ensured by allowing users to create a user specific password for their profile if they so desire. This password will be created at the same time the user profile is instantiated and will be used to unlock the account whenever the user returns to use the profile. Passwords must be between 6-14 characters long and must include at least one digit to make sure they are secure. The use of a password protected user profile will ensure that users do not have access to each other’s data. This will not only protect the privacy of each user that has a password protected account, but also ensure that users to not log into each other’s accounts on accident.

User 1 creates a profile with password “learning”. User 1 can return to the profile by typing the password “learning” upon entry. User 2 should not be able to use User 1’s profile without knowledge of their secure password, effectively protecting User 1’s data. This will allow users to learn at their own pace without fear of having their learning experience intruded on by other users. Also, this will prevent users from purposefully interfering with other user profiles in a malicious manner.

### Audit Requirements

The Time Machine software will be audited to ensure that it follows several different guidelines. These guidelines will implement rules that make sure the software is fast enough, safe enough, customized to each and every user, and able to save as much information as the end user finds necessary. To build a system that follow all of the necessary Time Machine guidelines including speed, customizability, and privacy. We do not foresee The Time Machine to elicit any sort of legal consequences resulting from breaching the privacy of other users, and therefore we do not see the need to seek legal counsel.

### Immunity Requirements

The Time Machine servers will be monitored by firewalls as a first line defense against attack. This will hopefully block most if not all malware attacks on the system. The user profiles that are stored on the servers will be encrypted as to not allow hackers to view the files. The scenario files that are stored on the servers will also be encrypted as to not allow malware to see the physical code and make it prone to attack. Every day at 12am and 12pm the system will automatically check for malware on the system that could be potentially dangerous to customer’s devices.

## Usability and Humanity Requirements

This section is concerned with requirements that make the product usable and ergonomically acceptable to its hands-on users.

### Ease of Use Requirements

The product should be easily usable by any child over the age of 7. This will be ensured by making the product simple to use. The product will follow in an iterative fashion so the user is not having to multitask and can focus on one important thing at a time. Many aspects of the game are to be explained visually within the simulation with heads up tutorials and instructions.

Example requirements include:

● Efficiency of use: Each user should be able to understand how to create a user profile and advance that profile throughout a quest on his or her first use.

● Ease of remembering: The user must only remember his or her username and password to reuse the product.

● Error rates: This product will allow a high volume of errors, as the user will be able to simple reset progress if they feel as though they have had too many errors.

● Overall satisfaction in using the product: The product should be satisfactory to users in such a way that they not only enjoy using the product more than using our competitor’s, but they also feel as though they learn more using our product than they learn otherwise.

These guidelines are necessary to ensure The Time Machine software results in a streamlined, smooth, enjoyable experience for the end user. We believe that the educational aspect of this product must be delivered smoothly so that the users easily receive it. The ability for children over the age of 7 to use the product sufficiently should be tested extensively and proven to be usable by more than 95% of all tested subjects. This will ensure that the software is delivered in a friendly, visually pleasing and user friendly manner.

### Personalization and Internationalization Requirements

The user will be able to customize the software in several ways, including the ability to set language preferences. The Time Machine software will sacrifice historical accuracy for this capability, and the user will be able to navigate the game using whatever language that he or she desires. This is necessary because we would like this product to be marketable internationally. If this product is going to be used by people of different cultures and ethnicities, then historical accuracy must be sacrificed to maintain a high level of user interaction. For example, Spanish users will be able to go back to ancient Greece and navigate the environment using the Spanish language while English users will be able to run the same scenario using the English language.

The language customizability consideration acts directly against historical accuracy, and for that reason the software does suffer somewhat in terms of educational value. However, we believe this to be necessary to reach as many users as we possibly can. The personalization of the language will allow the user to be able to absorb more information than if they had to read subtitles while listening to the culture in its native tongue.

### Learning Requirements

The Time Machine will be made understandable to its end users by using what we believe to be the software’s greatest asset, its ability to visualize environments for the user. The Time Machine will show the user exactly how to interact with the software in a variety of different situations, which will result in a positive immersive experience that will guide the user through every potential scenario.

We believe that the key to a positive experience using any sort of virtualization software lies within ease of use. We consider the fact that we have an incredible teaching tool at our disposal to be a blessing in pursuit of this goal, and we will use that blessing to improve our product. The product will visually show the user how to perform actions in game such as walking, navigating menus, how to solve puzzles, and interacting with the environment. This product may be most people’s first interaction with virtual reality since it is targeted at a younger user base and VR tech is fairly new. Giving them a clean, easy to use, straight forward gameplay they will feel much more immersed and invested in the learning experience.

### Understandability and Politeness Requirements

The accessibility requirements that will be fulfilled by The Time Machine software will include many different languages to create a broader customer base. This will allow the software to spread internationally, growing the customer base. For example, foreign speaking countries like China will be able to use it by changing the language to Chinese. Another requirement that will make the product easier to understand is the ability to change the difficulty. Younger users will need simpler problems and puzzles whereas older users will need harder more challenging problems.

### Accessibility Requirements

The requirements for how easy it should be for people with common disabilities to access the product. These disabilities might be related to physical disability or visual, hearing, cognitive, or other abilities.

To make The Time Machine more accessible, there are 3 settings the user can change. Users can change the brightness, volume, and font size letting the user set it to their own preference. The Time Machine will be accessible to users without the ability to walk since the joystick determines movement and selections are made with the buttons. The easier modes of the scenarios should be easy enough for users with learning disabilities or are mentally handicapped to enjoy the product.

### User Documentation Requirements

Users will first provide an email, first and last name upon account creation so that the system can be more personalized for the user. The user must provide this documentation if they want to use the product. As the user uses the software, information about the user is sent to the servers and stored under their own personal file.

Each system will come with a built-in instruction manual called “Help” that users can refer to if they have any questions about controls, how to change settings, and system requirements. We will provide this material to the user and make sure that it is simple and understandable so that a 7-year-old could understand it.

### Training Requirements

When the product is going to be used in a school as a learning tool, the instructor teaching the class should be well versed in the product. We will send a representative to the customer and show each of their teachers how to use the product. We will also be able to help the customers by having each of the teachers complete The Time Machine experience so that they know what the students will experience before they give the software to their classes. Teachers will more effectively be able to help students who are struggling to use the product.

## Look and Feel Requirements

### Appearance Requirements

The Time Machine software must look and feel high tech and superbly accurate no matter where it is being used. The time periods must look pristine and be fascinating in an aesthetic sense to capture the full appeal of its intended market. This will mean having incredibly powerful graphics within the game, as well as a powerful processor that will render those graphics. The need motivates the high-level graphics to fully engage the user within the educational experience. The diversity of each environment in which the user is given will make the user more engaged since it is something new each time. We believe that the better the graphics are, the more enthralled the user will be with the product, which will result in a higher educational experience.

### Style Requirements

The Time Machine software must make the end user feel as though they have been transported back in time to a different era. This will be done by ensuring that every part of the experience is styled perfectly and powerfully, from the graphics to the hardware itself. The gameplay of each scenario should reflect the atmosphere of the selected time. Making the user feel immersed in the experience and introduced to new cultures. The style of the hardware should be styled as though it were Alienware hardware, as their style fits our futuristic outlook. We believe that Alienware’s hardware styling is perfect for our product, and if our product can be made to appear like theirs, we will achieve a greater market share. Users will feel as though they are participating in a futuristic experiment, all while learning a great deal on their educational journey.

The product shall look like Alienware hardware, with neon lights decorating the vents where applicable and black and green used whenever possible. The hardware should not mimic Alienware’s style so closely that there is potential for a lawsuit for copying their intellectual property. Alienware’s unique style of hardware should be used only as a guideline, and should most certainly not be overtly copied. This futuristic approach should be mimicked inside of the software with the design of the menus and heads up displays being drawn in a futuristic style. This will make the user feel like they are visiting the past from the future and creating a contrast between old and new.

## Operational and Environmental Requirements

### Expected Physical Environment

The product is mainly intended to be used by students in school. The possible physical environment for the product can be a classroom, library, or home. The physical environments which we need to be concerned about are library and classrooms. When students use the machine, it is recommended that a trained instructor is advising the students in case they run into any problems they cannot solve or with technical help. The product should be able to work correctly on all student devices simultaneously. Since the product is designed to be used in a class room or library it must run quietly as to not disturb others around the user. Each time period should be designed to only last 15-30 minutes depending on the scenario so that students can complete an entire scenario in one class period.

### Requirements for Interfacing with Adjacent Systems

The product will connect and interface with servers for most of its functions. The product must be able to interact between The Time Machine servers and device operating systems without any glitches or fatal errors that could possibly crash the system. The product will also need to work with the different OS in VR systems. By effectively working on all VR operating systems, the data from each should be adapted and stored in each user profile the same way. This also means that users can sign in to the server from different operating systems and still be able to retrieve their personal data.

### Productization Requirements

The product will be distributed across all available app stores. It will be a paid application due to its high level of detail and design. Before the application is put on the market, select schools will receive free copies to test the product and provide feedback on how well it effects the learning experience for students. After the testing phase, vendors will be able to sell the application to schools or other educational establishments for an educator’s discount to offset the price of having to purchase the application for many devices.

### Release Requirements

The Time Machine will release new content twice a year. These releases will correlate with the beginning of each new school semester (Fall and Spring). This provides schools with new material to teach their students each semester so that they don’t have to do the same simulations repeatedly. With each new release of material there will also be updates to the user interface while making sure there are no bugs in the existing code.

## Cultural and Political Requirements

### Cultural Requirements

The product must not hurt religious sentiments and hurt any groups. Religion and oppression plays a large role in the history of the world so we must try to convey each as accurately and unbiased as possible. The product is not meant to push religious views on the user or make the user uncomfortable by disrespecting their religion. Also, the product must not have graphic imagery. The product is meant to be used in a school setting by young children so it would not be conducive to their learning experience by adding graphic imagery. Even though history can be graphic at times, it is not what we want our product to focus on.

### Political Requirements

The product must be unbiased towards international politics as to not offend or upset anyone. The scenarios are set in the past so there won’t be much to say about current politics but we must be careful not to convey the wrong message about a certain countries politics and policies that occurred during these historical periods.

## Legal Requirements

### Compliance Requirements

Users will be forced to accept a terms and conditions contract upon creating a new account. In this document, It will state that our company will be able to access your devices location and store users data indefinitely. It next will state that any malicious activity seen coming from the users IP address will result in a ban on the users IP address. The document will also convey that we will not sell your data to any third parties for profit. Once the user accepts our terms they will be able to use our product until we decide to update our terms and conditions.

We must also make sure that all the facts that we put into our system are true as to not be providing false information to all of our users. By providing false information we may offend a religion or ethnic group who may be able to sue us for slander.

### Standards Requirements

The environment in which the product will be developed will meet all of the requirements of workplace standards in the United States.

# Design

## System Design

### Design goals

In designing The Time Machine system, we have had various design tradeoffs that needed to be addressed in to make the product its absolute best. Without addressing these tradeoffs, we would not be able to make the product the best it can be for our customers and users. The first trade off that we must examine is educational value versus action. We need to find a good balance because if the software is too educational, users will think that it is boring. On the other hand, if the software is more action heavy, the users may not learn as much of the material defeating the purpose of the software. The Time Machine will try to lean more towards the educational side of the spectrum but not too far as to lose the interest of students. Another tradeoff that we had to consider was performance versus graphics. We want The Time Machine to be high resolution and aesthetically pleasing but we also don’t want the graphics to slow down the system too much. If the performance of the system is too slow to keep up with rendering the graphics, the long wait times and laggy play will outweigh the fact that the textures are very high resolution. Lastly, we must make sure that The Time Machine will have many features but not be too large of a project for our engineers to handle in the allotted time period. Late deployment of the product will make it harder to sell and put out into the field for students to use as a tool. This will cost more money for longer development times and raising the price of the finished product which we are trying to avoid.

## Current Software Architecture

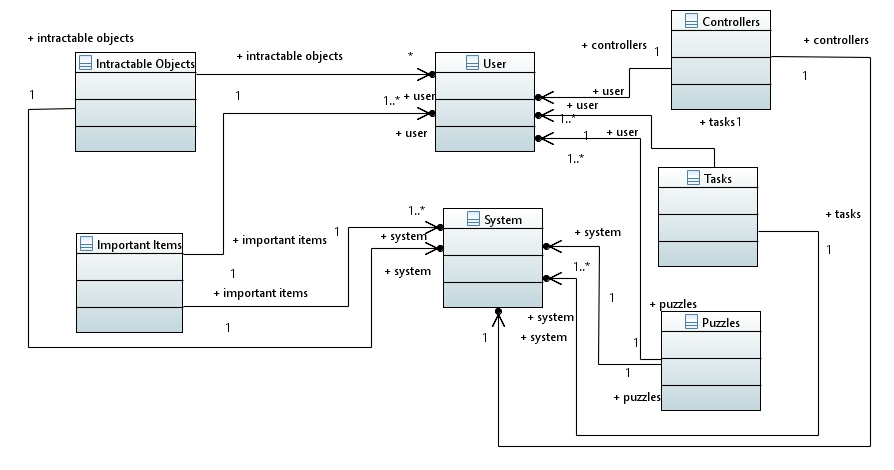
The architecture that we will base The Time Machine on will mainly consist of a client-server model. In this model the user will make requests on their end of the software which in turn will be sent to the server. The server then must decipher these requests and send the data back to the user to update what the user is seeing or the user’s status in the game.

## Proposed Software Architecture

### Overview

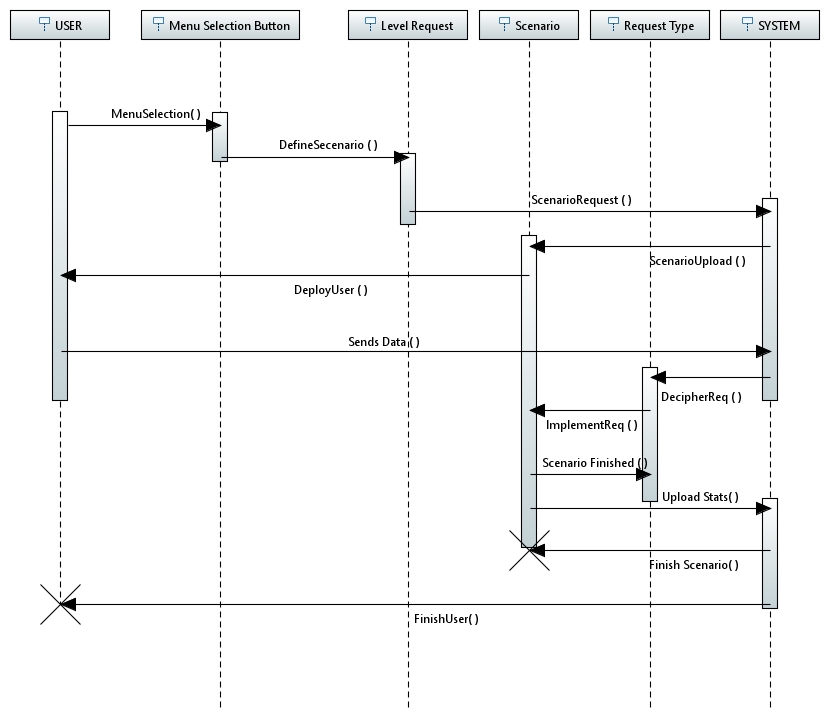
The finalized version of The Time Machine will consist of a combination of the client-server model and a 3 tier model. In this version of the software, the scenario will handle most of the users interaction request so that every request does not need to go all the way to The Time Machine servers and back to the scenario. This will decrease the load on the server substantially if there are more than one users accessing the system at the same time. If each user had to access the servers for each time that they moved their position or head, this could easily overload the servers.

### Class Diagrams



**Figure 5- Class Diagram**

### Dynamic Model

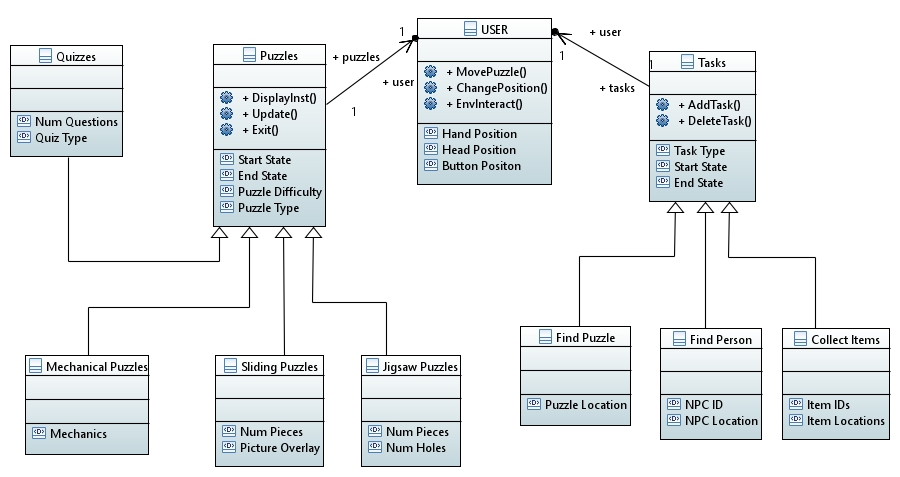


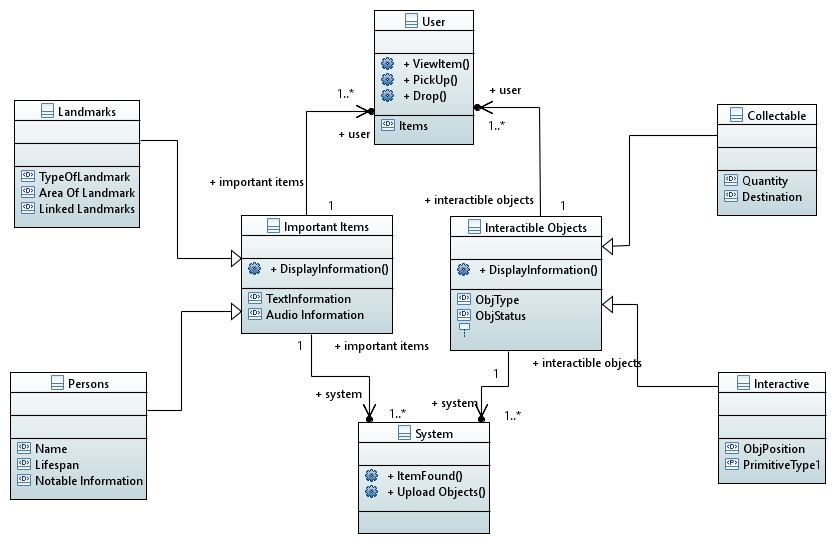
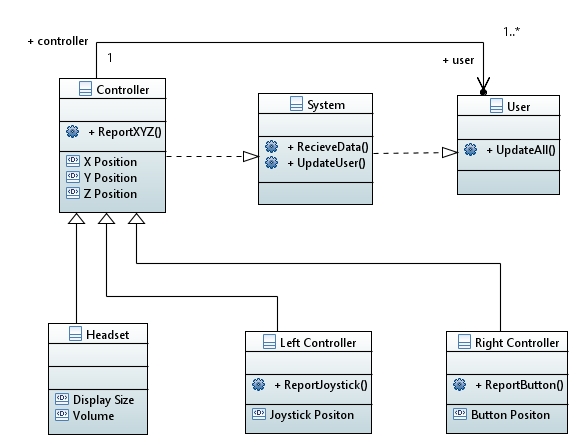
**Figure 6 – Sequence Diagram**

The initial use case diagram that we developed could be a little hard to read and had a lot of lines from the actors to show which actors would effect each use case. In this proposed sequence diagram, it is much easier to tell the timeline of events that the use cases follow. On the left side of the sequence diagram it shows the timeline of the user and on the far right side it shows the timeline of the system. The timelines in the middle show how each of the events progress the use cases from the start of the simulation until the end of the simulation. We first see that the user begins and then chooses a button from the main menu thereby sending a signal to the level request which will decipher the user’s choice and tell the system which historical event the user wishes to explore. The system then begins that scenario for the user and places the user in the scenario. Throughout the scenario the user sends data, whether it be movement requests or interaction request in which the server must be able to tell what updates to the scenario must be made. Once the scenario is finished, it must upload the users statistics to the servers to provide a scenario report in which could be later accessed by an instructor. After the statistics are saved the server must kill the scenario and then end the user’s activity within the scenario. Once we reach the bottom of the diagram the process would be repeated if the user chooses to start a new scenario.

### Subsystem Decomposition

The Subsystems that we wanted to take a closer look at are the puzzle and task system, important items subsystem, and the hardware subsystem. The first subsystem that we will discuss in the section is the puzzle and task system.

**Figure 7 – Puzzles and Tasks**

**Figure 8 – Items and Objects**

**Figure 9 - Controllers**

### Hardware / software mapping

**Figure 10 – Hardware/Software**

### Data Dictionary

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Description | Type | Date |
| User | The person who is currently using the software | Entity | 4/28/17 |
| UpdateAll | Updates the user based off all of the controller readings | Relation | 4/28/17 |
| ViewItem | User can view details about an item they have picked up | Relation | 4/28/17 |
| PickUp | User can pick up an intractable item | Relation | 4/28/17 |
| Drop | User can drop an item they have picked up | Relation | 4/28/17 |
| MovePuzzle | User moves the puzzle without moving their position | Relation | 4/28/17 |
| ChangePos | User changes their position in the environment | Relation | 4/28/17 |
| EnvInteract | User interacts with the environment | Relation | 4/28/17 |
| Items | An array of items that the user can fill with intractable items | Attribute | 4/28/17 |
| System | The Time Machine system that controls all events | Entity | 4/28/17 |
| UploadItems | Uploads all items and objects to the scenario | Relation | 4/28/17 |
| ItemFound | Removes item or object from the scenario | Relation | 4/28/17 |
| ReceiveData | Recieves data from all controllers | Relation | 4/28/17 |
| UpdateUser | Updates user based on data received from controllers | Relation | 4/28/17 |
| Puzzle | A game inside of the simulation that the user must solve | Entity | 4/28/17 |
| Start State | The initial position of pieces in a puzzle | Attribute | 4/28/17 |
| End State | The correct final positon of the puzzle | Attribute | 4/28/17 |
| Puzzle Difficulty | How hard the puzzle is in comparison to other puzzles | Attribute | 4/28/17 |
| Puzzle Type | The type of the puzzle | Attribute | 4/28/17 |
| DisplayInst | Displays instructions to solve the puzzle | Relation | 4/28/17 |
| Update | Updates the location of puzzle pieces | Relation | 4/28/17 |
| Exit | Exit the puzzle when puzzle reaches end state | Relation | 4/28/17 |
| Jigsaw Puzzle | A puzzle with pieces that either fit together or in designated slots | Entity | 4/28/17 |
| Num Pieces | The number of pieces in a puzzle | Attribute | 4/28/17 |
| Num Holes | The number of slots in which puzzle pieces will fit | Attribute | 4/28/17 |
| Picture Overlay | The picture that will be formed when a puzzle is completed | Attribute | 4/28/17 |
| Sliding Puzzle | A puzzle in which a grid of pieces has one missing. The user slides the pieces to form a picture | Entity | 4/28/17 |
| Mechanical Puzzle | A puzzle that has different pieces that requires manipulation by the user to solve | Entity | 4/28/17 |
| Mechanics | The physics of a mechanical puzzle | Attribute | 4/28/17 |
| Quiz Type | Type of quiz (Matching, trivia) | Attribute | 4/28/17 |
| Quizzes | A puzzle in which users solve questions about the scenario | Entity | 4/28/17 |
| Num Questions | Number of question within a quiz | Attribute | 4/28/17 |
| Tasks | An objective the users must complete | Entity | 4/28/17 |
| Task Type | The type of task to be completed | Attribute | 4/28/17 |
| AddTask | Adds the task to the task list | Relation | 4/28/17 |
| DeleteTask | Removes task from the task list | Relation | 4/28/17 |
| Find Person | A task in which a user must find an important historical figure | Entity | 4/28/17 |
| NPC ID | Identification number of a non-playable character | Attribute | 4/28/17 |
| NPC Location | Position in the environment where the non-playable character is located | Attribute | 4/28/17 |
| Find Puzzle | A task in which a user must find a puzzle in the environment | Entity | 4/28/17 |
| Puzzle Location | Position in the environment where the puzzle is located | Attribute | 4/28/17 |
| Collect Items | A task in which a user must collect items in the environment | Entity | 4/28/17 |
| Item IDs | Identification numbers of the items to be collected | Attribute | 4/28/17 |
| Item Locations | Position in the environment where the items are located | Attribute | 4/28/17 |
| Controller | A piece of hardware used to control the users character | Entity | 4/28/17 |
| X Position | The X coordinate of a controller | Attribute | 4/28/17 |
| Y Position | The Y coordinate of a controller | Attribute | 4/28/17 |
| Z position | The Z coordinate of a controller | Attribute | 4/28/17 |
| ReportXYZ | Reports the new X, Y, and Z coordinates of the controllers to the system | Relation | 4/28/17 |
| Headset | Controller used to view the environment | Entity | 4/28/17 |
| Display Size | The size of the display within the headset unit | Attribute | 4/28/17 |
| Volume | How loud the speakers in the headset are | Attribute | 4/28/17 |
| Left Controller | Controller containing a joystick to move player’s position | Entity | 4/28/17 |
| Joysitck Position | The current position of the joystick (up, down, left, right) | Attribute | 4/28/17 |
| ReportJoystick | Reports joystick location to the user | Relation | 4/28/17 |
| Right Controller | Controller containing buttons to interact with the environment | Entity | 4/28/17 |
| Button Position | The current position of the button (pressed or depressed) | Attribute | 4/28/17 |
| ReportButton | Reports button state to the system | Relation | 4/28/17 |
| Important Item | Items that upon viewing, the user is provided a heads up display | Entity | 4/28/17 |
| Text Information | Text that will appear in the heads-up window | Attribute | 4/28/17 |
| Audio Information | Audio that will play when important item is found | Attribute | 4/28/17 |
| Landmark | Important historical places, building, and architectural feats | Entity | 4/28/17 |
| TypeOfLandmark | Type of landmark stated under landmark definition | Attribute | 4/28/17 |
| AreaOfLandmark | Area that displays information when viewed | Attribute | 4/28/17 |
| LinkedLandmarks | Landmarks related to current landmark | Attribute | 4/28/17 |
| Persons | Important historical figures | Entity | 4/28/17 |
| Name | Name of the important person | Attribute | 4/28/17 |
| Lifespan | Year born and died of important person | Attribute | 4/28/17 |
| Notable Information | Important facts about what the important person did | Attribute | 4/28/17 |
| Intractable Item | An item that the user can interact with | Entity | 4/28/17 |
| ObjType | The type of object found by the user | Attribute | 4/28/17 |
| ObjStatus | The current status of the item | Attribute | 4/28/17 |
| Collectable Item | An item the user can use to complete tasks | Entity | 4/28/17 |
| Quantity | How many of a collectable item there is | Attribute | 4/28/17 |
| Destination | Where the collectable item needs to go | Attribute | 4/28/17 |
| Interactive Item | An item that the user can use in the environment | Entity | 4/28/17 |
| ObjPosition | Where the object appears on the display when in use | Attribute | 4/28/17 |
| DisplayInformation | Displays critical information about an object or item | Relation | 4/28/17 |

### Persistent Data management

When each player creates their profile, a directory is created on one of The Time Machine servers that will hold all information about an account. When the user finishes a scenario, a statistic report is uploaded to their directory, thereby effectively storing a user’s complete timeline of operation. When instructors view this information, it will be seen as a clearly laid out pdf file that will be easy enough for instructors to understand.

If a player decides to end their session early and save their progress, a “current session file will be saved in their directory. Thus, when they return to play, they will be given an option to upload last save or start a new session. If they start a new session, the file will be deleted and no statistics will be uploaded to the file.

### Access control and security

|  |  |  |
| --- | --- | --- |
| Actors  Objects | User | System |
| Headset |  | ReportXYZ |
| Left Controller |  | ReportXYZ  ReportJoystick |
| Right Controller |  | ReportXYZ  ReportButton |
| Interactive Items | ViewItem  PickUp  Drop  DisplayInformation | Display Information  ItemFound  UploadObjects |
| Important Items | ViewItem  DisplayInformation | DisplayInformation  ItemFound  UploadObjects |
| Puzzles | MovePuzzle  DisplayInst  Update  Exit |  |
| Tasks | AddTask  DeleteTask |  |

### Global software control

We will be able to control all of the software from our offices via admin accounts that allow for system maintenance and monitoring. Administrators can log into the system via their own computers without the virtual reality headset on. This way they can see the different statistics such as how many people are on the server, where users are accessing the server from, and any reported bugs that may have incurred. With this information, administrators can find the appropriate time to fix the servers or upload new content since they will be able to see how many people it will be effecting.

### Boundary conditions

We want to make sure that user data will not be lost upon system startup and system shut down. Users will have the choice to save their current progress if they need to leave the simulation for any reason and wish to return to their game later. We also want to incorporate an autosave feature that will automatically save the users progress after each task completed. This will help in cases in which the user’s equipment powers down improperly or if the user is removed from the scenario due to connectivity problems. In the case of either of those situations, the user would just have to reconnect to The Time Machine and be able to start reasonably close to where they left off.

Another boundary condition is we need to make each environment no bigger than what the system can efficiently handle. By creating a set, maximum size for scenarios, we will not run into performance issues as often. The maximum size for scenarios will consider surface area created, the number of NPCs, the number of objects and the number of puzzles. The set maximum will help developers to see how much space they are using and update the scenario accordingly.

## Subsystem services

As you can see in the first subclass diagram, the user can have a one to one ratio for puzzles and a many to one ratio of tasks. This means that users can only have one puzzle assigned to them or open at time. On the other hand, Users can have multiple concurrent tasks assigned to them that will be stored in the task list.

Focusing on the puzzles, we will incorporate 4 main kinds of puzzles. Each of these puzzles will have a start state, end state, and difficulty. First, there will exist jigsaw puzzles. In this type of puzzle, users will have pieces that need to fit together correctly or fit in holes to complete an image or item. The next type of puzzle that we developed are sliding puzzles. These puzzles will be scrambled images broken into several pieces with one slot missing. The objective of the user will be to move the pieces into the empty slot until the correct image is displayed. Third, there will be mechanical puzzles. A common example of a mechanical puzzles are Rubix cubes. In these puzzles, the user will have some sort of object or set of objects to manipulate to achieve a final state. The last kind of puzzle is a quiz in which users will be subjected to 1 or more questions that they will have to answer correctly. This doesn’t seem like a puzzle but it correctly fits our main class of puzzles in which it has a start and finished state as well as the user makes various “moves” during a quiz.

Tasks will similarly have a start and finish state, but unlike puzzles, the user must explore the environment in to solve them where as when users are in puzzles their character will be stationary. The first type of task is a find person task. In this task, users will have to find either another civilian NPC or an important historical figure NPC. Each of these NPCs will have a position and ID so that the system can recognize when the user reaches their target. Like the find person task, there will exist a find puzzle task in which a user must find the location of a puzzle to complete. Once the puzzle has been completed, the task will go away. This is so that if a player does not finish the puzzle, he/she will be able to find the puzzle again later in the simulation. The final type of task is one in which the user must explore the environment to collect items. This task ties together the first two since certain NPCs might need the user to find certain items for them. An example of this might be that an NPC might need the player to find pieces of a map in which upon retrieval will reveal the location of a puzzle that the user needs to solve.

The second diagram represents the system of the Important Items Class and the Intractable Items class. Just as a reminder, the important items are items that once the user looks at them, a heads-up display provides information on the item that they are looking at. Intractable objects are objects that the user can either pick up or manipulate to further their progression in the simulation. The system and the user can both have more than one number of important objects and items associated with it. The items and objects can only be accessed by the one system and one user though. This shouldn’t be a problem since this is a single player game so we won’t have to worry about items being stolen by other players.

The important items class can be split into two subclasses, Landmarks and Persons. Landmarks include important building, places, rivers, architectural feats, etc. Persons will include people who played an important role during the selected time period. Once a user views one of these items, the displayed box must provide concise, detailed facts about the person or place such as its name, lifetime, and accomplishments or why it’s important.

\Intractable objects closely relate to the important items since they too need to provide additional information to the users. The two subclasses of the intractable items class are collectable items and interactive items. Collectable items will be “stored’ by the user class under an array of items that are currently being held by the user. When the user drops the items by bringing them to their correct location, the items will be freed from this array. Information about the item will be displayed once the user picks up the item via a heads-up display window. Interactive items are specialty items that the user can move around the simulation or be able to use in their current environment to further the scenario. For example, if the user was playing the Boston Tea Party event in the American Revolution scenario, we would want the user to be able to pick up crates of tea and throw them off the sides of the boat.

The third model represents how the hardware components interact with the system and the user. We first started with a controller class since all three of the controllers will have to send their X, Y, and Z coordinates to the system and user. Then from this, each of the controllers has their own unique functions that the others do not. The headset that contains the speakers and display, will have to be able to tell the system what size the display is depending on the device the user is using. The left controller must track the position of the joystick and then send this data up the chain in to tell the system whether to move the position of the user. The right controller must tell if the user has pressed a button on the controller and then send this signal to the system to interact with the environment. The system class is dependent on the controller readings and the user is then dependent on the system to update what the user can sense.

## User Interface

We want the user interface to be easy to guide through in the menu screens and aesthetically pleasing to the eye while whilst partaking in a time machine simulation. This will entail having a well-designed UX for entering profile information as well as entering a simulation, and a UX with an optional HUD while within the simulation. The optional HUD will allow users to take some time with helpful tips displayed to the user so that learning how to use The Time Machine is a smooth and seamless process. The idea behind the optional HUD is that it will become less and less intrusive as the user dives deeper into the simulation.

Throughout the user’s time within The Time Machine, we want the majority of the experience to look beautiful. We believe that less HUD translates to more beauty, and therefore we only want The Time Machine to include helpful information when necessary. For example, user encounters a boat on a river within the simulation for the first time ever. A helpful tip is displayed which instructs the user how to get into the boat and control it. The user is then prompted, asking whether he wants to keep receiving this tip. Some users won’t mind the reoccurring tutorial but for more experienced players, they may not need to be reminded each time they get into a boat how to control it.

## Object Design

### Object Design trade-offs

The major design trade-off that will be encountered while The Time Machine is being developed will be regarding how well The Time Machine performs vs. how flexible The Time Machine is. The more flexible Time Machine is, the worse the performance would be and vice versa. We allowed this trade-off as we want The Time Machine to be able to reach as deep of a market as we see possible. We want anybody with a reasonable operating system to be able to use The Time Machine as an educational tool. Therefore, we will be incorporating graphics settings that will be adjustable on each individual Time Machine system. End users will be able to turn shadow rendering up or down, depending on how strong they want their graphics to perform making it easier on their machines and enhancing their gameplay.

### Interface Documentation guidelines

Each individual module must be documented sufficiently so that other developers are able to approach The Time Machine simulation and add on to the existing work if they so desire. We want to establish an open-source library of Time Machine simulations that users can download and explore if they so desire. Part of the motivation for this decision comes from a potential shortage of storage space on each machine. If users can download only what they desire, that problem will be alleviated. A potential situation might be that a user in Mexico creates a Time Machine module that covers the Aztecs. A user in San Diego can download that module and add on to it without having to download other scenarios that may not apply to their curriculum. Thus saving the users total space on their machines.

### Packages

Packages within The Time Machine simulation must cover parts of the development that will be used widely throughout many simulations. For example, there must be a package for adding water to a scene, such that it will be added and follow the laws of physics accurately. This will ensure that that physical objects appear and act more or less the same throughout various different Time Machine simulations. For example: water, grass, trees, etc. Some aspects will not be able to be replicated, and in such cases, multiple packages can be used for similar things.

A developer wants to make a pond in a simulation. He should be able to create a pond that appears and acts similarly to any other ponds that have been made using that same package. If different packages were used in different simulations, we would not have a consistent product which user may not approve of.

### Class Interfaces

Classes within The Time Machine should all become autonomous once the system begins initializing the selected scenario. Depending on the historical event and the difficulty settings selected, each of the classes will be filled with their predetermined data. Once they are filled with data, the objects can be manipulated only after the user initiates a defined function to change the class.

# Test Plans

## Features to be tested / not to be tested

The main features to be tested are:

* User is able to save progress and load progress
* User is able to interact with environment
* Important facts pop up when looking at an important item
* User does not glitch or walk through set boundaries
* Puzzle is exited after completion
* User task list is updated after obtaining or finishing a task
* Scenario is exits to main menu when no more tasks exist
* Multiple users can be using the system simultaneously

## Pass/Fail Criteria

For a test to pass, it must be able to pass the same test 100,000 times in a row for it to be a success. This guarantees users that the test case was tested thoroughly before it was released to the public. A failing test can be identified several ways. If the system crashes after test case, the test is a fail. If the system lags, glitches, or takes an excessive amount of time to complete a task, then it is considered a fail since it is not up to par without standards

## Approach

The first part of the software that needs to be tested is the back-end code. This is code that handles hardware readings, user mobility and physics of the game. Back end code also is important for saving and loading scenarios. Once this has been tested, we will focus on testing front end code such as the visuals and boundaries of the three-dimensional environment.

## Suspension and resumption

Testing may be suspended at certain times throughout the development process. If there are many bugs, the developers must take care of as much as they can to provide testers with a smoother testing experience. Once the developers believe that their code is working correctly, testing will resume to check to see if the bugs have been patched.

## Testing materials ( hardware / software requirements )

When testing The Time Machine software, it will be tested on each different VR platform. This will ensure that when the product is released, the product will be fully functional on all virtual reality operating systems. These platforms will each be updated with the most current operating system version and the most current version of The Time Machine software

## Test cases

Test cases are as follows:

Test Case Name: User may save at any point and it will be reflected in database

Entry Condition: User is inside of a scenario and has not yet finished all tasks

Flow of Events:

1. User is inside simulation
2. User accesses in game menu
3. User selects save button
   1. User is prompted that game has been saved
   2. Game data is sent to servers
4. User is exited from menu

Exit Condition: The user’s save data is stored in the database

Test Case Name: User may upload progress from the database via the main menu

Entry Condition: The user is at the main menu and has not started a new scenario

The user has a previously saved scenario in the database

Flow of Events:

1. User turns on the system
2. User logs into their account
3. User presses load button
4. System loads previously saved progress to device

Exit Condition: The user is inside of the past saved scenario

Test Case Name: Important item pop ups can be exited or time out after 30 seconds

Entry Condition: The user is viewing an important item

Flow of Events:

1. Heads up display window appears
2. User reads information from window
3. User presses close button
4. If user does not press close button, window fades away after 30 seconds

Exit Condition: Heads up display is clear of window

Test Case Name: User must check each wall and object to see if the boundaries are correct

Entry Condition: User is inside of the scenario

Flow of Events:

1. User moves position into wall or large, stationary object
2. User cannot move through the object

Exit Condition: User is not inside of object or through wall

Test Case Name: If all tasks are complete, the program should return to main menu

Entry Condition: User has no tasks in task list and no more tasks exist in the scenario

Flow of Events:

1. User finishes last task
2. System counts how many tasks are left
3. If there are 0 tasks left, the system ends the scenario

Exit Condition: User is returned to the main menu

Test Case Name: System should count amount of user errors per scenario

Entry Condition: User finishes the scenario

Flow of Events:

1. User partakes in many puzzles during the scenario
2. Each mistake the user makes is accumulated
3. When scenario is finished, error count is reported in the statistics

Exit Condition: User database file is updated

Test Case Name: System should count time spent in scenarios

Entry Condition: User finishes scenario

Flow of Events:

1. User starts scenario
2. Timer starts
3. User finishes scenario
4. Timer ends
5. Timer count is reported in the statistics

Exit Condition: User database file is updated

Test Case Name: After getting a task, it should be reflected in task list

Entry Condition: User is inside of the scenario

Flow of Events:

1. User triggers the start of a task
2. User is explained what the task is
3. System adds the task to the user’s task list

Exit Condition: Task appears in task list

Test Case Name: After completing a task, it should be reflected in the task list

Entry Condition: User is inside of the sceanario

Flow of Events:

1. User finishes all of a certain task’s parameters
2. System removes task from task list

Exit Condition: Task does not appear in the task list

## Testing schedule

The project will be tested on a weekly basis to ensure that good progress is being made. If testing goes well that means that there are little to no bugs so testing will not be suspended until there are too many bugs in the software. If there are too many bugs, the testers will not get the full experience and it will be difficult to use.

# Project Issues

## Open Issues

An open issue that we are facing is the potential for limited storage space. As each simulation will require a lot of memory, we must find a way to save storage space. One way we will do this is by making many of the simulations optional.

This will allow users to pick and choose how much memory they want to take up on their system. This is important, particularly within the educational market that we are attempting to penetrate, as many educational institutions do not have enough funding to purchase an excess of memory. A consideration for this issue is that some users will opt to store many more simulations than others if they are not space constrained. The end user will also be able to opt into having many more different user profiles, each of which will take up more memory.

A simulation takes 5gb. A user with 100gb could store 20 simulations. A user with only 50gb could store only 10. Or, a user profile takes 5 gb, and a 100 gb machine could have 20 user profiles. If a classroom or school has more than 20 students in a class, this could be problematic, especially if the school cannot really afford more than one VR device since they can be costly.

## Off-the-Shelf Solutions

### Ready-Made Products

We want The Time Machine to come with several Ready-Made products, i.e. several simulations that are developed by our own personal developers before we open The Time Machine to the open-source world.

We don’t want to have to depend on developers that are not on our payroll to create great content. This would be far too large of a risk to take, and we would much rather prefer to provide the end user with a solid amount of high quality simulations upon initial purchase. We will create a main storyline that the user can follow for each time period, while still leaving room for open source developers to make a big impact. For example, open source developers could focus exclusively on side quests. The main storylines of the American Revolution will be created by in-house developers. However, open-source developers are free to create storylines that branch off from the main scope of history. Open-source developers are also not going to have as many resources as we do and the expertise of our client, so our paid DLC will still be of better quality than user created levels.

### Reusable Components

A library of facts and information that we have compiled could be used by open-source developers to give their scenarios accurate information to use so that they don’t provide false information to other users such as younger users who will believe the fake fact.

### Products That Can Be Copied

We want every simulation to feel as though they have entered their own personal world, like large single-player RPG games such as Elder Scrolls or The Legend of Zelda. If we copy these games effectively, the user will be able to feel an immersive experience that intrigues them and motivates them to learn at the same time. This will provide a hearty experience for the user, and will encourage them to have fun while learning at the same time. Obviously, each simulation will not be designed as in-depth as massively successful RPG games. However, this should be strived for. We would like to see things such as open world exploration, side quests, and character customization within the design. Current RPG games revolve around one world with one common theme whereas our product will be able to replicate many different locations and time periods. Users should be able to be transported to the medieval times in a simulation and interact with NPC’s and the environment around them like in an RPG but then be able to turn around and explore the ancient roman empire seamlessly.

## New Problems

### Effects on the Current Environment

It is important for the application to perform consistently at high speeds while running many concurrent simulations. The Time Machine servers should be able to handle the load efficiently without disturbing user’s gameplay as much as possible. Since the product is meant to be used in a classroom setting, major problems could interrupt all classrooms causing problems for students and teachers.

### Effects on the Installed Systems

Previously installed systems should not be affected after installation of the product since we will be building most of our software for the game and servers from the ground up. In doing this, we must make sure that the newly implemented designs are cohesive and are able to communicate with each other well.

### Potential User Problems

The goal of the product is to give users the most immersive and entertaining simulation possible while keeping the product grounded with an educational background. A potential user problem could be we won’t know what balance of action and education will best suit our clients needs. Therefore, there should be a way to track the issues that users may have with our product. Before we deploy our product, we should test the software on various age groups to see how each feel about the product. These testing groups will be able to tell our staff what the difficulty of the game is, if it kept their interest or not, and whether they learned anything from completing the simulation. This will also benefit in the design of future scenarios.

### Limitations in the Anticipated Implementation Environment That May Inhibit the New Product

If the system has an issue with data corruption, there should be safeguards in place so we do not lose all our scenarios or user data. To combat this issue, we must have standby servers that are written to once per week that are not directly connected to the main system. That way if there was a system wide corruption or crash, all user data from that particular week may be lost, but not our whole program or user records.

Another problem that we may encounter is a network issue in which there is too large of a bandwidth load on a user’s or our own network. At that point, the game should pause and display a pop up window that states that there is a connectivity issue which will disappear once connectivity is resolved. The user is then started exactly where it left off before the connectivity issue occurred.

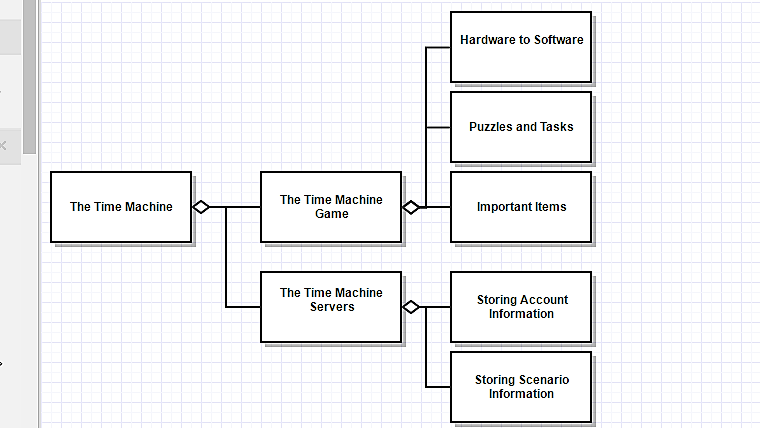
### Follow-Up Problems

The game is designed to handle a player of size 100,000. As the demand increases, the server load and data management will need to change. As stated above in the design portion of these plans, we hope to be able to integrate the gameplay code into the scenarios to free up server loads on the system. This may incur longer load times though since all the mechanics of the game will have to be uploaded with the levels as well.

## Tasks

### Project Planning

The structure present in above should be enough to develop a good quality product. It will be developed using Agile methodologies, and with the latest tools available in the industry. This will make the product adhere to technological changes.



**Figure 11 - WBS**

### Planning of the Development Phases

Objectives will help our developers throughout the development phase. Initial development activity: The development team of this product begins by verifying selection for standards, methods, and tools that will be utilized in all the future phases.

**Design Phase:** In this phase, we determine the hardware and network development environment as well as installing this software on all developer machines. We will also create test cases before writing code. A few examples of these may be test procedures, check network load, hardware compatibility, gnu usage, etc. We also must determine the design pattern the software will follow.

**Development Phase:** This is the phase where developers will have short, regular meetings before completing assigned objectives. This is to keep everyone working fast while being able to know what all developers are doing and can change requirements on the fly.

**Testing Phase:** After Development, The Time Machine software will need to be thoroughly tested to make sure that it meets all specified requirements. If the product does not meet all the requirements it will go back into the development phase

**Integration Phase:** Network and hardware components should satisfy the requirements defined in the design phase for the product. Completion of documentation will also occur in this phase in which we will have to provide a release announcement, game manual, maintenance manual, etc.

## Migration to the New Product

### Requirements for Migration to the New Product

Whenever an update is available, older versions of The Time Machine should be removed from the actives servers. This will free up a lot of memory considering that the full product will take up a lot of space on its dedicated server. This will also resolve any problems with compatibility since the updates should be fully functioning. If one piece is updated to an older version there may be some compatibility errors. Updates shall occur mainly around midnight on Fridays since school is not in session and younger students are our main target users. That way if the new update is not up to par with the old system, developers have until Monday to solve the problem and upload the newest version of the software.

### Data That Has to Be Modified or Translated for the New System

In each update, the first and foremost thing that will be taken care of is bug fixes. We want our program to run as smoothly as possible before we start to add more scenarios and functionality to the software. In the updates, we can also add components such as more tasks for users to complete, more important items and historical figures the users can learn about. Every update should be its own module to efficiently upload to the servers and cause less downtime. If there is a new development in VR technology, such as a new machine with better/worse performance, an update will have to occur which makes the software compatible with these machines

## Risks

## The first type of risks that need to be addressed when developing our product are those that could affect our schedule. We want to make sure that our development stays on track so this is a very important factor in planning the timeline of development. We want to make sure that our proposed timeline is realistic. If the schedule is unrealistic, it may result in more bugs, missing features, inaccurate facts and delayed release.

## The second type of risks that we need to address are risks that involve meeting our requirements. We want to make sure that all our functional and non-functional requirements are met to provide the users with a solid experience. We also must take into consideration that the requirements may change and adapt as development progresses continues. To combat this, we should have meeting every week to see if we are still on track with our requirements and to see if any requirements need to be added or removed. We want to make sure that the product will work as a complete system and that we produce the best quality product for our client and customers.

## Costs

A function point is a "unit of measurement" to express the amount of functionality a product can provide to a user.

Caper Jones rule of thumb to calculate the effort

Effort in staff months = (function points ÷ 150) x function points0.4

● 12 of input and output flows on the work context

● 12 of business events

● 12 product use cases

● 17 functional requirements

● 34 nonfunctional requirements

● 6 requirements constraints

● 93 function points

Effort required according to Caper Jones rule:

Calculated using 93 function points.

Effort in staff months ≈ 3.8 months

## Waiting Room

Requirements for Future Releases:

Requirement #: 1 Requirement Type: Waiting Room Version no:2

Description: Product must provide option to choose language of interaction

Rationale: Making product available in multiple languages

Originator: Jairaj Singh Shaktawat

Customer Satisfaction: Customer Dissatisfaction:

Supporting Materials:

History: 4/27/17

Requirement #: 2 Requirement Type: Waiting Room Version no:2

Description: Adding region specific new content for different parts of the world

Rationale: Make product appealing to International Markets

Originator: Jairaj Singh Shaktawat

Customer Satisfaction: Customer Dissatisfaction:

Supporting Materials:

History: 4/27/17

Requirement #: 3 Requirement Type: Waiting Room Version no:2

Description: Introducing a mature content version of The Time Machine

Rationale: Make product appealing to the young adult market. The software can very well . provide a Game of Thrones type experience.

Originator: Jairaj Singh Shaktawat

Customer Satisfaction: Customer Dissatisfaction:

Supporting Materials:

History: 4/27/17

Requirement #: 4 Requirement Type: Waiting Room Version no:3

Description: Introduction of “What-if” scenarios for some historical events

Rationale: Product also aims to give users a different perspective about what would have . happened if the events didn’t pan out as they did.

Originator: Jairaj Singh Shaktawat

Customer Satisfaction: Customer Dissatisfaction:

Supporting Materials:

History: 4/27/17

Requirement #: 5 Requirement Type: Waiting Room Version no:3

Description: Open source level development

Rationale: We want some older users to be able to develop their own versions of the simulation and side quest to fill the time gap in between inhouse releases.

Originator: Eric Heilman

Customer Satisfaction: Customer Dissatisfaction:

Supporting Materials:

History: 4/28/17

## Ideas for Solutions

One problem that we encountered is determining the correct difficulty to age ratio. The solution must also include an option for the user to choose the version of the lesson which suits his/her educational needs. We don’t want to teach 8th grader 4th grade history and vice-versa. A solution to this is having the user select their grade level upon account creation and thus be able to adapt to their experience.

Another problem we encountered is trying to develop the system to work on all platforms. With that being said we do not want to spend too much of our resources making the game too high performance and not too low performance. We decided to combat this issue with a quality adjustment in the settings menu as to not take up too much time of the developers who could be working on more important things.

## Project Retrospective

The project gave us a perspective about the complexity of the building a VR simulation product. The project in a very specific in its goals and all the things are in place for its successful implementation. It was interesting to try and develop a fully functional piece of software that could one day benefit the lives of children across the world. The initial goal of the project was to bring people together and promote world culture and I feel as though we achieved this goal through our combined effort. It was very difficult to figure out what each of the systems of the project were and how they would all communicate with each other but in the end, everything seemed to work itself out into one cohesive system. The requirements for the project are very specific and detail oriented. The cost is defined in very specific numbers. The constraints and challenges are well defined. This project will help expand this field of historical virtual reality. Hopefully the model for this system could help others in the effort to build their own version and have a better understanding of how what requirements need to be met.

# Glossary

The glossary defines terms that may not be familiar to all readers. This is especially important if the document is expected to reach a wide and varied audience, such as school children. The glossary may be placed at either the beginning or the end of the document.

**Virtual Reality:** A three-dimensional computer program in which a user uses a headset with a display inside of it in order to view the three-dimensional space.

**VR:** Stands for virtual reality.

**Hardware:** Physical tools that are used to interact with computer code.

**Software:** A system of computer programs that result in a final product.

**Server:** A device on a network of computers that handles the storage of resources/data

**Glitch:** When a software component malfunctions

**Bugs:** See Glitch

**Heads-Up Display:** A set of items that the user can see that isn’t part of the environment but contains important facts and information.

**HUD:** Stands for Heads-Up-Display

**RPG:** Stands for Role-Playing Game

**DLC:** Stands for Downloadable Content

# References / Bibliography

|  |  |
| --- | --- |
| [1] | Robertson and Robertson, Mastering the Requirements Process. |
| [2] | A. Silberschatz, P. B. Galvin and G. Gagne, Operating System Concepts, Ninth ed., Wiley, 2013. |
| [3] | B. Bruegge A. Dutoit, Object-Oriented Software Engineering Using UML, Patterns, and Java Third ed., Pearson, 2010 |
| [4] | M. Fowler, UML Distilled, Third Edition, Boston: Pearson Education, 2004. |

# Index

This section provides an index to the report. The sample below was generated using the “Mark Entry” and “Insert Index” items from the “Index” section on the “References” tab, and can be automatically updated by right clicking on the table below and selecting “Update Field”. To remove marked entries from the document, toggle the display of hidden paragraph marks ( the paragraph button on the “Home” tab ), and remove the tags shown with XE in { curly braces. }

Design 61, 65

Requirements 36, 53, 60

Test 66, 67