**Software Requirements Specification**

**for the**

**UMBC Virtual Tour 2.0 System**

**Document # CMSC447-05-FA2018-G03-SRS-01A**

Revision A

26 October 2018

Table of Contents

[**0.1 26 October 2018: CMSC447-05-FA2018-G03-SRS-01A** 3](#_Toc528360955)

[1 Scope 4](#_Toc528360956)

[**1.1 Identification** 4](#_Toc528360957)

[**1.2 Definitions, Acronyms and Abbreviations** 4](#_Toc528360958)

[**1.3 References** 4](#_Toc528360959)

[**1.4 Document Overview** 5](#_Toc528360960)

[2 Overall Description 5](#_Toc528360961)

[**2.1 System Purpose and Perspective** 5](#_Toc528360962)

[**2.1 System Functions** 6](#_Toc528360963)

[**2.2 User Characteristics** 6](#_Toc528360964)

[**2.3 Constraints** 7](#_Toc528360965)

[**2.4 Assumptions and Dependencies** 7](#_Toc528360966)

[3 Requirements 7](#_Toc528360967)

[**3.1 Required states and modes** 7](#_Toc528360968)

[**3.2 CSCI Functional Requirements** 7](#_Toc528360969)

[**3.2.1 Virtual Tour Interface CSCI (VTI)** 7](#_Toc528360970)

[**3.2.2 Virtual Campus Explorer CSCI (VCE)** 10](#_Toc528360971)

[**3.2.3 Virtual Parking Finder CSCI (VPF)** 11](#_Toc528360972)

[**3.2.4 Virtual Unity Engine CSCI (VUE)** 13](#_Toc528360973)

[**3.3 CSCI external interface requirements** 15](#_Toc528360974)

[**3.3.1 Interface identification and diagrams** 15](#_Toc528360975)

[**3.4 CSCI internal interface requirements** 15](#_Toc528360976)

[**3.5 CSCI internal data requirements** 15](#_Toc528360977)

[**3.6 Adaptation requirements** 15](#_Toc528360978)

[**3.7 Safety requirements** 16](#_Toc528360979)

[**3.8 Security and privacy requirements** 16](#_Toc528360980)

[**3.9 CSCI environment requirements** 16](#_Toc528360981)

[**3.10 Computer resource requirements** 16](#_Toc528360982)

[**3.10.1 Computer hardware requirements** 16](#_Toc528360983)

[**3.10.2 Computer hardware resource utilization requirements** 16](#_Toc528360984)

[**3.10.3 Computer software requirements** 16](#_Toc528360985)

[**3.10.4 Computer communications requirements** 17](#_Toc528360986)

[**3.11 Software quality factors** 17](#_Toc528360987)

[**3.12 Design and implementation constraints** 17](#_Toc528360988)

[**3.12.1 Standards Compliance** 17](#_Toc528360989)

[**3.13 Personnel-related requirements** 17](#_Toc528360990)

[**3.14 Training-related requirements** 17](#_Toc528360991)

[**3.15 Logistics-related requirements** 17](#_Toc528360992)

[**3.16 Other requirements** 18](#_Toc528360993)

[**3.17 Packaging requirements** 18](#_Toc528360994)

[**3.18 Precedence and criticality of requirements** 18](#_Toc528360995)

[4 Requirements Traceability 18](#_Toc528360996)

**0 Revision History**

**0.1 26 October 2018: CMSC447-05-FA2018-G03-SRS-01A**

Release A contains the basic specification for the UMBC Virtual Tour 2.0 System.

**1 Scope**

This specification establishes the functional, performance, and development requirements for version 1.0 of a software application enabling virtual tours of the UMBC campus.

**1.1 Identification**

Title: UMBC Virtual Tour 2.0

Abbreviation: VT2

Version Number: 1.0

**1.2 Definitions, Acronyms and Abbreviations**

API Application Programming Interface

CSCI Computer Software Configuration Item

GUI Graphical User Interface

HTML Hyper Text Markup Language

IRC Imaging Research Center

OSM Open Street Map

SIMD Single Instruction, Multiple Data

SRS Software Requirements Specification

SSE2 Streaming SIMD Extensions 2

TCP Transmission Control Protocol

UMBC University of Maryland, Baltimore County

Unity The Unity cross-platform game engine

VCE Virtual Campus Explorer CSCI

VPF Virtual Parking Finder CSCI

VTI Virtual Tour Interface CSCI

VT2 UMBC Virtual Tour 2.0

VUE Virtual Customized Unity Engine CSCI

WebGL The Web Graphics Library, a cross platform JavaScript API for rendering 2D and 3D graphics in a web browser

XML Extensible Markup Language

UDP User Datagram Protocol

**1.3 References**

The following standards apply:

IEEE Std 830-1998 IEEE Recommended Practice for Software Requirements Specifications

MIL-STD-498 Military Standard Software Development and Documentation

UMBC Style Guide https://styleguide.umbc.edu/

Unity User Manual

(2018.2) <https://docs.unity3d.com/Manual/index.html>

WebGL <https://docs.unity3d.com/Manual/webgl.html>

**1.4 Document Overview**

This document is organized as follows: Section 1 identifies the scope of this document and lists the definitions, abbreviations, acronyms, and references used therein. Section 2 provides an overview of the system and a brief description of its architecture. Section 3 lists all of the functional requirements for the system, including the four primary CSCIs, as well as the qualification methods for each CSCI. Finally, section 4 describes the traceability requirements for the system.

**2 Overall Description**

**2.1 System Purpose and Perspective**

A virtual campus tour is an important component of a university’s strategy for recruiting students. In an environment of intense competition for students of all types—domestic and international, in-state and out-of-state, and undergraduate and graduate—a strong virtual tour application can convince a prospective student to apply or visit the campus in person. Additionally, virtual campus tours can help current students and visitors navigate their way to their classes or special events.

UMBC currently has several websites that nominally offer virtual tours of the campus. The Undergraduate Admissions UMBC Virtual Tour (located at https://undergraduate.umbc.edu/visit/virtual-tour.php) provides 9 panoramic views of the campus, though it claims to offer 25 views. A virtual tour site for the graduate school (https://gradschool.umbc.edu/discover/vtour/) simply provides a link to the same site that hosts the panoramic campus views noted above. Additionally, a UMBC undergraduate student created a basic virtual tour mobile application for Android devices in 2014 titled, “Introducing UMBC Tours - A Virtual Campus Tour Experience for Android” (https://www.youtube.com/watch?v=zRI61jkUDT4). However, this implementation had extremely limited functionality and did not represent a significant improvement on the applications offered on the UMBC website.

The software described in this specification (the UMBC VT2 system) shall dramatically improve the currently available UMBC virtual tour applications by importing the UMBC campus map and building information into the Unity game engine. It shall allow users to select any location on a three-dimensional map of the campus and allow them to explore it freely. It shall provide browser-based access to this system through a web application. Moreover, the new system shall provide several other useful features, including the ability to highlight valid parking locations based on user status. The system will primarily benefit prospective students seeking to familiarize themselves with the campus environment and current students, faculty, and visitors trying to find their classes or event venues and seeking the best place to park.

**2.1 System Functions**

The system has four primary functions that correspond to the CSCIs specified in section 3:

1. **Virtual Campus Explorer (VCE):** This CSCI shall allow the user to freely explore the virtual UMBC campus map from a chosen starting point.
2. **Virtual Parking Finder (VPF):** This CSCI shall identify parking lots where the user is allowed to park based on the user’s status (faculty member, commuter student, residential student, visitor, etc.).
3. **Virtual Tour Interface (VTI):** This CSCI shall provide a browser-based interface for the VT2 system based on the WebGL framework. The interface shall provide the main menu for the system, allowing the user to select either the VCE function or the VPF function.
4. **Virtual Unity Engine (VUE):** This CSCI shall provide a customized version of the Unity Engine that includes accurate 3D renderings and textures of the UMBC campus buildings. It shall allow natural movement around the campus’ exterior spaces with motion and camera effects generally expected of first and third-person point of view video games.

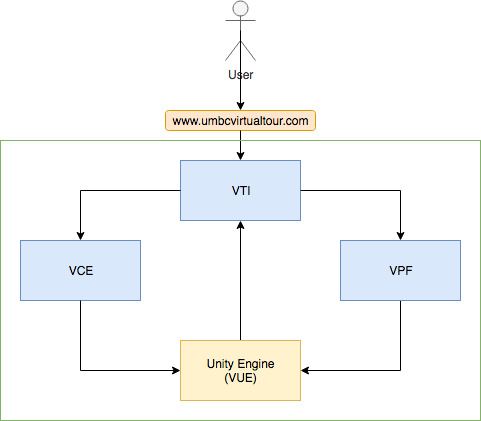


Figure 1 System Architecture

**2.2 User Characteristics**

Users of this system are assumed to possess basic familiarity with internet browsers and websites, but no other special knowledge or skills are required. All user interaction with the system will take place through browser-based menus, and the system will prompt the user to take action with clear and simple instructions when necessary.

**2.3 Constraints**

The system relies on UMBC’s IRC office to provide the object files necessary for creating three-dimensional renderings of the campus buildings. As of late October 2018, the IRC department did not have renderings and textures for some of the buildings. As a result, version 1.0 of the web application will only contain a subset of the 43 buildings that comprise the UMBC main campus.

Additionally, the WebGL framework that allows the Unity engine to be run within a web browser is not currently supported on mobile devices. Therefore, the user must access the system through a desktop computer or laptop. Also, a compatible browser as described in the Unity WebGL manual (https://docs.unity3d.com/Manual/webgl-browsercompatibility.html) must be used. Most modern browsers (Firefox, Chrome, Safari, Microsoft Edge) are supported. Future versions of the system will provide support for mobile computing using a different framework.

Finally, version 1.0 of the software will not provide the user with directions to or from parking lots or buildings. It is intended that the system will provide this capability in a future release.

**2.4 Assumptions and Dependencies**

It is assumed that the CSCIs identified in this version of the SRS are the basic CSCIs necessary to meet customer requirements. Once the software engineering team has successfully implemented these CSCIs, the team may proceed with implementation of “reach goals” such as direction-finding algorithms and mobile support upon agreement with the customer.

No other special assumptions or dependencies have been identified.

**3 Requirements**

**3.1 Required states and modes**

No states or modes are required.

**3.2 CSCI Functional Requirements**

This section describes the functional requirements for each of the four CSCIs: the Virtual Tour Interface CSCI (VTI), the Virtual Campus Explorer CSCI (VCE), the Virtual Parking Finder CSCI (VPF), and the Virtual Unity Engine CSCI (VUE).

**3.2.1 Virtual Tour Interface CSCI (VTI)**

3.2.1.1 The VTI shall be displayed through the website https://umbcvirtualtour.com.

This requirement shall be tested using demonstration to show that the VTI is displayed when the user goes to the specified website.

3.2.1.2 The VTI shall be built using HTML5 and JavaScript ECMAScript 2018.

This requirement shall be tested using inspection to verify that the code is written in the correct language.

3.2.1.3 The VTI shall use the WebGL JavaScript API to deploy the VUE.

This requirement shall be tested using inspection to confirm that the code uses the correct API.

3.2.1.4 The VTI main page (the website home page) shall include a welcome banner identifying it as the UMBC Virtual Tour 2.0 Website.

This requirement shall be tested using demonstration to verify that the VTI main page contains the banner.

3.2.1.5 The VTI shall include a link to the UMBC home page (www. umbc.edu) and display one of any of the UMBC logos according to the UMBC style guidelines.

This requirement shall be tested using demonstration to confirm that the VTI contains a link to the UMBC home page and to compare the logo on the VT2 website with the logos from the UMBC style guide web page.

3.2.1.6 The VTI main page shall display a menu to the user with the following options: “About this Website”, “Explore the Campus”, “Find Parking”, and “Help”.

This requirement shall be tested using demonstration to show that the VTI main page displays the menu with the correct options.

3.2.1.7 Clicking the “About this Website” option shall take the user to a page that provides a brief introduction to UMBC and identifies the purpose of the website as providing virtual tours of the university.

This requirement shall be tested using demonstration to show that once the user chooses “About this Website”, the VTI displays a page with a brief introduction to UMBC along with the other information listed in the requirement.

3.2.1.8 Clicking the “Explore the Campus” option shall direct the user to a new page on the website that launches the VCE.

This requirement shall be tested using demonstration to show that choosing “Explore the Campus” behaves as the requirement states.

3.2.1.9 Clicking the “Find Parking” option shall direct the user to a new page on the website that launches the VPF.

This requirement shall be tested using demonstration to show that choosing “Find Parking” behaves as the requirement states.

3.2.1.10 Clicking the “Help” option shall direct the user to a new page on the website that provides a tutorial on how to use the VCE and VPF. The length of the tutorial shall be no fewer than 150 words and no more than 300 words for each of the VCE and VPF CSCIs.

This requirement shall be tested using demonstration to show that clicking on “Help” will load a new page that provides a tutorial.

3.2.1.11 All VTI web pages other than the main page shall have a clickable button labeled “Exit” or “Home” that returns the user to the VTI main page.

This requirement shall be tested using demonstration to show that the designated button exists and functions as specified.

3.2.1.12 The website shall store no data related to the user (such as cookies) other than a user agent string necessary for determining browser compatibility.

This requirement shall be tested using inspection to confirm that the code does not allow for the storage of any data other than user agent strings.

3.2.1.13 The website shall be compatible with the following web browsers: Firefox 4+, Google Chrome 9+, Opera 12+, Safari 5.1+, Internet Explorer 11+, and Microsoft Edge build 10240+.

This requirement shall be tested using demonstration to show that when the webpage is opened in each web browser, it correctly loads and functions as per all functional requirements listed in this document.

3.2.1.14 If the user accesses the site with a browser other than one listed in requirement 3.2.1.13, the VTI shall display an error message stating that the user is using an incompatible browser and listing the acceptable browser types.

This requirement shall be tested by demonstration; namely, attempting to access the site with a browser other that one of those specified above, and confirming that the system displays the appropriate error message in response.

3.2.1.15 If the user attempts to access the website from a mobile device, the website shall display an error message stating that WebGL is not supported on mobile devices.

This requirement shall be tested by demonstration; namely, attempting to access the site with a mobile device, and confirming that the system displays the appropriate error message in response.

**3.2.2 Virtual Campus Explorer CSCI (VCE)**

3.2.2.1 The VCE shall load within 15 seconds and shall display a progress bar indicating the time remaining until loading is complete.

This requirement shall be tested using demonstration and measurement, repeatedly accessing the website and verifying that the VCE loads in a maximum of 15 seconds.

3.2.2.2. Upon launch, the VCE initial view shall be zoomed out such that the entire campus map is displayed in overhead view.

This requirement shall be tested using demonstration to show that the entire campus map is visible in the initial view.

3.2.2.3 The VCE shall provide a prompt telling the user to double-click a location on the campus to start exploring.

This requirement shall be tested using demonstration to show that the prompt is displayed.

3.2.2.4 After the user double-clicks a location on the map, the system shall zoom to a third-person (3P) street-level view of that part of the map.

This requirement shall be tested using demonstration to show that the program functions as the requirement states.

3.2.2.5 If the user clicks a non-walkable area of the campus (any area other than a sidewalk or other walkway, road, parking lot, or grass), the system shall zoom in to the closest walkable location; alternatively, the system shall display a message informing the user that an invalid location was chosen and prompt the user to click a walkable location.

This requirement shall be tested using demonstration; namely by repeatedly clicking on non-walkable areas of the map and confirming that the system behavior each time conforms to one of the two specified options.

3.2.2.6 The user’s avatar shall take the form of the UMBC mascot, the Chesapeake Bay Retriever True Grit.

This requirement shall be tested using demonstration to show that the avatar is in the form of UMBC’s mascot.

3.2.2.7 When in 3P view, the system shall provide a method of shifting to first-person (1P) view and the system shall provide a text box informing the user how to shift to this view.

This requirement shall be tested using demonstration to show that program functions as the requirement states.

3.2.2.8 The user shall be able to advance through the 3D rendering of the map by pressing the up, down, left, and right arrow keys on the keyboard or by double clicking a location in the distance.

This requirement shall be tested using demonstration to show that the user is able to move through the 3D rendering of the map in the correct direction as arrow key pressed or location double clicked.

3.2.2.9 When in 1P or 3P view, the system shall display an inset overhead view map showing the user’s current location.

This requirement shall be tested using demonstration to show that the overhead map is displayed.

**3.2.3 Virtual Parking Finder CSCI (VPF)**

3.2.3.1 Upon launch, the VPF shall display a menu prompting the user to identify as a student, faculty, or visitor.

This requirement shall be tested using demonstration to show that the prompt is displayed and has the correct options.

3.2.3.2 If the user selects the student option, the VPF shall prompt the user to identify the type of student parking permit possessed. The options for the permit shall be displayed in two parts: a letter in white font within a colored circle, and the name of the type of permit. The options shall be as displayed in the table below:

Table 1 Student Parking Permit Types

|  |  |
| --- | --- |
| **Letter and Color** | **Name** |
| ‘A’, red | Commuter Student |
| ‘B’, green | Walker Community Resident |
| ‘C’, yellow | Residential Student (Besides Walker) |
| ‘F’, orange | Freshman Resident Student |

This requirement shall be tested using demonstration to show that the correct options are displayed in the correct format as stated in the requirement.

3.2.3.3 If the user selects the faculty option, the VPF shall prompt the user to identify the type of faculty parking permit possessed. The options for the permit shall be displayed in two parts: a letter in white font within a colored circle, and the name of the type of permit. The options shall be as displayed in the table below:

|  |  |
| --- | --- |
| **Letter and Color** | **Name** |
| ‘D’, purple | Faculty/Staff |
| ‘E’, purple | Gated Faculty/Staff |

This requirement shall be tested using demonstration to show that the correct options are displayed in the correct format as stated in the requirement.

3.2.3.4 Both menus described in requirements 3.2.3.2 and 3.2.3.3 shall also include an option, “I don’t have a permit.” If the user selects this option, the VPF shall display a message telling the user that current faculty and students can request a permit through the UMBC parking services website and providing a hyperlink to the website. Additionally, the VPF shall display an option allowing the user to “See Paid Parking Options.”

This requirement shall be tested using demonstration to show that the message is displayed with the correct hyperlink and the “See Paid Parking Options” button is presented.

3.2.3.5 After successfully selecting a permit type, the VPF shall prompt the user to select a destination building. The VPF will present a drop-down menu with the 25 buildings identified in requirement 3.2.4.4 below. Additionally, the VPF shall allow the user to select a building by clicking it on the map.

This requirement shall be tested using demonstration to show that a drop-down menu of buildings is displayed and that the user can select a building by double-clicking.

3.2.3.6 Once the user selects a building, the VPF shall highlight all the parking lots on the map where the user can park based on the user’s parking permit. Additionally, the VPF shall identify the closest parking lot to the destination building where the user is allowed to park. This closest lot shall be highlighted in a different color from the other lots and shall be labeled “Recommended Parking Lot.”

This requirement shall be tested using demonstration to show that the correct parking lots are highlighted based on the permit selected in requirement 3.2.3.2 or 3.2.3.3 and that the parking lot highlighted a different color is the one closest to the destination building selected in requirement 3.2.3.5.

3.2.3.7 If the user selects the “Visitor” option, then the VPF shall present a menu with the options, “Special Event Parking” and “Paid Parking Options.”

This requirement shall be tested using demonstration to show that if the “Visitor” option is selected, the user is presented a menu with the two options listed in this requirement.

3.2.3.8 If the user selects “Special Event Parking,” the VPF shall highlight the stadium lot.

This requirement shall be tested using demonstration to show that the stadium lot is highlighted if the user selects “Special Event Parking”.

3.2.3.9 If the user selects “Paid Parking Options” in requirements 3.2.3.8 or 3.2.3.4, the VPF shall prompt the user to select a destination building. The VPF will present a drop-down menu with the 25 buildings identified in requirement 3.2.4.4 below, as well as a “No specific building” option. Additionally, the VPF shall allow the user to select a building by clicking it on the map.

This requirement shall be tested using demonstration to show that the user is given a drop-down menu to select a building as well as the “No specific building” option.

3.2.3.10 Once the user selects an option under requirement 3.2.3.9, the VPF shall then highlight all the paid parking lots on campus, and if the user selected a building, it shall highlight the closest paid parking option to that building in a different color and that lot shall be labeled “Recommended Parking Lot.”

This requirement shall be tested using demonstration to show that all of the paid parking lots on campus are highlighted, and that if a destination building was selected, the nearest parking lot is correctly highlighted in a different color from the other lots.

**3.2.4 Virtual Unity Engine CSCI (VUE)**

3.2.4.1 The VUE shall use Unity Personal version 2018.2.13.

This requirement shall be tested using inspection to show that the correct Unity Personal version is used.

3.2.4.2 The VUE shall use UMBC campus map data from OSM as the basis for its map.

This requirement shall be tested using inspection to show that the map data is from OSM.

3.2.4.3. The VUE shall enhance the basic OSM map by importing models and textures obtained from the UMBC IRC office for 25 campus buildings.

This requirement shall be tested using inspection to show that the OSM map is enhanced by imported models and textures.

3.2.4.4. The 25 campus buildings noted in requirement 3.2.4.3 above shall be the following:

Table 2 UMBC Building Names and Abbreviations

|  |  |
| --- | --- |
| **Name** | **Abbreviation** |
| Administration Building | Admin |
| Biological Sciences Building | Biology |
| Campus Police and Central Plant | Police |
| Chesapeake Hall | Chesapeake |
| The Commons | The Commons |
| Engineering Building | Engineering |
| Erickson Hall | Erickson |
| Event Center | Event Center |
| Fine Arts Building | Fine Arts |
| Harbor Hall | Harbor |
| Information Technology/Engineer Building | ITE |
| Mathematics & Psychology Building | Math & Psych |
| Meyerhoff Chemistry Building | Meyerhoff |
| Performing Arts and Humanities Building | PAHB |
| Patapsco Hall | Patapsco |
| Physics Building | Physics |
| Potomac Hall | Potomac |
| Public Policy Building | PUP |
| Retriever Activities Center (RAC) | RAC |
| Sherman Hall | Sherman |
| Sondheim Hall | Sondheim |
| Student Development & Success Center (SDSC) | SDSC |
| Susquehanna Hall | Susquehanna |
| True Grit’s | True Grit’s |
| University Center | UC |

This requirement shall be tested by obtaining confirmation from the UMBC IRC Department that the correct models were obtained, and through visual comparison of the imported models and textures as applied to the buildings in the Unity engine to the appearance of the actual campus buildings.

3.2.4.5 When in overhead view, the map shall display the abbreviated building name from Table 1 above; mousing over the building shall display the full name of the building.

This requirement shall be tested using demonstration, namely by confirming that the map in its initial view displays the abbreviated names for every building when the mouse is not hovering over any building, and by hovering over every building in turn and confirming that the full name is displayed.

3.2.4.6 When in 1P or 3P view, the map shall display the full name of the building when approached by the user from any direction.

This requirement shall be tested using demonstration. Each building on the campus map shall be approached from multiple directions to confirm that the abbreviated name is displayed until the user nears the building, at which point the full name shall be displayed.

3.2.4.7 The VUE shall accurately render the relative heights of the campus buildings.

This requirement shall be tested using inspection to show that the code creates the correct sizes of the buildings. Additionally, it shall be tested by observing the map in VCE view and visually confirming that campus buildings correctly connect on multiple levels and that buildings are clearly distinguishable as being farther up or down the hills that form the UMBC campus terrain.

3.2.4.8 While in 1P or 3P view, the VUE camera view shall remain behind the user at all times and shall not pass through the exterior of buildings.

This requirement shall be tested by showing that the camera maintains the correct position when the user approaches a building or other boundary point on the map.

3.2.4.9 The VUE shall only allow the user to move on “walkable areas” of the map defined as sidewalks, campus walkways, roads, parking lots, and grass. If the user tries to move into any other area, the user’s avatar shall continue to display walking motion, but no forward progress shall occur.

This requirement shall be tested using through demonstration, by observing the system’s behavior when the user attempts to move into a non-walkable area and confirming that the motion occurs as described.

3.2.4.10 In addition to the campus buildings, the VUE custom map shall also display grassy areas, roads, pavement, sidewalks, and paths that match the actual UMBC campus.

This requirement shall be tested by visually comparing the 3D Unity map of the campus with the actual campus appearance.

3.2.4.11 The VUE shall display the lot numbers of all parking lots, as well as an icon with a letter within a colored circle as noted in requirements 3.2.3.2 and 3.2.3.3 that indicates which permit types allow parking in that lot.

This requirement shall be tested by examining each lot in the VCE map, comparing it with the corresponding lot on the parking map on the UMBC parking services website, and confirming that it has the correct lot number and icon.

**3.3 CSCI external interface requirements**

**3.3.1 Interface identification and diagrams**

All interaction with the user shall occur through the VTI CSCI. As noted in section 3.2.1, the VTI shall be a web-based GUI.

**3.4 CSCI internal interface requirements**

As noted in requirement 3.2.1.3, the VTI shall use HTML5 and the WebGL JavaScript API.

**3.5 CSCI internal data requirements**

All internal data requirements shall be left to the design.

**3.6 Adaptation requirements**

No adaptation requirements have been identified.

**3.7 Safety requirements**

No specific safety requirements have been identified.

**3.8 Security and privacy requirements**

There are no specific security or privacy requirements, as the system will not record any logon information or other personal data related to the user, nor does the system house any confidential data.

**3.9 CSCI environment requirements**

3.9.1 The computer acting as the server and host for the website http://umbcvirtualtour.com shall meet the requirements specified on the “System Requirements for Unity 2018.2” webpage located at https://unity3d.com/unity/system-requirements. Specifically. the computer shall use one of the following operating systems: OS: Windows 7 SP1+, macOS 10.11+, Ubuntu 12.04+, or SteamOS+.

3.9.2 The website shall be accessed through one of the following web browsers: Firefox 4+, Google Chrome 9+, Opera 12+, Safari 5.1+, Internet Explorer 11+, or Microsoft Edge build 10240+.

**3.10 Computer resource requirements**

**3.10.1 Computer hardware requirements**

3.10.1.1 The computer acting as the server and host for the website http://umbcvirtualtour.com shall have a graphics card with DX10 (shader model 4.0) capabilities.

3.10.1.2 The computer acting as the server and host for the website http://umbcvirtualtour.com shall have a CPU with SSE2 instruction set support.

3.10.1.3 The computer acting as the server and host for the website http://umbcvirtualtour.com shall have a minimum 2.4 GHz processor with 8 GB of memory.

**3.10.2 Computer hardware resource utilization requirements**

No requirements for computer hardware resource utilization have been identified.

**3.10.3 Computer software requirements**

No computer software requirements have been identified other than those already stated above.

**3.10.4 Computer communications requirements**

The system shall communicate over the internet through standard web protocols, namely TCP and UDP.

**3.11 Software quality factors**

3.11.1 Maintainability – All code written for the VT2 application shall be properly documented to ensure maintainability. Specifically, all code shall include comments in accordance with the standards set forth in the HTML5 style guide; all files shall include headers with the file name, purpose, author, and last modified date; and all functions shall include pre and post conditions.

3.11.2 Reliability – The system shall not crash except due to operating system faults or other external errors.

3.11.3 Availability – The system shall always be available when accessed except in the case of external faults such as hardware failures.

3.11.4 Portability – The VT2 application shall be portable to any hardware server that meets the requirements outline in section 3.10. Likewise, the VT2 application shall be accessible by any user whose browser meets the requirements in section 3.9.

**3.12 Design and implementation constraints**

**3.12.1 Standards Compliance**

The VT2 system shall comply with the standards described in section 1.3. Additionally, The VT2 system shall comply with the UMBC Style Guide (https://styleguide.umbc.edu/) with regard to the use of the UMBC name and logo.

**3.13 Personnel-related requirements**

Version 1.0 of the software shall support 1 user at a time. The VTI CSCI shall provide a help menu and prompts for each screen.

**3.14 Training-related requirements**

No special training is required to use the system. The VT2 website shall provide prompts and a help section to assist the user in navigating the site.

**3.15 Logistics-related requirements**

While the customer shall have responsibility for maintaining the server on which the software resides after testing and acceptance, the development team shall provide technical support for problems encountered with the software via the Jira ticketing system, email, and live chat. The technical support team shall respond to all requests within one business day.

**3.16 Other requirements**

Use of the Unity engine within the VT2 system shall comply with the requirements of Unity Technologies’ licensing policy. Specifically, the VT2 system shall initially use a free unity license for students and in-class instruction as described on the Unity website at https://store.unity.com/education. Once testing and acceptance is complete and UMBC incorporates the system into its official website, the VT2 system shall convert to an appropriate paid license.

**3.17 Packaging requirements**

The system shall be packaged as a single compressed file containing binaries, executables, and source code for the web application and all files necessary to install the customized Unity engine. Additionally, the file shall contain a README with information about how to install and run the software on a web server. Initially, the development team shall have responsibility for maintaining the server on which the software runs. After completion of acceptance and verification, the development team shall transfer responsibility for the server to the customer.

**3.18 Precedence and criticality of requirements**

The development team and customer have determined that the most critical requirements are implementing the VTI, VCE, and VUE CSCIs. Therefore, development of these CSCIs shall take precedence over the VPF CSCI and any additional CSCIs proposed in the future.

**4 Requirements Traceability**

This SRS serves as the origin of the requirements for the VT2 system. Section 3 describes how each CSCI functional requirement can be directly traced to the overall system goals described in section 2. Thus, there are no separate system requirements necessitating traceability other than the functional requirements presented in this document. Likewise, no other documents, such as statements of work or interface specifications, exist that require traceability.