|  |
| --- |
| American Video Gaming Company |
| Software Project |
| C188 Performance Assessment |

|  |
| --- |
| Timothy Javins  10-19-2021  Version 1 |

Contents

[A. **Introduction** 3](#_Toc86688837)

[A.1. Purpose Statement 3](#_Toc86688838)

[A.2. Overview of the Problem 3](#_Toc86688839)

[A.3. Goals and Objectives 3](#_Toc86688840)

[A.4. Prerequisites 3](#_Toc86688841)

[A.5. Scope 3](#_Toc86688842)

[A.6. Environment 3](#_Toc86688843)

[B. **Requirements** 4](#_Toc86688844)

[B.1. Business Requirements 4](#_Toc86688845)

[B.2. User Requirements 4](#_Toc86688846)

[B.3. Functional Requirements 4](#_Toc86688847)

[B.4. Nonfunctional Requirements 4](#_Toc86688848)

[C. **Software Development Methodology** 4](#_Toc86688849)

[C.1. Advantages of the Waterfall Method 4](#_Toc86688850)

[C.2. Disadvantages of the Waterfall Method 5](#_Toc86688851)

[C.3. Advantages of Adaptive 5](#_Toc86688852)

[C.4. Disadvantages of Adaptive 5](#_Toc86688853)

[C.5. Best Suited Method 5](#_Toc86688854)

[D. **Design** 6](#_Toc86688855)

[D.1. Flowchart 6](#_Toc86688856)

[D.2. UML Diagram – Entity Types 7](#_Toc86688857)

[E. **Testing** 8](#_Toc86688858)

[E.1. Testing Type 8](#_Toc86688859)

[E.1.1. Creating Entities 8](#_Toc86688860)

[E.1.2. Third-Party Integration 9](#_Toc86688861)

[E.1.3. Geo-blocking 10](#_Toc86688862)

# Introduction

Provide a brief introduction to the proposed system. This section should be no longer than one paragraph.

# Purpose Statement

Provide a brief overview of the purpose of this document.

# Overview of the Problem

Provide a brief overview of the problem that the proposed solution will solve.

# Goals and Objectives

Provide the goals and objectives for the project and solution.

# Prerequisites

Outline any aspects that need to be in place prior to the design, development, and implementation of the project proposed in this document. Be sure to be clear and concise for all listed prerequisites. Also, clearly outline why each prerequisite is needed.

*Note: If no prerequisites are needed, include a paragraph justifying why there are no prerequisites.*

|  |  |  |  |
| --- | --- | --- | --- |
| Number | Prerequisite | Description | Completion Date |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Scope

This is a software-only project and does not include hardware. The system will be deployable to the cloud as well as on-premises structures.

Provide a brief overview of what the proposed solution will cover and what the proposed solution will not cover. It is important to set clear boundaries for the project.

# Environment

The system’s User Interface (UI) will be served from a web server and utilize AVGC’s authentication system. The UI will facilitate passing data between the User and the CRM database. The system will support the following browsers running on Windows 10, macOS 10 – 12, iOS 13 – 15, and Android 8.1 – 12:

* Browsers based on Chromium 95
* Firefox 93
* Safari 15

# Requirements

This proposal will address the following five requirements:

* Integrating with other systems
* Extending function and diversity of support via integration
* Keeping data in the United States (US)
* Ease of use
* Data types

Additional requirements may be discussed at project milestones.

# Business Requirements

The system will allow users to create data to transmit via secure protocols to a database physically located in the United States. Access to the stored data from outside the United States will be denied.

# User Requirements

The user interface (UI) will employ responsive design principles and common design conventions to allow users to intuitively create data with virtually any device and no training.

# Functional Requirements

The system UI will be web-based to allow for general, platform-agnostic compatibility. The system will employ a well-established, well-known technology stack for the storage, maintenance, service, and manipulation of data. This will serve as a central point of integration for other systems, such as advanced forecasting tools and auto-scaling services.

# Nonfunctional Requirements

The system will enumerate the following data types and incorporate them into the objects it stores in the database:

* Stakeholder
* Individual
* Business
* Contact

By employing well-established and well-known technologies, the system and all data stored in the database will be supportable by third parties as needed and as authorized. Robust system documentation will be created as a part of the development process and included as a deliverable.

# Software Development Methodology

The following is a discussion of the traditional waterfall method versus the agile development method.

# Advantages of the Waterfall Method

The waterfall method has the following strengths:

* A clear roadmap with deliverables defined from the beginning
* The process is intuitive
* Deviations are easy to identify
* Scope creep is easy to manage
* Stakeholder buy-in occurs at the beginning
* Good for short, well-defined projects

# Disadvantages of the Waterfall Method

The waterfall mothed has the following weaknesses:

* Flaws in the requirements or design can follow all the way to deployment
* Poor agility when requirements changes occur during development
* Requirements must be precisely discerned and known in advance
* All phases of development must be done in sequential order
* Resources for subsequent phases are idle until the preceding phase completes
* Adverse risks become more likely as project length increases

# Advantages of Adaptive

Adaptive has the following strengths:

* Less rigid structure allows for better response to change
* Users can begin benefitting from the software earlier in the development lifecycle
* Users can provide feedback during development
* Deeper user engagement during development results in greater user buy-in
* Higher levels of user satisfaction

# Disadvantages of Adaptive

Adaptive has the following weaknesses:

* Adapting to changes can mean higher overall development costs

# Best Suited Method

Waterfall is known as a **predictive** development model: working with known requirements, the design team *predicts* what will work and the software engineers build it. This often works well with short timeframes where requirements are unlikely to change during development. However, AVGC is in a growth phase where the future is technologically unknown and includes unknown third parties and unknown APIs. As portrayed in the requirements document, **the technological needs of this project are unpredictable**.

As an **adaptive** development model, **the agile method would be best suited to AVGC’s unknown needs.** As the company grows and system requirements change, the agile method will not rely so heavily on assumptions about the future and will allow the project to change with the business. A usable system will be delivered at every milestone rather than only at the end of the development process.

# Design

The system’s graphical user interface (GUI) will direct non-geo-blocked users to authenticate into the system via AVGC’s authentication system in accordance with the Zero Trust model. Authenticated users will be able to create new stakeholders and view information on existing stakeholders, such as notes on past/upcoming communications, associations with other stakeholders (including contacts) and schedule information. In order to integrate with other systems and extend functionality, such as reporting and forecasting, the system will support application programming interface (API) calls, including representational state transfer (REST).

*If the proposed solution is hosted, the proposal must include a clear demonstration of how connectivity outages, service level agreements (SLAs), upgrades, custom development, ability to refuse upgrades, support, and maintenance will be handled. An environment where enhancements or changes can be tested prior to deployment of production must also be provided.*

# Flowchart

The following flowchart depicts the flow of information and data across the systems.

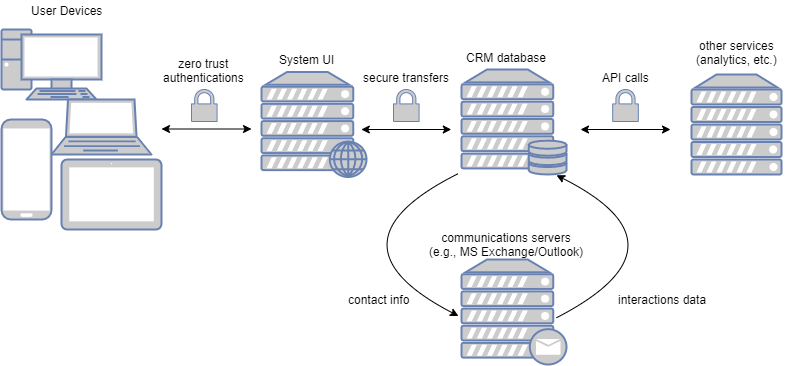


Figure 1: System in Production Environment

# UML Diagram – Entity Types

The following UML diagram details the objects derived from the required data types.

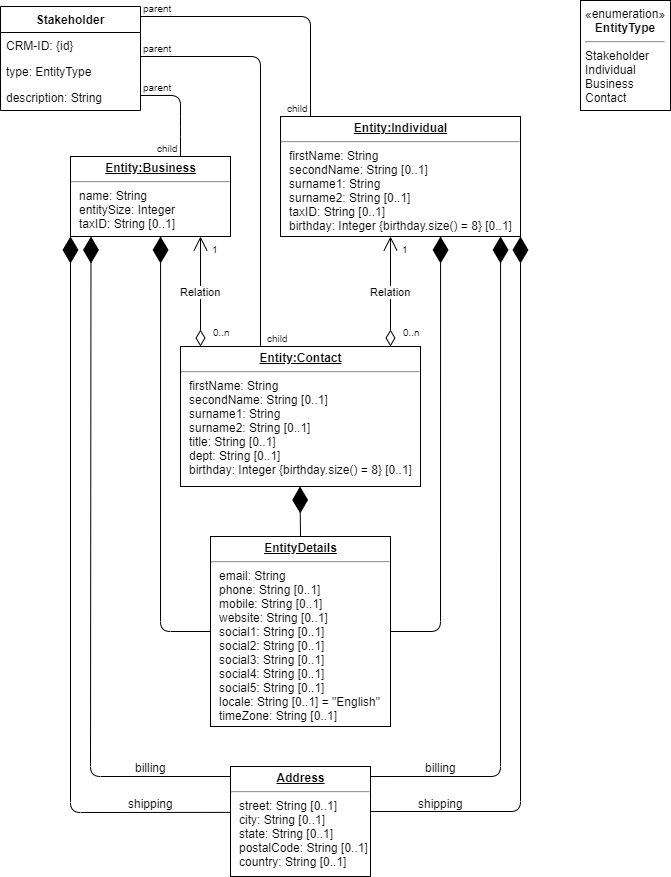


Figure 2: Entity Objects

# Testing

The testing process will verify functionality for creating new database entries, integrating with third-party software solutions, and blocking data from travel across US borders.

# Testing Type

In addition to unit testing, the software will be subjected to the following feature tests prior to alpha testing.

# Creating Entities

|  |
| --- |
| The software must create new entity objects based on the following data types:   * Stakeholder * Contact * Individual * Business   Upon creation, these objects must be stored in a database with unique identifiers. |
| Preconditions:  Prior to testing, the software must be considered a minimum viable product, deployed in a test environment, and running on two devices that can connect to the appropriate test database via networking. A test database must be running and available via networking. |
| Steps:   1. From the main GUI, choose to create a new stakeholder. 2. Choose the stakeholder entity type. 3. Try to submit a new stakeholder without completing all required fields. 4. Try to submit a new stakeholder with invalid data in required fields (see UML diagram). 5. Try to submit a new stakeholder with invalid data in optional fields (see UML diagram). 6. Try to submit a new stakeholder with required and some optional fields completed. 7. Repeat the above steps for each entity type (see UML diagram). 8. On the second test device, go to the main GUI and view each of the new stakeholders created in the previous steps. |
| Expected results:   1. The main GUI should present an option for creating a new stakeholder. 2. The user should be able to choose between the four entity types. 3. Once an entity type is selected, the software should present fields for the user to enter data to match the selected entity type (see UML diagram). 4. If any required field is left blank, the submission should fail with a prompt to complete the field. 5. If any field contains invalid data, the submission should fail with a prompt to correct any errors. 6. The submission should succeed with a confirmation message if there are no invalid data entries and the required fields are complete. 7. Each of the four stakeholder entity types should successfully submit to the database. 8. The second test device should present all information submitted for each stakeholder that was created via the first test device, each with a unique identifier called “Stakeholder ID”. |
| Pass/Fail: PASS  The test included, for each entity type, 30 unique cases and two duplicates totaling 128 cases. Half of the unique cases and one duplicate for each entity type were created using test device A. The other half were created using test device B. At the end of the test, both test devices presented the same list of 128 stakeholders. Each stakeholder was identified with a unique Stakeholder ID. Each entry on the list was populated with the correct information for the given stakeholder. |

# Third-Party Integration

|  |
| --- |
| The software should integrate with third party software via shared database connectivity and API calls. |
| Preconditions:  The software must pass test “E1.1. CREATING ENTITIES” in order populate the test database. The test database must be available via networking. The tester must have access to an instance of Tableau Server as well as cURL or Postman installed on the test machine. |
| Steps:   1. Make a REST API request via cURL/Postman to sign into the test server. (GET) 2. Create a Tableau test project. (POST) 3. Query the project. (GET) |
| Expected results:  The server should return XML content populated with data from the “E1.1. CREATING ENTITIES” test. |
| Pass/Fail: PASS  The tester used Postman to access Tableau Server and create a test project. The query on the test project successfully returned XML content generated from all 128 entities from “E1.1. CREATING ENTITIES”. |

# Geo-blocking

|  |
| --- |
| The software must store its data in the US and reject access to the data from outside the US. |
| Preconditions:  The database must be physically located in the US. The software must pass tests “E1.1. CREATING ENTITIES” and “E.1.2. THIRD-PARTY INTEGRATION”. The tester must have access to a Virtual Private Networking (VPN) service that supports non-US-based regions. |
| Steps:   1. Connect to a global VPN service on a test device. 2. Change the VPN region to outside the US. 3. Attempt to login to the software. 4. Attempt to login to the database via REST API (see “E.1.2. THIRD-PARTY INTEGRATION”). 5. Repeat steps 2-4 with a different VPN region. |
| Expected results:  The software should deny any login request originating from outside the US. The database should reject any REST requests originating from outside the US. |
| Pass/Fail: FAIL  The software passed test cases for both the Netherlands and Canada. The tester attempted to connect to the software while routing the network traffic through the Netherlands in the first case and Canada in the second. In both cases, the software denied access.  The database passed the Netherlands test case but not the Canada test case. The tester’s attempt to connect to the database via REST API was rejected while connected through the Netherlands. However, the test failed at step 4 when the tester was able to login to the database via Canada. Server logs show access to the test database was granted with testing credentials and an IPv6 address based in Canada. |