American Video Gaming Company

Software Project

C188 Performance Assessment

Timothy Javins

10-19-2021

Version 1

WESTERN GOVERNORS UNIVERSITY

CONTENTS

A. Introduction 3
A.1. Purpose Statement 3
A.2. Overview of the Problem 3
A.3. Goals and Objectives 3
A.4. Prerequisites 3
A.5. Scope 3
A.6. Environment 4
B. Requirements 4
B.1. Business Requirements 4
B.2. User Requirements 5
B.3. Functional Requirements 5
B.4. Nonfunctional Requirements 5
C. Software Development Methodology 5
C.1. Advantages of the Waterfall Method 5
C.2. Disadvantages of the Waterfall Method 5
C.3. Advantages of Agile 6
C.4. Disadvantages of Agile 6
C.5. Best Suited Method 6
D. Design 6
D.1. Flowchart 7
D.2. UML Diagram – Entity Types 8
E. Testing 9
E.1. Functional Tests 9
E.1.1. Creating Entities 9
E.1.2. Third-Party Integration 10
E.1.3. Geo-blocking 11

A. INTRODUCTION

This is a solution for the customer relationship management (CRM) needs that American Video Game Company (AVGC) faces as it continues to grow. The proposed system will enable AVGC to create new new stakeholders, link various types of stakeholders, and view summaries of interactions with stakeholders even when those interactions occur in other integrated systems. The system will generate data and provide a means to link data to stakeholders for various purposes, such as scheduling, analytics, and forecasting.

A.1. PURPOSE STATEMENT

The purpose of this document is to establish the scope of the CRM system for AVGC's consideration.

A.2. OVERVIEW OF THE PROBLEM

AVGC's customer base is growing quickly, with sales increasing 42% over the past two years. AVGC's current approach to customer management is not centralized, not standardized, and includes manual and automated processes. With 2,000 users and counting, the company has outgrown this disconnected approach and needs a cohesive, streamlined system.

A.3. GOALS AND OBJECTIVES

These are the goals and objectives of the CRM solution:

- Identify stakeholders
- Track stakeholder contacts
- Enhance privacy and security
- Link stakeholders and contact events
- Scale with growth
- Unify other CRM system components (email, analytics, purchases, etc.)

A.4. PREREQUISITES

In order to integrate with pre-existing and future system components, the designers and developers have the following prerequisites:

Number	Prerequisite	Description	Completion Date
1	Database info	Provide details on related databases	11/30/2021
2	Server info	Provide details on related servers	11/30/2021
3	3rd Parties	Identify third-party service components, such as hosting services and analytics service providers	11/30/2021

A.5. SCOPE

This is a software-only project. The following items are in scope:

- Graphical User Interface (GUI)
- Enumeration of data types (see B.4.)

- Database object creation and editing
- A means of integration with other software applications
- Reading and presenting information from a central database
- Testing and deployment to cloud-based and on-premises AVGC servers

While the system will be deployable to the cloud as well as on-premises structures, the project does not include hardware. The following items are out of scope:

- Installation of physical hardware
- Telecommunications provisions
- Developing or setting up authentication services
- Advanced analytics
- Forecasting
- User schedule management
- Hosting
- AVGC Sales processing

A.6. ENVIRONMENT

The system's User Interface (UI) will be served from a web server and utilize AVGC's authentication system. The system will facilitate passing data created by the User on their device to the CRM database. For resiliency and availability, the system should be deployed to multiple geographic regions. Cloud providers—such as Amazon Web Services (AWS), Microsoft Azure, Google Cloud Platform, and others—offer the best scalability across several regions of the United States (US). The design team is available for consultation.

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

B. REQUIREMENTS

This proposal will address the following five requirements:

- Keeping data in the United States (US) (B.1.)
- Ease of use (B.2.)
- Integrating with other systems (B.3.)
- Data types (B.4.)
- Extending function and diversity of support via integration (B.4.)

Additional requirements may be discussed at project milestones.

B.1. BUSINESS REQUIREMENTS

The system will allow authenticated users to create data to transmit via secure protocols to a database physically located in the United States. The user's location will be determined by network data (such as

network address) or device data where possible. Access to the stored data from outside the US will be denied.

B.2. USER REQUIREMENTS

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

B.3. 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. Deployment to a private or hybrid cloud is recommended.

B.4. NONFUNCTIONAL REQUIREMENTS

The system will enumerate the following data types and incorporate them into the objects it stores in the database (Figure 2):

- 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.

C. SOFTWARE DEVELOPMENT METHODOLOGY

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

C.1. 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

C.2. 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

C.3. ADVANTAGES OF AGILE

The agile approach 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

C.4. DISADVANTAGES OF AGILE

The agile approach has the following weaknesses:

- Adapting to changes can mean higher overall development costs
- Frequent user feedback means diverting talent from other also-important tasks
- If the requirements truly are known precisely up front, this is an inefficient way to develop them

C.5. 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. The requirements document mentions an undefined scope of existing tools and processes without defining any of them. As portrayed in the requirements document, **the technological needs of this project are unpredictable**.

As an **adaptive** development framework, **the agile approach 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.

D. 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 (Figure 2). Depending on the user's role, they will access information on existing stakeholders, 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).

D.1. FLOWCHART

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

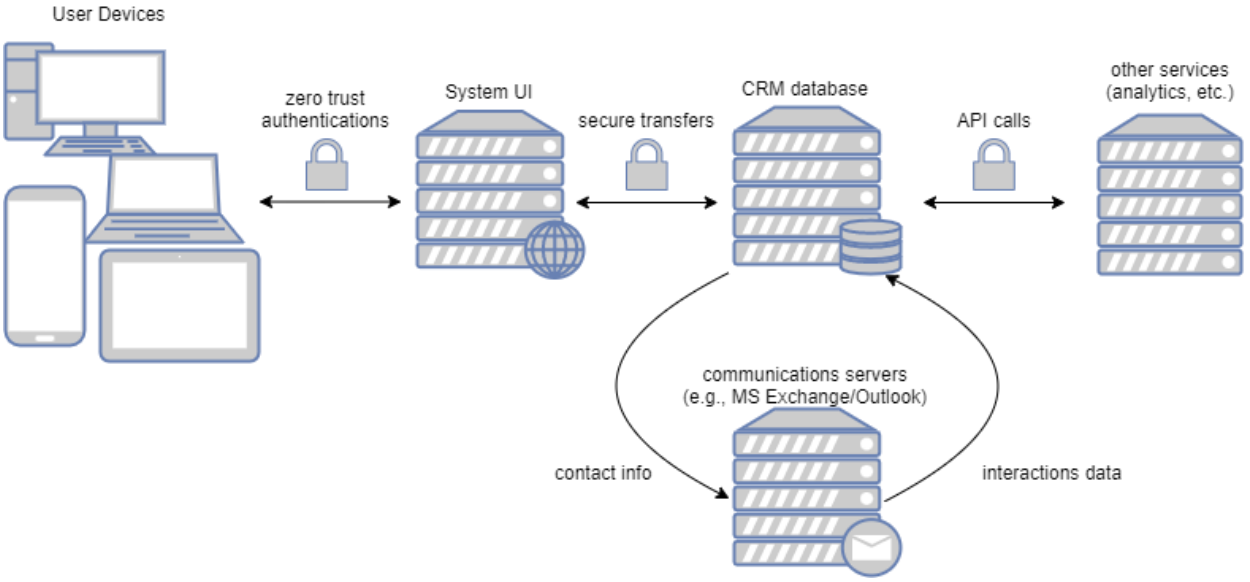

Figure 1: System in Production Environment

D.2. UML DIAGRAM – ENTITY TYPES

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

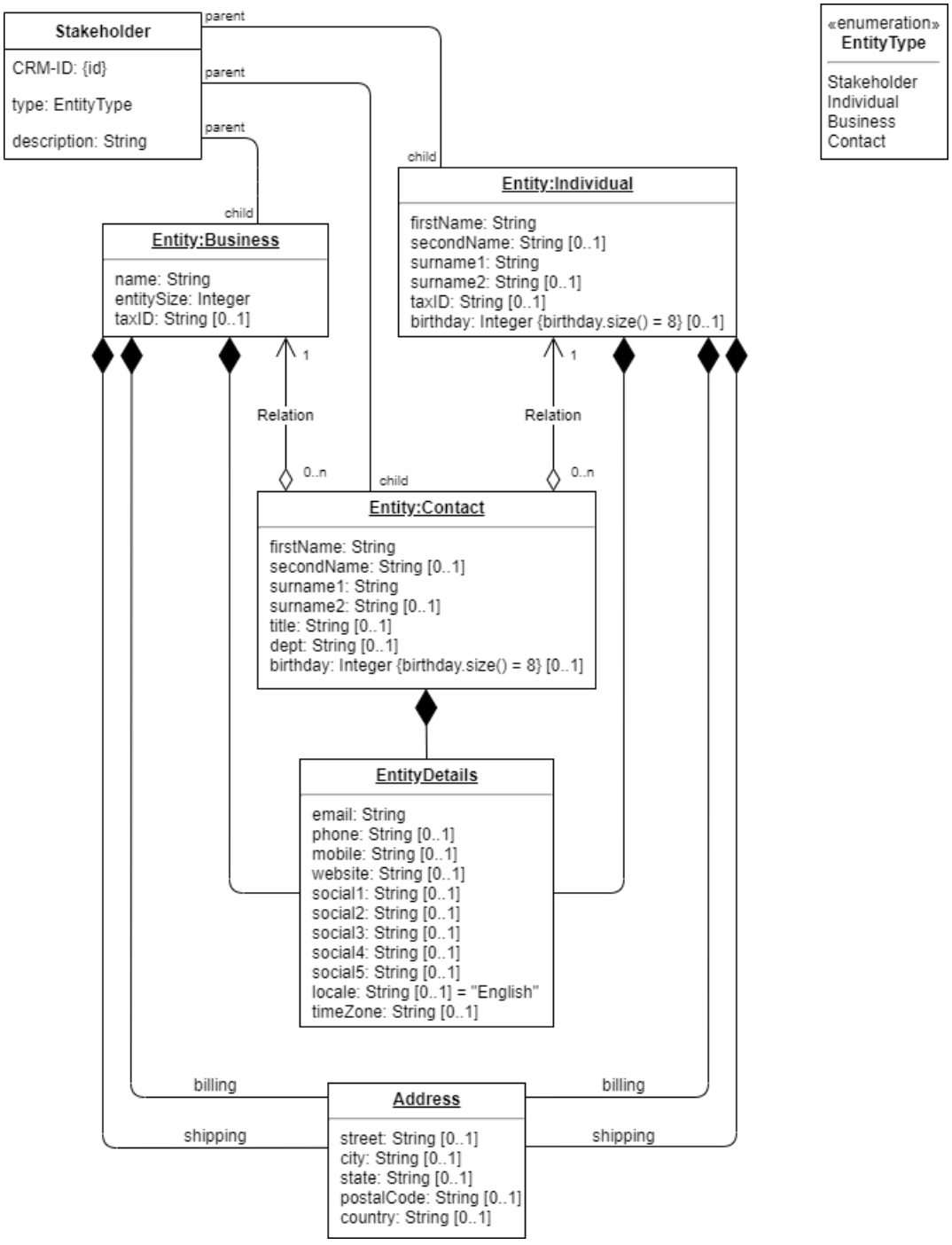

Figure 2: Entity Objects

E. 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.

E.1. FUNCTIONAL TESTS

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

E.1.1. CREATING ENTITIES

Requirement to be tested: 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.

E.1.2. THIRD-PARTY INTEGRATION

Requirement to be tested: 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”.

E.1.3. GEO-BLOCKING

Requirement to be tested: 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.