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

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

Contents

[A. Introduction 3](#_Toc7598012)

[A.1. Purpose Statement 3](#_Toc7598013)

[A.2. Overview of the Problem 3](#_Toc7598014)

[A.3. Goals and Objectives 3](#_Toc7598015)

[A.4. Prerequisites 3](#_Toc7598016)

[A.5. Scope 3](#_Toc7598017)

[A.6. Environment 3](#_Toc7598018)

[B. Requirements 4](#_Toc7598019)

[B.1. Business Requirements 4](#_Toc7598020)

[B.2. User Requirements 4](#_Toc7598021)

[B.3. Functional Requirements 4](#_Toc7598022)

[B.4. NonFunctional Requirements 4](#_Toc7598023)

[C. Software Development Methodology 5](#_Toc7598024)

[C.1. Advantages of the waterfall method 5](#_Toc7598025)

[C.2. Disadvantages of the waterfall method 5](#_Toc7598026)

[C.3. Advantages of {a different method} 5](#_Toc7598025)

[C.4. Disadvantages of {a different method} 5](#_Toc7598025)

[C.5. Best suited 5](#_Toc7598027)

[D. Design 6](#_Toc7598028)

[D.1. Storyboard or Flowchart (Change title to fit needs) 6](#_Toc7598029)

[D.2. UML Diagram (Change title to fit needs) 7](#_Toc7598030)

[D.3. GUI (Change title to fit needs) 8](#_Toc7598031)

[E. Testing 9](#_Toc7598032)

[E.1. Testing Type (change name to fit your needs) 9](#_Toc7598033)

[E.1.1. Test Name 1 9](#_Toc7598034)

[E.1.2. Test Name 2 9](#_Toc7598035)

[E.1.3. Test Name 3 10](#_Toc7598036)

[F. Sources 12](#_Toc7598037)

# Introduction

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

# A.1. Purpose Statement

Provide a brief overview of the purpose of this document.

# A.2. Overview of THE PROBLEM

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

# A.3. Goals and Objectives

Provide the goals and objectives for the project and solution.

# A.4. 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 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# A.5. Scope

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.

# A.6. Environment

Describe the IT and hardware environments that the solution will be deployed in.

# Requirements

This proposal will address the following five requirements:

* Consolidating contact and business information (functional)
* Integrating with other systems (via dB, functional)
* Extending function and diversity of support via integration (third-parties, non-functional)
* Keeping data in the United States (business)
* Ease of use (user)
* Data types (non-functional)

# Business Requirements

Provide a brief introduction to the business requirements for the proposed system.

# User Requirements

Provide a brief introduction to the user requirements for the proposed system.

# Functional Requirements

Provide a brief introduction to the functional requirements for the proposed system.

•

The system must be compatible with the below list of operating systems and browsers.

• latest Chrome and Chromium

• latest Firefox

• I.E 9 and above

• Safari 6.0

• mobile & tablet

• iOS7 Safari

• iOS7 Third Party Browsers (Chrome and Firefox)

• Android 4.0 Chrome

# NonFunctional Requirements

Provide a brief introduction to the nonfunctional requirements for the proposed system.

# SOFTWARE DEVELOPMENT METHODOLOGY

The company has selected the waterfall software development methodology for this project. Examine the waterfall methodology and compare it to other software development methodologies (e.g., Agile). Include a brief introduction to the development process as well.

*Note: All subsections are required. Refer to the requirements section and rubric section of the assessment for additional information.*

# Advantages of the waterfall method

Describe the advantages of the waterfall methodology and how they will benefit this project.

# disAdvantages of the waterfall method

Describe the disadvantages of the waterfall methodology and how they may hinder this project.

# Advantages of {A DIFFERENT METHOD}

Describe the advantages of a different methodology and how they will benefit this project.

# disAdvantages of {A DIFFERENT method}

Describe the disadvantages of a different methodology and how they may hinder this project.

# best SUITED

Describe why the waterfall methodology is the best software development methodology for this project.

**OR**

Provide the details of a different development process and outline why you would have selected it and how it would have been better suited for this project.

# Design

graphical user interface (GUI)

application programming interface (API) calls

documentation (JavaDocs)

Provide a brief overview of the proposed design.

*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.*

# Storyboard or Flowchart (Change title to fit needs)

Provide a storyboard or flowchart of the application.

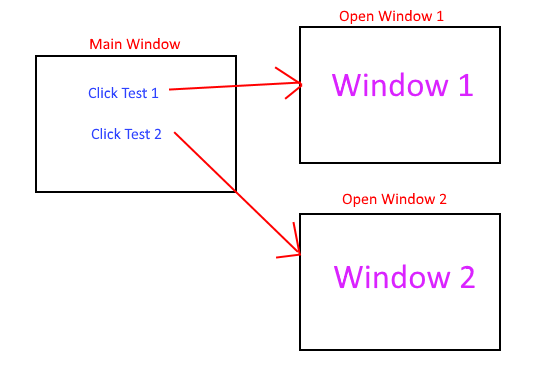
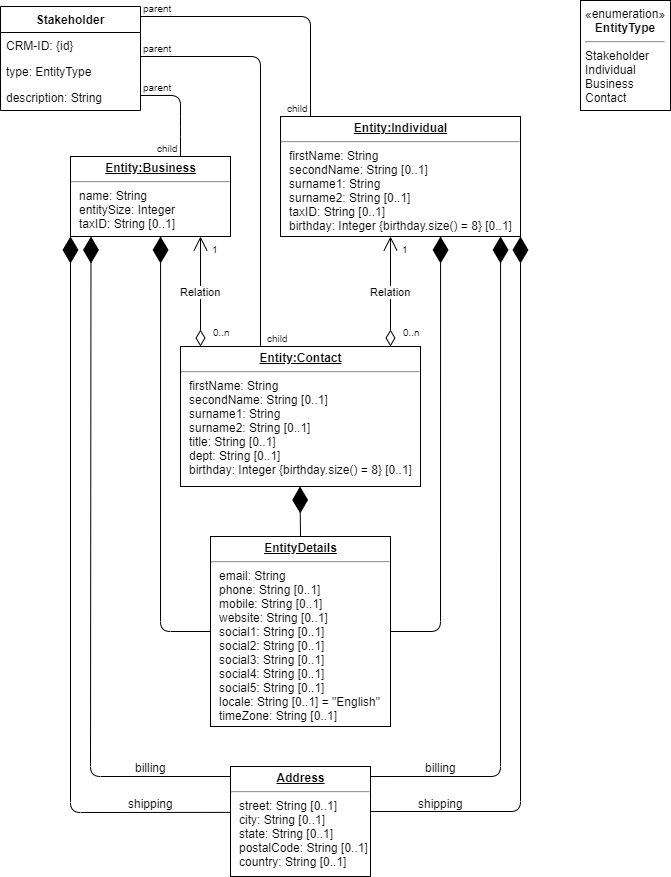


Figure 1: Sample Storyboard

# UML Diagram (Change title to fit needs)

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



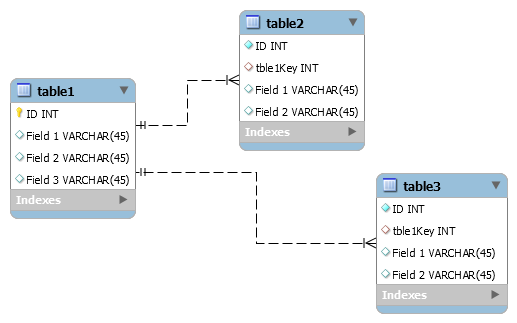


Figure 2: Sample Database

# GUI (Change title to fit needs)

Provide a mock-up of the proposed GUI forms that will be used in the proposed solution. Also, clearly indicate where the GUI components point inside the application.

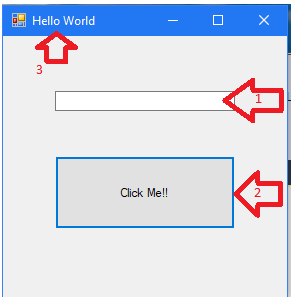


Figure 3: Sample GUI Mock-up

|  |  |  |  |
| --- | --- | --- | --- |
| GUI Control Mapping | | | |
| ID | Control | Property | Data Source |
| 1 | Textbox | On application open text = “” or null | NA |
| 1 | Textbox | On click of button text = “Hello World” | Internal Variable |
| 2 | Button | On click change text of textbox 1 to “Hello World” | Internal Variable |
| 3 | Form | Text= “Hello World” |  |

# Testing

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

# Testing Type

In addition to unit testing, the software will be subjected to the following feature tests prior 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 network. A test database must be running and available via networking. |
| Steps:   1. From the main GUI, choose to create a new stakeholder. 2. Choose a 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. |
| 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 and Canada. 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. |