|  |
| --- |
| Snapple Computers and Software |
| CRM System |

|  |
| --- |
| Robert J Robinson  4-21-2022  Version 1.0.0 |

Contents

[A. Introduction 3](#_Toc105253415)

[A.1. Purpose Statement 3](#_Toc105253416)

[A.2. Overview 3](#_Toc105253417)

[A.3. Goals and Objectives 3](#_Toc105253418)

[A.4. Prerequisites 4](#_Toc105253419)

[A.5. Scope 5](#_Toc105253420)

[A.6. Environment 5](#_Toc105253421)

[A.6.1 Platform 5](#_Toc105253422)

[A.6.2 Database 6](#_Toc105253423)

[A.6.3 Programming Languages 6](#_Toc105253424)

[A.6.3 Browser Support 6](#_Toc105253425)

[B. Requirements 6](#_Toc105253426)

[B.1. Business Requirements 7](#_Toc105253427)

[B.2. Data Integerty and Audit Support 7](#_Toc105253428)

[B.3. User access 7](#_Toc105253429)

[C. SOFTWARE DEVELOPMENT METHODOLOGY 8](#_Toc105253430)

[C.1. Advantages of the waterfall method 8](#_Toc105253431)

[C.2. disAdvantages of the waterfall method 8](#_Toc105253432)

[C.3. Advantages of AGILE 8](#_Toc105253433)

[C.4. disAdvantages of AGILE 9](#_Toc105253434)

[C.5. Waterfall method Is best SUITED 9](#_Toc105253435)

[D. Design 10](#_Toc105253436)

[D.1. Soft Delete / Hard Delete Flow Chart 10](#_Toc105253437)

[D.2. UML Diagram 11](#_Toc105253438)

[D.3. GUI (Change title to fit needs) 12](#_Toc105253439)

[E. Testing 13](#_Toc105253440)

[E.1. Testing Type (change name to fit your needs) 13](#_Toc105253441)

[E.1.1. Stress Test 13](#_Toc105253442)

[E.1.2. Browser Compatability Test | Blackbox testing practice 14](#_Toc105253443)

[E.1.3. Soft delete record test | Whitebox testing 15](#_Toc105253444)

# Introduction

In 1987, the early days of computers and connected devices, software solutions were being developed to replace the then popular Rolodex system. Customer relationship management systems (CRM) proved to be a successful replacement of a rolodex. CRMs evolved over the years, and many features that exist today were not even thought of in the early 1990s. Today, CRMs like Salesforce, Act, Oracle’s NetSuite power the biggest companies with tens of thousands of customers allowing them to maintain contact, run sales reports, and track any data needed to build strong relationships. The American Video Game Company needs a fast and power CRM that is accessible from anywhere. This proposal describes a CRM that will be a cloud-based web application built on top of Heroku, a leading cloud platform that has prebuilt integrations that will allow for rapid development and allow the ability to expand as needed by the business. Current business needs for the American Video Game Company include sales tracking, customer engagement, micro-transaction records, and the ability for admins to maintain this data from anywhere.

# A.1. Purpose Statement

This document will outline a detailed proposal on how we will meet the needs of the American Video Game Company. Covered in this proposal is an outline on a cloud first approach to build a sustainable application that can grow as the dammed increases. The proposal will also discuss several components related to data integrity with the use of hard and soft delete via user authorization, and audit tracking for government compliance.

# A.2. Overview

AVG has seen substantial user growth in recent years, 42% over the previous two years. This explosion of popularity has caused the current CRM system to be almost un-usable to an end-user, and the company needs a reliable and scalable design. Users report that the current system is slow and often crashes, sales data is lost, and it has no API that other teams within the company or third-party companies could use to enable partnerships with other systems more efficiently. The system is outdated and needs to be rebuilt from the ground up while preserving the existing data and protecting some workflows. With the proposed solution, all of the pain points will be resolved but building out a custom designed CRM that will allow quick and reliable data lookup, preservation of existing data, and the ability to connect and expand into other systems within the company.

# A.3. Goals and Objectives

The requirements document expressed concern for several areas of focus. The CRM needs to scale with minimal intervention seamlessly. The system should also preserve the existing data and allow for easy entry of new data. The system also needs to meet the goals as outlined in the requirements document. This service should also have entry points so that other internal and external systems can interact with the CRM to achieve business goals.

These goals can be achieved through the following objectives:

* The system will consolidate all contact and business information into one place, improving efficiency.
* Be able to generate reports on the company’s activities and interactions with contacts.
* Ability to control access to features based on roles and permissions for the company's internal and remote users.
* The new system should enable access to 3rd party marketing companies under contract.
* Can manage activities and track sales.
* Integrates with other systems to allow for sharing of data.
* Has industry-leading and government-compliant security practices.
* It can be enhanced and scaled with minimal intervention.

# A.4. Prerequisites

Before work can begin and these goals can be achieved, there need to be some perquisites resourced. Firstly, Heroku will be the cloud provider, and thus will need to generate the proper accounts and start to provision the required resources. A fast software framework, Ruby on Rails, will be used to build the system. This framework has a vast footprint allowing for rapid development and integration. And lastly, some AWS resources will need to be provisioned in order to meet the scale that is needed in this project.

|  |  |  |  |
| --- | --- | --- | --- |
| Number | Prerequisite | Description | Completion Date |
| 1 | Heroku | A cloud-based platform as a service is needed due to the scaling and maintenance needs of the system. Heroku is spread out across the United States and has many tools relating to database management, network management, user-based access, and proper audit logs. Built on top of one of the world's largest cloud providers, Amazon web services, Heroku will scale quickly and keep costs low. | 30 JUN 2022 |
| 2 | Ruby on Rails v. 7 w/ Ruby 3.0.1 | Ruby On Rails is a popular web framework that allows developers to build and scale applications quickly. Ruby, the language behind this framework, provides a syntactically forgiving syntax and has a vast community that supports all members, often called rugbyists. Rails, a "Full-Stack" framework, will handle the presentation and business logic and interact with any back-end store via ruby gems packages. | 30 SEP 2022 |
| 3 | AWS Aurora: Postgres | As AWS built their "serverless" stack, they introduced a serverless database called Aurora, powered by a Postgres engine. As the back-end store for all the required data, this service will scale with the application and keep maintenance costs low, as AWS will handle all of that. | 15 JULY 2022 |
|  |  |  |  |

# A.5. Scope

This proposal will focus on building and maintaining this product service to AVG.

* Hard and Soft Delete
* Audit Logging
* Browser Compatibility
* Load testing application for >500 active users.
* Cloud-Native Application Development

This proposal will not focus on:

* Sale Tracking
* Report Generation
* Contact Management
* Data Types

# A.6. Environment

## A.6.1 Platform

Our primary infrastructure will be built on Heroku, a cloud-based platform-as-a-service. Heroku has an easy-to-navigate dashboard that allows users of any skill level to deploy and scale their applications without all the overhead of system administrators, DevOps engineers, or site reliability engineers. Heroku describes its web servers as "dynos." Dynos are isolated, virtualized Linux containers designed to execute code based on a user-specified commands. This containerization allows for rapid and seamless scaling while also preserving the security of having a standalone server. Once deployed, the application has a configurable auto-scaler that will scale up or down as needed. This auto-scaling allows for peak traffic to go unimpeded, and when times are slow, scale back active dynos to save on resource allocation and costs. The initial distribution starts the auto scaler at four dynos and scales up to as many as thirty dynos as demand increases.

## A.6.2 Database

Amazon Web Services has a type of database engine called Aurora. Aurora uniquely describes itself as a serverless database, which scales up, like how an auto-scaling group would allocate more servers, when the traffic demand is increased. Removing the need to constantly manage the database infrastructure, this service saves costs and allows lower latency to the end users and services that maintain a connection. Aurora uses the open-source Postgres SQL database engine.

## A.6.3 Programming Languages

The application itself will be built using the Model-View-Controller architecture with the popular web framework Ruby on Rails. Version 7, the latest version, provides a wide range of packages that will allow for rapid development of the CRM application and quickly deploys to the Heroku platform. The CRM application will be robust and fast with built-in security best practices and plenty of support for third-party vendors. Rails handle all database interactions (including database migrations) and provide a secure API for vendors to interact with. The presentation layer, built with JavaScript and HTML5, will be accessible from any modern browser (Chrome, Firefox, Safari, Brave) via desktop or any mobile application.

## A.6.3 Browser Support

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

• Android 4.0 Chrome

• I.E 9 and above ( Edge )

• Safari 6.0

• iOS7 Safari

• iOS7 Third Party Browsers (Chrome and Firefox)

• The latest Chrome and Chromium

• The latest Firefox

• Mobile & tablet

# Requirements

As the requirements document outlined, the new CRM service will scale with ease, allow for easy integration with current and future third-party services through a robust API, and will be easy to use and modify. The following section will outline those requirements.

# Business Requirements

The following requirements will be met by the proposed solution:

* Ability to “soft-delete” records in the database and “hard” delete if the user has proper authorization
* Ability to audit access of users by recording activity
* Will be deployed to the cloud via the Heroku Platform
* Will be able to support 500 users concurrently
* Supported by modern browsers and operating systems

# Data Integerty and Audit Support

The system will be designed using a RDBMS database. This database will have columns defined for hard and soft delete. Checkpoints in the software will determine a user’s ability to modify these options. An example would be a regular user may have the ability to “soft delete” a record, on the datastore this would set a timestamp on the “deleted at” column (see fig. 2), thus rendering it nonvisible to other general users. Users with higher privileges, power users, will still be able to view these soft deleted records and then make the decision to completely remove the record from the datastore (see fig. 1). Other tables will also be defined so that power users can view an audit log, a special record in the datastore, of all activity relating to records. This will assist in maintaining compliance with various government data compliance requirements.

# User access

As the American Video Game company is growing, we decided that a cloud-native solution through Heroku would be the most saleable and cost effective. This will meet the requirement of being able to be accessed quickly and easily. Another requirement that this approach resolves that of the needing the ability to support 2000 users with 500 of them being connected at the same time. Because of the distributed nature of cloud solutions, the load balancers built in Heroku will allow for well over the 500 concurrent requirements. Building this system with Ruby on Rails, the application will generate the needed html and JavaScript that can be access for any modern web browser, on operating systems such as Windows, MacOS, and Linux distribution, thus fulfilling the requirement of supporting various browsers and operating systems.

# SOFTWARE DEVELOPMENT METHODOLOGY

The software development methods proposed for this project include the Agile and waterfall methodologies. The agile method was made famous by the "Manifesto for Agile Software Development," which defined and explained twelve principles relating to the development of software projects. The waterfall method, the one selected for this project, is a sequential development process that flows like water down a hill, cascading into the next phase of the project's steps.

# Advantages of the waterfall method

As the requirements document is defined as the product, this provides a clear structure for creating this system. From the ground up, the system is built in the planning phase, down to the buttons that will be selected. The waterfall method is predictable and stable, as all aspects of the project are planned out before any code is written. This process will allow for a clear testing framework to check against and create documentation beforehand so that the code is as described.

# disAdvantages of the waterfall method

Some disadvantages of the waterfall method include the possibility of being inflexible. Because all the planning is done beforehand, making changes once the project can add complexity and change everything downstream. If AVG suddenly needs to change something, this will prove not easy, depending on the project's phase. Another disadvantage is that testing is only done towards the end of the project, and if there are issues discovered, it could set back the project. The final disadvantage would be that the project will take a substantial amount of time and will only interact with AVG at the end, disconnecting them from the process.

# Advantages of AGILE

The advantages of Agile development allow for much flexibility, reduce errors, and allow a team to change on the fly. AVG could benefit from this methodology because agile would provide an iterative approach to building and developing features, enabling continuous deployment and constant feedback from AVG and its users.

# disAdvantages of AGILE

Some disadvantages would be:

* Ongoing development means that the project never ends
* Limited processes and controls

# Waterfall method Is best SUITED

The waterfall method allows a total design upfront for the application. This will allow stakeholders the foresight of issues that may stem and provide a one-time development cost for the company. As outlined, receiving all the requirements at the start of a project allows managers and engineers time to flesh out the project and provide detailed timelines when delivery can be accomplished. This allows for better resource allocation, early test plans to make sure that the deliverable is up to standards. The company can test fix the application before it is shipped to the company.

# Design

The design documents that follow show the various parts of the Soft delete including a flowchart, a UML diagram describing the class and database structures, and a representation of the GUI that will be used by the end user.

# Soft Delete / Hard Delete Flow Chart

The flowing image show the path that can be taken to either hard delete a file or soft delete a file.

Diagram

Description automatically generated

Figure 1: Deletion Flow Chart

# UML Diagram

The following figure is a UML describing the user’s relationship to the clients and their games. This diagram also shows that `Client`s and `Game`s generate a `AuditLog` when any activity is acted upon. The AuditLog is a polymorphic table that allows for expansion of the auditable models in the database.

Diagram

Description automatically generated

Figure 2: Basic Record UML

# GUI (Change title to fit needs)

Below is a mock-up of the dashboard of the CMS. It depicts the clients table and show the actions that can be taken.

Graphical user interface, application

Description automatically generated

Figure 3: Sample GUI Mock-up

|  |  |  |  |
| --- | --- | --- | --- |
| GUI Control Mapping | | | |
| ID | Control | Property | Data Source |
| 1 | Button | On click of button text = “Search”, will dynamically query table for record | Text box #3 |
| 2 | Button | On click of button text = “Delete”, will process logic in Fig. 1 to hard or soft delete a record | Table Variable |
| 3 | Text Box | When a user types in record search terms used in conjunction with the Search button #1 | Internal Variable |
| 3 | Label | Text= “Power User” Users Auth status |  |

# Testing

Provide a brief introduction to the proposed testing solution. The tests need to be from 3 utterly different functionality aspects, and testing the same element with slightly different criteria is unacceptable.

\*\*Note: *Add and remove subsections as needed to cover all the testing needs.*

# Testing Type (change name to fit your needs)

Provide a brief introduction paragraph.

# Stress Test | White and Black Box Testing

|  |
| --- |
| Requirement to be tested  A stress test will conduct on the application once it has been deployed to verify that Heroku’s load-balancing meets the requirements outlined by the company. |
| Preconditions: Conditions that must be present before test case can successfully run  The testing coordinator will arrange for 600 testing users to access the application at set intervals while the testing coordinator documents responsiveness and reviews metrics that Heroku provides on its primary dashboard. This time is set to start at 1300 (T) hours today, 200 users will login and start using the site in any way that the individual tester wishes to. At T+10min, 200 more pre-identified users will sign on and do the same, followed by a final group T+20min. All users must have verified that they have accounts in the CMS system, and they have adequate internet access. Test users have been instructed to document their results and forward to the testing coordinator at the conclusion of the test. |
| Steps: The steps the tester must execute to test the feature.   1. At 1300 the test users have been instructed to begin. 2. Testing coordinator has logged into the Heroku dashboard and has set the refresh rate on the metrics to every 2 seconds. 3. At every increase the testing coordinator will document the CPU usage, memory usage, and throughput of the application. 4. Periodically throughout the test, the test users must document their results 5. At the conclusion of the test, the testing coordinator will analyze all of the user’s documentation and generate a report. |
| Expected results:  Users should be able to access the web application without incident. The Test coordinator should notice that the auto-scaling feature should allocate more resources as demand increases. |
| PASS:  Testing coordinator reports that the stress test was completed without incident. During the test, not a single user reported any degradation in responsiveness, and the application always has a less than 500ms response time. The test coordinator also noted that Heroku only allocated half of the defined resources for this test inferring that the system could support many more than the requirement of 500 users. |

# Browser Compatability Test | Blackbox testing practice

|  |
| --- |
| Requirement to be tested  The requirements state that the application built must be compatible with a verity of browsers. Once the application is ready for delivery a user will manually test this aspect of the requirements. |
| Preconditions: Conditions that must be present before test case can successfully run   1. Users must download and install the following web browsers.  * On MacOS – Chrome, Safari, and Firefox * On an iOS Device – Chrome for iOS, Firefox for iOS, and have the most up to date version of Safari  1. User must ensure that they have access to the internet, and are not blocked by firewalls or behind any proxy servers. |
| Steps: The steps the tester must execute to test the feature.  The following steps must be repeated on every device and browser listed   1. Open browser. 2. Verify connection to the internet by navigating to https://google.com 3. Navigate to <https://cms.americanvg.com> . 4. Verify that the CMS service loads with no errors. |
| Expected results: Expected results and any side effects such as updating a database, writing to a file, etc.  The browser should load without errors to all browsers. The CMS should be free of broken links, and should load in a time >500ms. |
| PASS:  The test showed that all browsers were supported and had a responsive loading time. No reported issues with the navigation and application were free from broken links. It is concluded that the application is accessible from both mobile and desktop devices on any modern browser. |

# Soft delete record test | Whitebox testing

|  |
| --- |
| Requirement to be tested  User can soft delete a record in the database via the GUI |
| Preconditions: Conditions that must be present before test case can successfully run  Tester must have a User account without poser user credentials. Application must be deployed and accessible. Tester must have access to the database via a SQL GUI to validate the test. There must be at least one client record in the database. |
| Steps: The steps the tester must execute to test the feature.     1. In the SQL GUI – run following SQL query `SELECT \* FROM clients` 2. Verify that the record returned has “deleted at = NULL” 3. Open a browser and navigate to <https://cms.americanvg.com> . 4. Login by filling out username and password in the provided text boxes. 5. Click on the red “Delete” button next to the client record in the table 6. In the SQL GUI – run following SQL query `SELECT \* FROM clients` 7. Verify that the “deleted at” column is not set to the current timestamp |
| Expected results: Expected results and any side effects such as updating a database, writing to a file, etc.  ` SELECT \* FROM clients ` returns a table of clients, and the selected client for deletion remains in the database with the deleted\_at column is now populated with a timestamp. |
| PASS:  Tester reported that the button worked, and that the record was still in the database with the timestamp updated from NULL to the current timestamp. This concludes the requirements and tests now show that the product is ready for delivery. |