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

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

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

Customer Relationship Management (CRM) systems need to be fast and available from anywhere. This CRM will be a cloud-based web application built on top of Heroku, a leading cloud platform, 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, that starts with the introduction, is to outline the proposal to build a highly scalable CRM for the American Video Game Company (AVG). The following will outline the requirements, define the software methodology that will be used, design of the system, and how this system will be tested.   
  
TODO: Finish this later.

# A.2. Overview of THE PROBLEM

In recent years, AVG has seen substantial user growth; 42% over the previous 2 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 system. Users report that the current system is slow, and often crashes, sales data is lost, and has no API that other teams within the company, or third party companies could use to enable partnerships with other systems more easily. The system is outdated, and needs to be rebuilt from the ground up, while also preserving the existing data and protect some workflows.

# A.3. Goals and Objectives

The requirements document expressed a concern for several areas of focus. The CRM needs to seamlessly scale with minimal intervention. 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 users, both internal and remote.
* The new system should enable access to 3rd party marketing companies under contract.
* Can manage activities and tracks sales.
* Integrates with other systems to allow for sharing of data.
* Has industry leading and government compliant security practices.
* Can be enhanced and scaled with minimal intervention.

# A.4. Prerequisites

Before work can begin, and these goals can be achieved, there needs to be some perquisites resourced. Heroku will be the cloud provider, and the system will be built with Ruby on Rails, a popular ruby based web framework.

|  |  |  |  |
| --- | --- | --- | --- |
| 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. Heroku is built on top of one of the largest cloud providers, Amazon web services, this solution will scale easily, but will also 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 the ability to build and scale applications quickly. Ruby, the language behind this framework, provides a syntactically forgiving syntax, and has a huge community that supports all members, often called rugbyists. Rails , a “Full-Stack” framework, will handle the presentation, business logic, and can interact with any back-end store via ruby gems packages. | 30 SEP 2022 |
| 3 | AWS Aurora: Postgres | As AWS has built our their “serverless” stack, they introduced a serverless database called Aurora, that is 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

As outlined in the CRM requirements document, this proposal will explain, in detail, of what will be covered in this agreement.   
  
The requirements document expressed a need for security, scalability, and the ability to build more robust services to allow for engagement with customers, sales tracking, and an ability to interact with various third-party vendors, this proposal will focus on building and maintaining this product as a service to AVG. Our company, Snapple computers, will create Heroku platform resources to which the SLAs for connectivity and security (<https://devcenter.heroku.com/articles/security-privacy-compliance>) under the shared model. Snapple Computers will develop and deploy the CRM application and will set up service portals should any part of the application need attention.

# A.6. Environment

Our primary infrastructure will be built on Heroku, cloud-based platform-as-a-service, a leading provider for web applications. Heroku has an easy to navigate dashboard, that allows users of any skill lever the ability to deploy and scale their applications without all the overhead of system administrators, DevOps engineers, or site reliability engineers. Heroku describes their web servers as “dynos”. Dynos are isolated, virtualized Linux containers that are designed to execute code based on a user-specified command. This containerization allows for rapid and easy scaling, while also preserving the security of having its own 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. All web service assets will be served from these dynos. The initial allocation with start the auto scaler at four dynos and scale up to as many as thirty dynos as demand increases.

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 easily deploys to the Heroku platform. With built in security best practices, and plenty of support for third party vendors, the CRM application will be robust and fast. Rails handles all database interactions (including database migrations) and provides a secure API for vendors to interact with. The presentation layer, that will be built with JavaScript and HTML5, will be accessible from any modern browser (Chrome, Firefox, Safari, Brave) via desktop or any mobile application.

The finial part of the infrastructure will be the AWS Aurora database. Only needing to provision and configure this service on AWS, this database will allow only a connection from our Heroku platform and will scale as demand scales.

# Requirements

As outlined in the requirements document, the following will provide a brief introduction on the business requirements, user requirements, and the functional requirements. This proposal has the focus on keeping the system as accessible as possible and allow for automatic scaling. The requirements document also outlined the need for the ability to soft delete files, keep an audit log of all actions taken. The business requirements of being able to forecast model and track sales, preserve previous sales from current data set, and handle user contracts, and the user requirements of handling even more user growth will also be outlined in the following segments.

# Business Requirements

The creation of this new application solution will require a data migration from the current solutions. In the presentation layer there will be a portal that will allow for the upload of excel sheets of current data, that will populate the new database. This will allow for metrics and reports to be built on new and previous data via a special administration portal. Business analysists can use this data, accessed form a special portal, the ability to write domain specific SQL queries that can return the forecasting data needed.

# User Requirements

The new system will also allow for a single place to manage customer and vendor contracts, by alerting the AVG users of the status of the contract, when it is set to expire, and provide any outstanding data needed for a contract to be fulfilled. All contract information, its metadata, and any associated data will be stored in the aurora database.

# Functional Requirements

Audit Logs were a key requirement needed, and this will be accomplished at the data layer. Rails will run commands on data models when conditions are met. This will include a special database model called an Audit Log. This will record any access or changes made by a logged in user.

Soft Delete is also a requirement met by placing a special type of flag on the database record. These soft-deleted will not be visible to a normal user, admins can still use this data via a database console to use as needed in forecasting sales models if needed.

# NonFunctional Requirements

Using the above-mentioned platform as a service, will allow this system to scale as the business users and end users grow. Being cloud based also has the benefit of allowing access from any device from anywhere without the need to maintain hosted on prem servers. This means, that with a sudden uptick in users from anywhere in the company’s domain, these autoscaling features will permit rapid concurrent access with low latency. These services allow business goals to continue to function at excessively busy times.

# SOFTWARE DEVELOPMENT METHODOLOGY

The software development methods proposed for this project include the Agile method and the waterfall method. The agile method was made popular by the “Manifesto for Agile Software Development”, that defined and explained twelve principals 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 projects’ steps.

# Advantages of the waterfall method

As the requirements document is defined as the product, this provides a clear structure on what and how to create this system. Going 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. Moving through this process will allow for a clear testing framework to check against, create documentation beforehand, so that the code is exactly 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 before hand, making changes once the project can add complexity, and change everything downstream. If AVG suddenly has the need to change something, this will prove difficult, depending on what phase the project is in. Another disadvantage is that testing is only done towards the end of the project, and if there are issue discovered, could set back the project. The final disadvantage would be that the project will take a substantial amount of time and will only be interacted with AVG at the very end, disconnecting them from the process.

# Advantages of AGILE

The advantages of Agile development allow for a lot of flexibility, reduces errors, and allows 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. This would allow continuous deployment, and constant feedback from AVG and its users.

# disAdvantages of AGILE

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

Provide a brief overview of the proposed design.

*Note: These subsections may be copied, rearranged, and modified to fit the needs of the solution. At least two visual representations of your design need to be present.*

# Storyboard or Flowchart (Change title to fit needs)

Provide a storyboard or flowchart of the application.

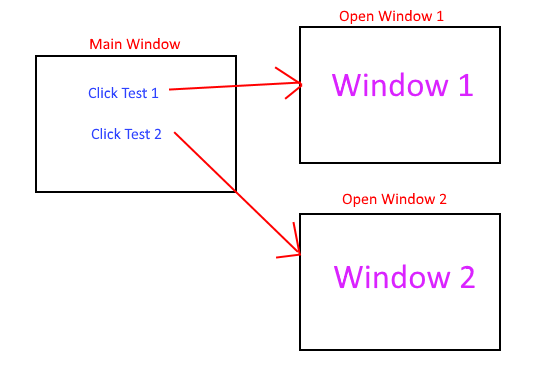


Figure 1: Sample Storyboard

# UML Diagram (Change title to fit needs)

Provide a set of UML diagrams that cover the proposed solution. This can include but is not limited to class diagrams, database diagrams, and use case diagrams. Also, ensure that all diagrams are clearly discussed and noted.

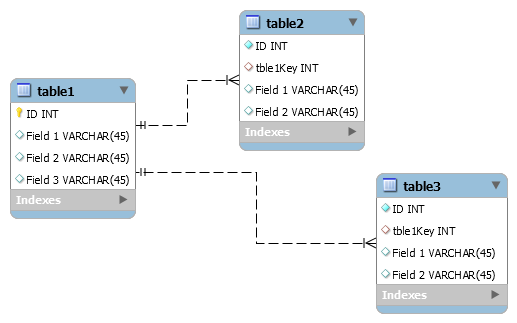


Figure 2: Sample Database

# 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.

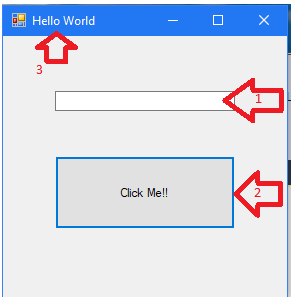


Figure 3: Sample GUI Mock-up

|  |  |  |  |
| --- | --- | --- | --- |
| 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

Provide a brief introduction to the proposed testing solution. The tests need to be from 3 completely different functionality aspects. Testing the same aspect with slightly different criteria is not acceptable.

\*\*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.

# Test Name 1

|  |
| --- |
| Requirement to be tested |
| Preconditions: Conditions that must be present before test case can successfully run |
| Steps: The steps the tester must execute to test the feature. |
| Expected results: Expected results and any side effects such as updating a database, writing to a file, etc. |
| Pass/Fail: Mark whether the test case passed or failed. The results can be compiled and used to determine if the application is ready for delivery/release. |

# Test Name 2

|  |
| --- |
| Requirement to be tested |
| Preconditions: Conditions that must be present before test case can successfully run |
| Steps: The steps the tester must execute to test the feature. |
| Expected results: Expected results and any side effects such as updating a database, writing to a file, etc. |
| Pass/Fail: Mark whether the test case passed or failed. The results can be compiled and used to determine if the application is ready for delivery/release. |

# Test Name 3

|  |
| --- |
| Requirement to be tested |
| Preconditions: Conditions that must be present before test case can successfully run |
| Steps: The steps the tester must execute to test the feature. |
| Expected results: Expected results and any side effects such as updating a database, writing to a file, etc. |
| Pass/Fail: Mark whether the test case passed or failed. The results can be compiled and used to determine if the application is ready for delivery/release. |

# Sources

Place the sources that you used here.

*Note: See the sources section in the requirements and rubric. If you did not use any outside sources, you may delete this section.*