|  |
| --- |
| Eggsceptional Software Solutions |
| Cracking the Market |
| ESS’s solution to American Video Game Company CRM Problem |

|  |
| --- |
| Brandon Egbert  2-14-2022  V1.1 |

Contents

[A. Introduction 3](#_Toc98262070)

[A.1. PUrpose Statement 3](#_Toc98262071)

[A.2. Overview of THE PROBLEM 3](#_Toc98262072)

[A.3. Goals and Objectives 3](#_Toc98262073)

[A.4. Prerequisites 3](#_Toc98262074)

[A.5. Scope 4](#_Toc98262075)

[A.6. Environment 4](#_Toc98262076)

[B. Requirements 5](#_Toc98262077)

[B.1. Integration 5](#_Toc98262078)

[B.2. Hosting 5](#_Toc98262079)

[B.3. Reporting 5](#_Toc98262080)

[B.4. Versioning and data management 6](#_Toc98262081)

[B.5. Functional Requirements 6](#_Toc98262082)

[C. SOFTWARE DEVELOPMENT METHODOLOGY 6](#_Toc98262083)

[C.1. Advantages of the waterfall method 6](#_Toc98262084)

[C.2. disAdvantages of the waterfall method 6](#_Toc98262085)

[C.3. Advantages of the agile method 6](#_Toc98262086)

[C.4. disAdvantages of the agile method 6](#_Toc98262087)

[C.5. best SUITED 6](#_Toc98262088)

[D. Design 8](#_Toc98262089)

[D.1. Storyboard 8](#_Toc98262090)

[D.2. Rough database sample 9](#_Toc98262091)

[D.3. GUI 9](#_Toc98262092)

[E. Testing 10](#_Toc98262093)

[E.1. Testing Type (change name to fit your needs) 10](#_Toc98262094)

[E.1.1. Import existing data 10](#_Toc98262095)

[E.1.2. New Interaction 11](#_Toc98262096)

[E.1.3. Order Management 12](#_Toc98262097)

# Introduction

Eggsceptional Software Solutions (ESS) proposes that American Video Game Company (AVGC) use our proprietory CRM software solution ***Eggforce***. Eggforce has been used by hundreds of companies for hundreds of thousands of sales representatives to track and manage billions of transactions. With a simple UI and incredibly flexible framework, Eggforce was built to fit any CRM need out there.

# A.1. PUrpose Statement

The purpose of this document is to propose the CRM software solution Eggforce to AVGC, and collaborate to ensure AVGC’s needs are met.

# A.2. Overview of THE PROBLEM

The current environment is outdated, inaccessible, and requires significant effort to track and monitor sales. Many of the current processes are immutable because the current systems can’t adapt as new requirements are found as the business grows.

# A.3. Goals and Objectives

Import existing data to keep records and remove the need to maintain the old data

Use Eggforce, a cloud based solution so it is accessible anytime, anywhere

Maintain historical records for AVGC’s needs

Allow customization as needed

Centralize all data

Remove the risk and cost to AVGC

Provide excellent support and training on Eggforce

Provide access to several different reporting tools

# 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 |
| 1 | Contract | Both companies must outline agreeable terms and expectations, so that there is no confusion or ambiguity regarding each other’s role in the project. | 3/1 |
| 2 | Identification | ESS must work with AVGC to identify and understand the exact needs of AVGC so that the needs of AVGC are met | 4/1 |

# A.5. Scope

Eggforce will help to manage sales, clients, transactions, and all of the data related to these. ESS will provide initial configuration of the program, including reasonable customizations. After implementation, ESS will provide support for any issues. ESS will also manage the application and data access with high availability. ESS and Eggforce will not be used for any non-sales related functions.

# A.6. Environment

Eggforce will be a SaaS solution and have very stringent requirements because it is used for sales. There will need to be at least 99.5% availability, and be accessible via several web browsers, including but not limited to Chrome, Chromium, Firefox, IE9+, Safari 6+, iOS7 Safari, iOS7 Chrome and Firefox, Android 4.0 Chrome, and the default web browser for the other mobile and tablet devices used by AVGC. Eggforce will also need to use high-level encryption, the latest security protocols, and constant monitoring for potential risks.

# Requirements

AVGC has several requirements that must and should be met. Eggforce will be adapted to address as many of these needs as possible. Eggforce will also attempt to meet any remaining requirements within reason should they not already be met. Below are some more detailed requirements that Eggforce will meet:

# Integration

Eggforce was built with Microsoft integration, including Active Directory (on prem, cloud-based, and hybrid environments) and Office 365. On-premise Exchange can be configured as well, but it is discouraged. Contacts in Eggforce can be enabled to sync with AVGC’s Outlook contact and phone directory, and will be exportable as well. Custom contacts can be included or excluded at AVGC’s discretion. Data can also be exported into many different file types including txt, xlsx, csv, docx, and pdf. Calendar syncing can also be enabled for appointment reminders and automatic video conferencing links. Data can be imported as well to reduce the time it takes to update multiple records

# Hosting

Eggforce uses a SaaS model, and is subscription based. The software has multiple redundancies, data centers, and load-balancing servers to ensure QoS meets expectations. Outages, SLAs, upgrades, customization post-implementation, support, and maintenance will be negotiated immediately after the initial contract is signed, but before any design is done to ensure clarity and proper expectations are set. ESS is typically willing to adopt the policies provided by any client within reason, and strives to follow best practices where possible, such as having dev, QA, and production environments to minimize issues.

ESS is based in the USA, but operates internationally. Each data center only holds data from the country it is in to follow local regulations as accurately as possible. Access control groups can be created to prohibit the data from being accessed unless it is from within the US. Separate groups can be created to allow access from anywhere, but will only be done at AVGC’s discretion.

# Reporting

Eggforce will have the ability to provide several different reports with customizable degrees of detail. Dashboards, summary reports, custom filters, analysis, and many other features to help with forecasting and client management will be available. These reports will be exportable and can be adjusted based on role.

# Versioning and data management

All data in Eggforce will be controlled by role. Some roles can edit records, some delete records, and others restore records. All changes will be tracked and versions available for recovery. There will be safety nets to prevent improper deletion.

# OS and Browser support

Eggforce is designed to be run through a web browser. Chrome, Chromium, Firefox, IE9+, Safari 6.0, and most mobile browsers. There is a mobile app and mobile webpage for maximum accessibility and availability

# SOFTWARE DEVELOPMENT METHODOLOGY

AVGC has selected the waterfall software development methodology for this project. This section will compare the waterfall method against the agile method and determine which one is better suited for this project

# the waterfall method

The waterfall method could work well for this project because Eggforce is a generic framework that won’t meet AVGC’s needs as-is. This allows ESS to start with a mostly blank slate to create a custom product. Since everything will be flushed out prior to the development, the scope and KPIs can be clearly defined.

The waterfall method needs all of the details and specific requirements explicitly described from the start. If these are at all ambiguous, it can add months to the development time. This can also create a pseudo scope-creep, since any additional requirements that were not expressed initially would add additional work. It is also dependent on the previous stakeholders doing their tasks correctly and in a timely manner. No stage can start without the previous one being close or at completion

# the agile method

Agile could output a solution faster and more efficiently. While the more complex tasks and requirements are in progress, the development team can start working on some of the more basic requirements, like a basic contracting functionality. As new needs and functionality are discovered, they can be implemented at a later time, rather than needing the full scope initially.

As with the waterfall method, not having specific requirements from the start can be a hinderance. If the scope is ambiguously defined, some work may need to be re-done over and over again because it relied on certain pieces that are no longer necessary or incompatible

# best SUITED

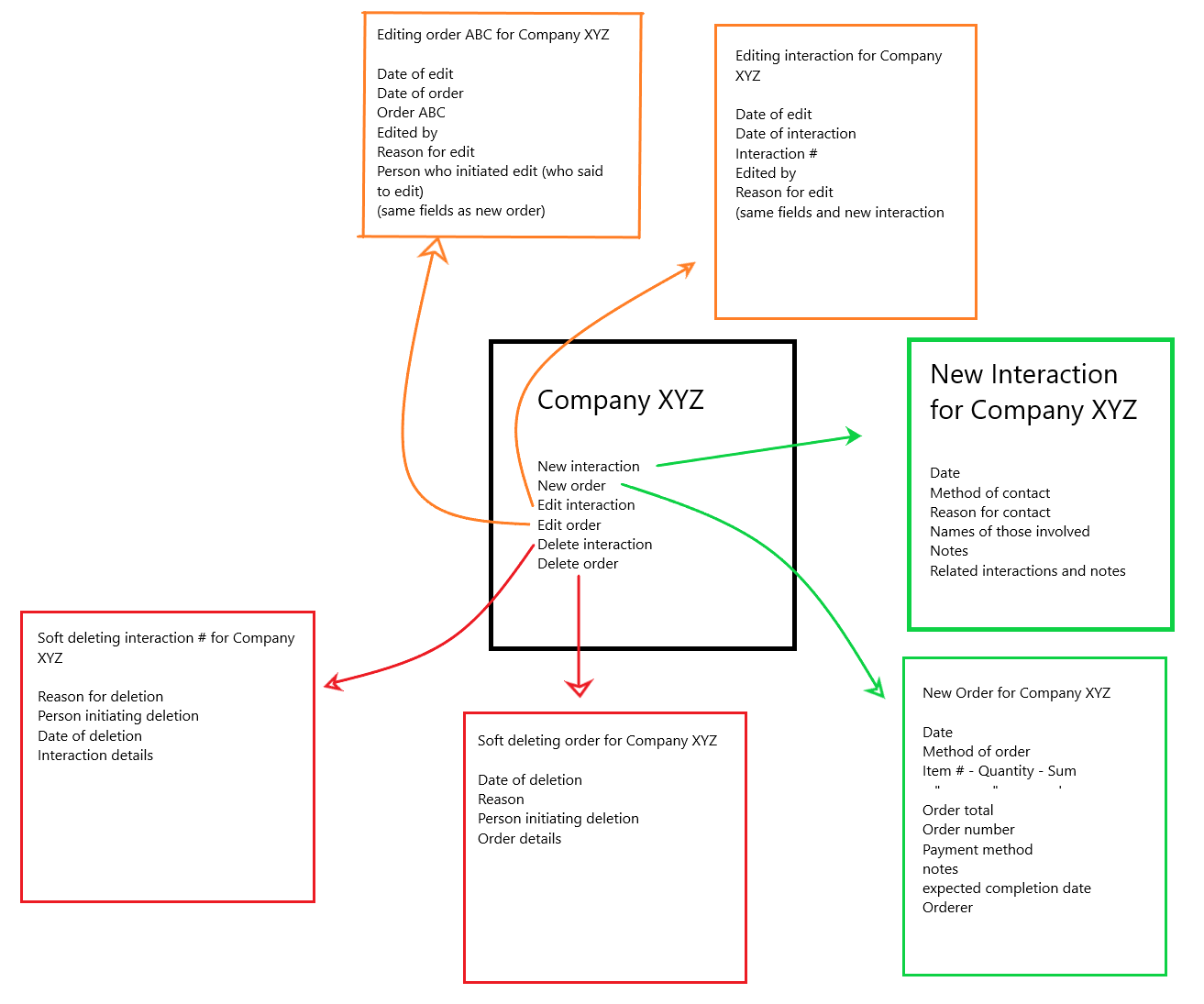
I think the waterfall method would be the best for this project. The requirements are already mostly defined, and AVGC seems to have a good grasp on what they need. While agile would work well too, I think that finishing the requirement and design phase would be quick and unambiguous. The waterfall method works best when the requirements are already defined, as they are here. There is a clear linear progression that can move to the next step as the previous one is completed. In this case, ESS can redirect resources to other projects as those personnel complete their section of responsibility.

The project wouldn’t really benefit from agile, because there isn’t really an opportunity to start on some basic tasks. Everything mostly depends on the previous step being completed, and already should have a good understanding of the requirements. Using agile would just add needless complexity and may require re-working features if requirements weren’t accurately established

# Design

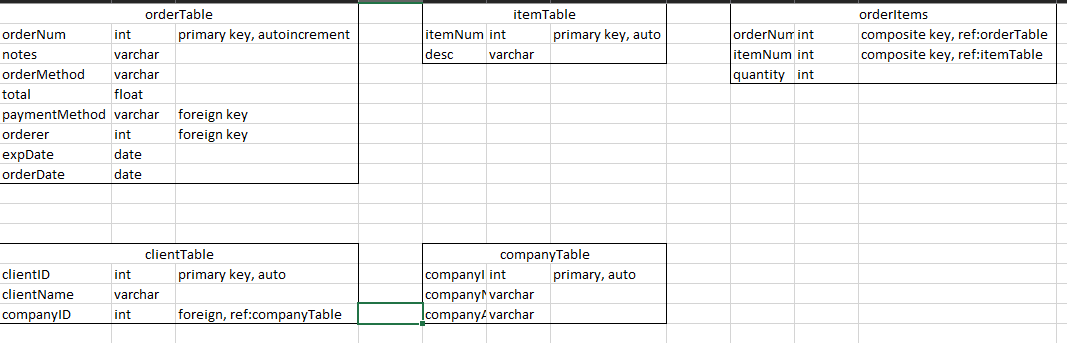
# Storyboard

This is a sample diagram starting at the home page of a company to provide context of how it might work. It has links to create, edit, or delete interactions and orders. It also shows some fields applicable to each option. This isn’t a representation of the final version, and subject to change. More detail will be completed during the design phase.



# Rough database sample

This sample UML diagram shows how the database might be structured. It follows the SQL best practices and the 3rd normal form. Most people won’t see the actual database structure, but it is important to ESS and AVGC that the database be easy to scale



# Testing

Below are 3 tests designed to test functionality and data integrity

# Unit test

This test will evaluate and import data from the pre-existing data as accurately and error-free as possible

# Import existing data

|  |
| --- |
| Requirement to be tested  Importing of data from current data set with no errors |
| Preconditions: Conditions that must be present before test case can successfully run  All applicable fields from old software need to be present in Eggforce  Have existing data exported from old database  Have dev database setup |
| Steps: The steps the tester must execute to test the feature.   1. Organize data for sql import (make sure any tables with references are completed last) 2. Validate data types for each field 3. Fill out sql commands for import 4. Run commands to import data into dev environment 5. After each table, check for errors 6. If errors occur, attempt to redo on that table if possible, and fix commands for prod 7. Once entire database is in, wipe database and create from full script |
| Expected results: Expected results and any side effects such as updating a database, writing to a file, etc.  Expected result is that database is fully operational with no errors |
| 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.  Pass if script has no errors  Fail if script has an error |

# e.2 Functional test

This test is designed to test the new interaction function and prevent clearly wrong inputs

# E.2.1 New Interaction

|  |
| --- |
| Requirement to be tested  New Interaction function |
| Preconditions: Conditions that must be present before test case can successfully run  Have dev environment setup  Have gui setup |
| Steps: The steps the tester must execute to test the feature.   1. Create new interaction with numbers for all fields, then submit 2. Create new interaction with letters for all fields, then submit 3. Create new interaction with special characters for all fields, then submit 4. Create new interaction with all blank fields, then submit 5. Create new interaction with reasonable inputs, then submit |
| Expected results: Expected results and any side effects such as updating a database, writing to a file, etc.  Only the numbers and reasonable inputs forms should appear in the database. The rest should throw and error and prevent submission |
| 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.  Application passes if it meets the expected results, and fails if bad data is entered into the database |

# e.3 Acceptance test

This test is to make sure that AVGC is satisfied with the permission levels for each role

# E.3.1 Order Management

|  |
| --- |
| Requirement to be tested  Order management for each different access type |
| Preconditions: Conditions that must be present before test case can successfully run  ACLs in place  Edit order page in place  Orders already in database (real or fake) |
| Steps: The steps the tester must execute to test the feature.   1. Attempt to archive 3 orders with each access level 2. Attempt to un-archive 3 orders with each access level 3. Attempt to edit 3 orders with each access level 4. Attempt to rollback changes to 3 orders with each access level 5. Attempt to soft delete and restore 3 orders with each access level 6. Attempt to hard delete 3 orders with each access level |
| Expected results: Expected results and any side effects such as updating a database, writing to a file, etc.  Roles that should have the ability to do each task can, and roles that shouldn’t have that ability don’t |
| 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.  If all steps are successful, it passes. If a single one fails, the test fails |