Design Template

Devin Perdue

Best Software Company

Software Project Template

American Video Game Company Proposal

5/2/23

Version 1.2

WESTERN GOVERNORS UNIVERSITY

CONTENTS

A. INTRODUCTION

We at Best Software Company have developed and proposed a CRM solution that addresses your company's needs. In the following sections, we address several of your requirements, our proposed methodology, a design layout of your ticketing and order management systems, and adequate testing that shows our product works as intended.

A.1. PURPOSE STATEMENT

The purpose of this document is to propose a working product that addresses several of AVGC's requirements.

A.2. OVERVIEW OF THE PROBLEM

Sales have increased 42% in the past two years, leading to the company outgrowing its current system. A system that cannot accommodate the increasing customer base can lead to slower response times or system crashes. These issues will also lead to the inability to keep adequate records of company communication or inquiries. The company also has a disconnected set of custom-built tools in spreadsheets and database management systems. This disconnection decreases productivity and does not allow day-to-day activities to run smoothly.

A.3. GOALS AND OBJECTIVES

- Provide a fully functional order management system and the ability to track sales
- Robust security and the ability to keep customer information secure
- Develop a product that can be enhanced and scalable
- Provide a system that is easy to use, intuitive, and user friendly
- Have a future roadmap that provides updates and future development

A.4. PREREQUISITES

Number	Prerequisite	Description	Completion Date
1	Collect Data	Collect data on the five requirements and what is required with each.	1 Week
2	Download SQL	Download SQL Server 2022. This is needed to make the order management database.	3 days
3	Download Virtual Desktop	Download virtual desktop to perform compatibility testing in section E.	3 days

A.5. SCOPE

The proposed solution will cover the following:

- Develop an order management system that will turn quotes into orders and complete a sale
- Provide a ticketing system solution that will track all communication and inquiries to the company
- Make a product that is supported by all the latest operating systems and browsers
- Cloud-based model for hosting
- A cloud-based system that will support 2000 users and 500 users at peak times as well as be scalable

The following is out of the scope of this proposal:

- This proposal will not go into specific detail on robust security and how to develop an adequate security system for the end-users and AVGC
- Providing a UI library design that will hold all the available games AVGC has to offer
- We will also not propose a solution for how end-users can initiate the refund process if users are not happy with the product

A.6. ENVIRONMENT

- Latest Chrome and Chromium
- Latest Firefox
- I.E 9 and above
- Safari 13.0 and higher
- Mobile & tablet
- iOS10 Safari
- iOS10 Third Party Browsers (Chrome and Firefox)
- Android 7.0 Chrome
- SQL Server 2022
- AWS CLI Version 2

B. REQUIREMENTS

Requirements identify what a company needs. They must be clear, unambiguous, consistent, prioritized, and verifiable (Stephens, 2015, p.54-60). We at Best Software Company will provide solutions to several of your companies requirements, such as hosting, the ability to support 2,000 users and withstand 500 users at any given time as well as being scalable, having OS (Operating System) and browser support, a ticketing system that can track all communication and inquiries for contacts, and order management which will turn a quote into an order.

B.1. BUSINESS REQUIREMENTS

Business requirements are high level-goals for the project. For instance, one of your company's high-level requirements is to develop an order management system. We will create a database to turn a quote into an order and complete the sale. The database will also provide order tracking, taking orders, converting quotes to orders, reordering, part ordering, and customer self-serve. We will show the order tracking features work by providing functional testing.

B.2. USER REQUIREMENTS

User requirements refer to how the end user will use the product. For example, the ticketing system will allow end users to contact the company. The ticketing system will track all communication, including who called, the date/time, and other relevant details. The database will also keep an audit trail so that AVGC can view the information in the future. To show that our system will provide your company with this, we will perform functional testing, as seen in section E, to show how the ticketing system records all communication and provides an audit trail.

B.3. FUNCTIONAL REQUIREMENTS

Functional requirements describe the product's capabilities. For example, we will provide a cloud-based system with the capabilities to have support for most OS and browsers. In addition, our cloud-based system will provide support for the following:

Latest Chrome and Chromium

Latest Firefox

I.E 9 and above

Safari 13.0 and higher

Mobile & tablet

iOS10 Safari

iOS10 Third Party Browsers (Chrome and Firefox)

Android 7.0 Chrome

We will provide a compatibility test in section E to show that our system supports all these browsers and operating systems.

We will also provide a cloud-based model for hosting. We will provide a detailed plan addressing concerns for connectivity issues, SLAs (service level agreements), the potential for upgrades, custom development, 24/7 support, the ability to refuse upgrades if desired, and a plan for maintenance. We will outline all these points in detail in contracts that will show all the services we will provide you with and how.

B.4. NONFUNCTIONAL REQUIREMENTS

Nonfunctional requirements are about the application's behavior, such as performance, reliability, and security (Stephens, 2015, p.63). We will develop a cloud-based system that will support 2,000 users and withstand 500 users at any given time and be scalable to provide more support as your company grows and expands. Our system will support these users by being a cloud-based system allowing flexibility and scalability. We will prove this, as you will see in section E Testing. We will perform a performance test to show response time and CPU and memory storage as the product runs with different numbers of users.

C. SOFTWARE DEVELOPMENT METHODOLOGY

The development process is a set of steps and procedures a company performs to provide a product. There are several types of development models. However, the two main groups these derive from are predictive versus adaptive. Each has its benefits and disadvantages, as you will see below. As the name implies, predictive models predict what needs to be done beforehand, while with an adaptive model, you can adjust the goals and design as required throughout the project. The waterfall methodology is a predictive model where you finish each step before moving on to the next. The phases for this model are the requirements, design, implementation, verification, deployment, and maintenance (Stephens, 2015, p.267-270).

C.1. ADVANTAGES OF THE WATERFALL METHOD

The advantages of the waterfall method are predictability, stability, and cost-saving. With the waterfall method, you predict and plan everything from the beginning. Having this set schedule provides predictability, and the company and customer know from the beginning when everything will be done. Stability is also provided since everything is predicted and planned from the beginning; this gives the customer stability without the worry of designs and goals changing throughout the project. Finally, this model provides cost-savings since everything is predicted and planned out front, and if everything goes to plan, this model can save the customer money since the model will not change plans or goals throughout the project (Stephens, 2015, p.267-270).

C.2. DISADVANTAGES OF THE WATERFALL METHOD

Some of the disadvantages of the waterfall model include inflexibility, later initial release, and big design up front (BDUF). Since the waterfall method is a predictive model, everything is predicted from the beginning, and any changes, good or bad, can be hard to accommodate. The users will also receive a later initial release because the product will be released once the development is complete. Finally, BDUF requires you to plan out every detail upfront. This is difficult because it requires planning everything (Stephens, 2015, p.267-270).

C.3. ADVANTAGES OF SASHIMI

Sashimi is like the waterfall method. The main difference is that the phases overlap. This, of course, gives it one of its advantages: flexibility. You can go back and make necessary changes whenever you complete a step. For instance, you can backtrack and restart if the customer wants to change requirements, or a specific design will not work. Another advantage is that you can have an earlier release than the traditional waterfall model. What allows this is the overlap between all the phases. You can have people working in all stages simultaneously, allowing the product to be released sooner. Instead of going one phase at a time until completion, you can have people working on all of them at once. This model also increases productivity. Since you are not limited to one phase at a time, people such as developers can start working on code while the requirements and design are being worked on. This will also allow people to work more towards their strengths and so-called “get ahead of the game” (Stephens, 2015, p.272-273).

C.4. DISADVANTAGES OF SASHIMI

One of Sashimi's disadvantages is miscommunication. This comes from one of its advantages, the overlap in phases. For instance, developers can start developing code while requirements and design are still being defined. If there are changes in conditions and developers or other project staff are unaware of these changes, this can lead to delayed project release and miscommunication. Another disadvantage is goals for Sashimi can be too ambitious. The overlap in phases can lead to earlier release dates which can cause project managers to expect higher goals. Overambitious goals can lead to missed deadlines and excessive workloads. The final disadvantage of Sashimi is progress can be harder to track. With traditional waterfall methods, each step is completed fully before moving to the next, which makes tracking progress easy. With Sashimi, since there is an overlap between phases, everyone can work on all phases simultaneously. This can make monitoring progress complex and challenging to provide an anticipated release date (Aatmaj, 2022).

C.5. BEST SUITED

AVGC should use the waterfall model. AVGC's proposal is a smaller project well defined with their CRM requirements document. These types of projects are what the waterfall method is best used for. AVGC is a rapidly growing company that will need a well-defined and predictable roadmap to continue to develop and expand. The waterfall method provides predictability and stability with projects and will significantly benefit AVGC now and in the future as the company continues to grow.

D. DESIGN

As seen below, our proposed designs are for order management and the ticketing system. Regarding the ticketing system, we created a flowchart outlining how the system will work to obtain new contacts, store the information in the database, and leave an audit trail. With the order management system, we created a database diagram to outline how the proposed order management system will work. This outlines a proposed database we developed and how each required material will be stored in databases to provide a functioning ordering system.

D.1. TICKETING SYSTEM FLOWCHART

The flowchart below demonstrates how the proposed ticketing system will function, starting with determining if the interaction is unique and then breaking down the required information.

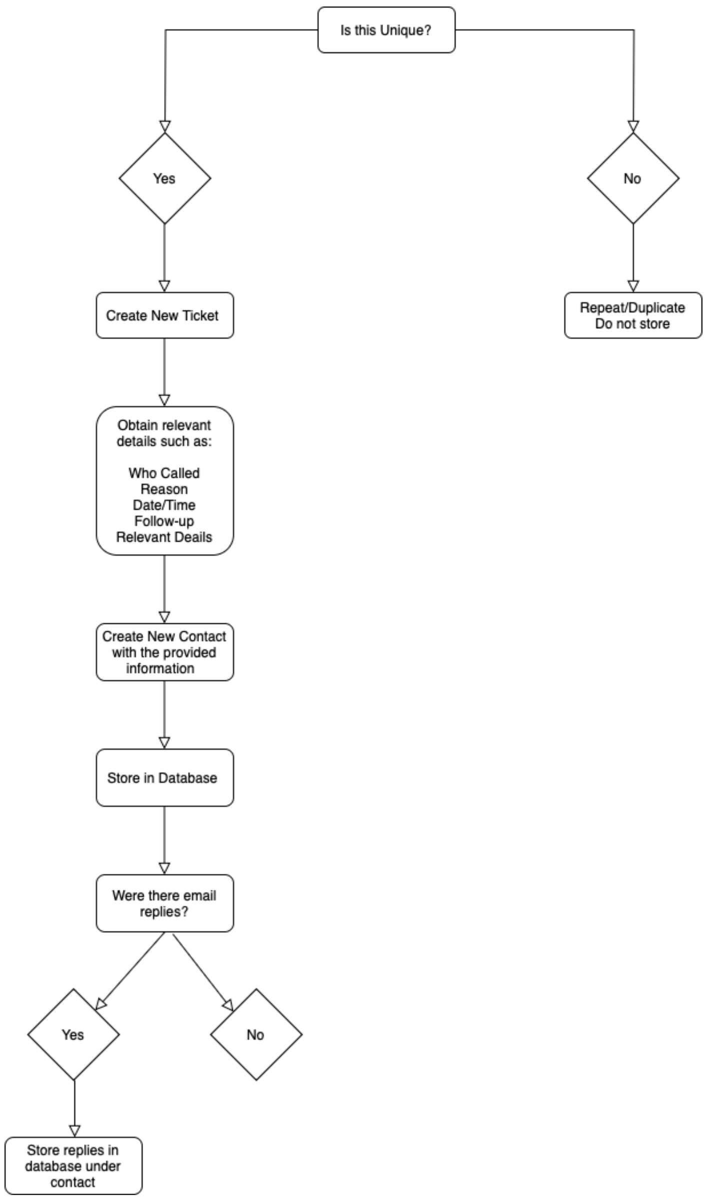
Is this Unique?
Yes
No
Create New Ticket
Repeat/Duplicate Do not store
Obtain relevant details such as:
Who Called
Reason
Date/Time
Follow-up
Relevant Deails
Create New Contact with the provided information
Store in Database
Were there email replies?
Yes
No
Store replies in database under contact

Figure 1: Ticketing System Flowchart

D.2. ORDER MANAGEMENT DATABASE DIAGRAM

The database diagram below is broken down into five tables: Customers, Orders, Parts, Shipments, and Quotes to Order. The lines connecting each database show entity relationships and primary and foreign keys.

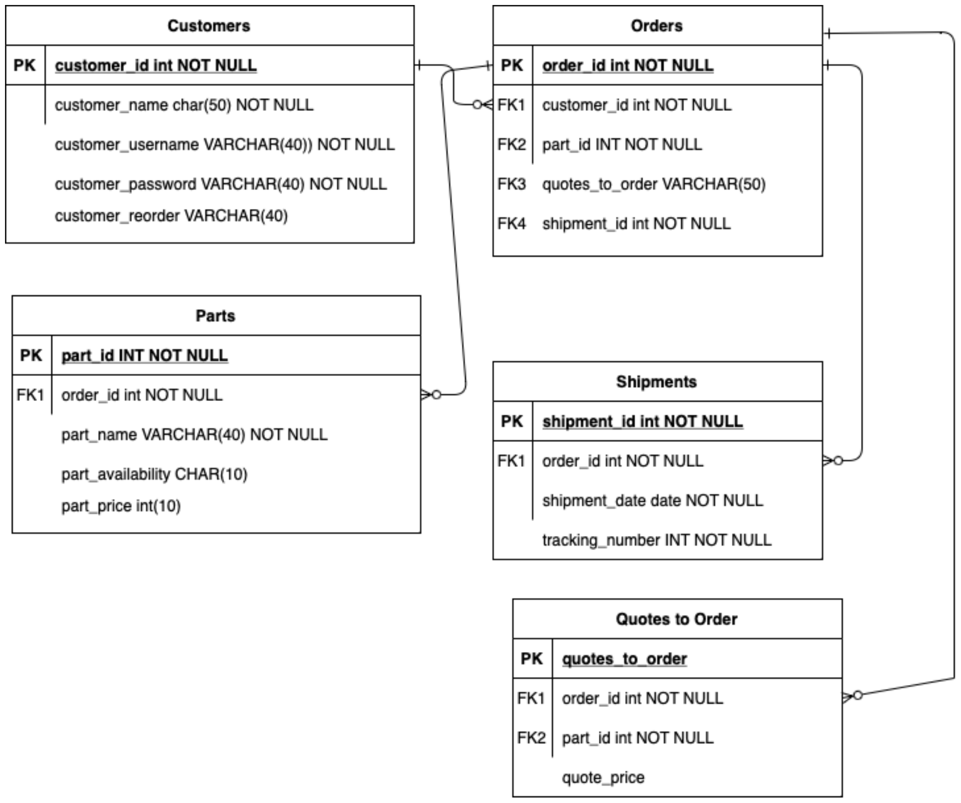
Customers
PK customer_id int NOT NULL
customer_name char(50) NOT NULL
customer_username VARCHAR(40)) NOT NULL
customer_password VARCHAR(40) NOT NULL
customer_reorder VARCHAR(40)
Orders
PK order_id int NOT NULL
FK1 customer_id int NOT NULL
FK2 part_id INT NOT NULL
FK3 quotes_to_order VARCHAR(50)
FK4 shipment_id int NOT NULL
Parts
PK part_id INT NOT NULL
FK1 order_id int NOT NULL
part_name VARCHAR(40) NOT NULL
part_availability CHAR(10)
part_price int(10)
Shipments
PK shipment_id int NOT NULL
FK1 order_id int NOT NULL
shipment_date date NOT NULL
tracking_number INT NOT NULL
Quotes to Order
PK quotes_to_order
FK1 order_id int NOT NULL
FK2 part_id int NOT NULL
quote_price

Figure 2: Order Management Database Diagram

E. TESTING

With our testing methods, we will address three requirements. These are support for older browsers, the ability to support 2000 users and 500 at any given time, and the order management system. With the support for older browsers, we will perform a compatibility test to see that our product is compatible with older browsers. Regarding the ability of our system to support 2000 users and 500 at any given time, we performed a performance test to show how our product will run with loads of customers. Finally, with the order management system, we performed a functional test to see if our database meets all the desired requirements.

E.1.1. COMPATIBILITY TEST

Requirement to be tested The system will run on all provided browsers: Latest Chrome and Chromium Latest Firefox I.E 9 and above Safari 13.0 and higher Mobile & tablet iOS10 Safari iOS10 Third Party Browsers (Chrome and Firefox) Android 7.0 Chrome
Preconditions: The system must be completed and ready for testing. Tester needs access to desired compatibility tester such as BrowserStack or Virtual Desktops. Tester needs to have knowledge/experience of how compatibility tester works and what to look for (Hamilton, 2023a).

Steps: 1. Determine which browser or browsers the tester wants to test for 2. Load up BrowserStack, Virtual Desktop, or other desired compatibility tester 3. Set up the environment to test the system in the desired browser/OS 4. Run the system and look for any bugs or design and layout glitches 5. Report all errors
Expected results: The expected result for this test is that with each browser and operating system, the compatibility test finds additional bugs specific to each browser and OS. This will allow us to improve the quality of our system to provide a better product.
Pass/Fail: The compatibility test above has passed. After running each compatibility test for the described browsers and operating systems, we received minor bugs to fix, design, and layout glitches specific to certain browsers and operating systems. We will fix some of these minor bugs and provide quality products.

E.1.2. PERFORMANCE TEST

Requirement to be tested To provide the ability to support 2000 users and 500 at one time during peak times and be scalable for more users in the future.
Preconditions: Have a mock testing environment that can imitate the concurrent users. Prepicked testing environment and testing tools.

Steps: 1. Start with 250 users 2. Check response time and CPU and memory usage 3. Increase to 500 users 4. Check response time and CPU and memory usage 5. Increase users until the system crashes to determine the absolute maximum number of users (This will also show the scalability the system has) 6. Fine Tune and Retest to keep the system below a desired response time and CPU and memory usage (Hamilton, 2023b)
Expected results: The expected result from this test is to see the response time for 250 and 500 users. These results will give us a response time of 4 to 5 seconds. This will also show us the CPU and memory usage with 250, 500, and maximum users when the system crashes. This will also show us where to refine and retune certain areas to improve response time.
Pass/Fail: The performance test passed with the different user loads. It provided us with a 4-second response time, and CPU and memory usage were low to where it would not overload the user's computer or mobile device. The system was able to function excellently with peak loads of 500 users. We could push the system to 2000 users simultaneously before the system crashed. This provides the company with scalability in the future as the company grows.

E.1.3. FUNCTIONAL TEST

Requirement to be tested The ticketing system will keep track of all communication and inquiry for contacts. It will track who called, the reason, the date/time, follow-up, and relevant details. These must be unique and capture all email replies.

Preconditions: The ticketing system must be developed
Steps: The steps the tester must execute to test the feature. 1. Send an email to the company with specific information such as who called, reason, date/time, follow-up, and relevant details. 2. Check the database and see if the system created a contact and stored the appropriate information 3. Send email replies to the company 4. Check the database to see if the information is stored 5. Resend the original details 6. Check to see if the database stored repeat information (This will test if the database is testing whether the information is unique or not)
Expected results: The expected results are that after an email is sent to the company, the ticketing system will pick up the email and separate the information appropriately into who, reason, date/time, follow-up, and relevant details. Once complete, each reply will be stored in the database, leaving an audit trail. If a repeat email or communication is trying to be sent, the ticketing system will not store repeats and will make sure the values are unique.
Pass/Fail: The functional test passed with the ticketing system. After emailing the company, the database stores all the relevant information correctly and in the right areas. It also captured all email replies from the customer and did not store repeat values keeping all the stored data unique.

F. SOURCES

Aatmaj. (2022, May 29). *Sashimi waterfall model of development*. DEV Community. https://dev.to/aatmaj/sashimi-waterfall-model-of-development-574o

Hamilton, T. (2023, March 25). *What is compatibility testing? forward & backward example*. Guru99. https://www.guru99.com/compatibility-testing.html

Hamilton, T. (2023b, May 2). *Performance testing tutorial – types (example)*. Guru99. https://www.guru99.com/performance-testing.html

Stephens, R. (2015). *Beginning software engineering* (1st ed.). John Wiley & Sons, Incorporated.