

Harvey Software Solutions CRM Proposal for AVGC

Title Page.....	1
Table of Contents.....	2
A. Introduction.....	3
A.1. Purpose Statement.....	3
A.2. Overview of the Problem.....	3
A.3. Goals and Objectives.....	3
A.4. Prerequisites.....	3
A.5. Scope.....	4
A.6. Environment.....	4
B. Requirements.....	5
B.1. Business Requirements.....	5
B.2. User Requirements.....	5
B.3. Functional Requirements.....	5
B.4. Non-Functional Requirements.....	5
C. Software Development Methodology.....	6
C.1. Advantages of the waterfall method.....	6
C.2. Disadvantages of the waterfall method.....	6
C.3. Advantages of Agile.....	6
C.4. Disadvantages of Agile.....	6
C.5. Best suited.....	7
D. Design.....	8
D.1. Flowchart.....	8
D.2. GUI.....	9-14
E. Testing.....	15
E.1 User Entry White Box Testing	15
E.2 Concurrent User Exhaustive Testing.....	16
E.3 Incremental Integration testing.....	17
F. Sources.....	17

A. Introduction

We at Harvey Software Solutions have read AVGC's proposal for a new CRM and believe we've come up with an affordable solution that can satisfy both parties comfortably. In this document you will find a list of requirements, a dive into our development methodology rational, an overview of our application design, and our quality testing results. We hope you remember HSS when it comes time to choose a bid

A.1 Purpose Statement

The purpose of this document is to elaborate on the design requirements and our proposed solutions, design methods, and testing for AVGC's new CRM solution.

A.2 Overview of the Problem

AVGC has seen an explosive 42% growth over the past two years and their system is aging and can no longer keep up with their employees nor users demand. Their system is disjointed, with different employees, in different workplaces, using different tools, and the splintering is beginning to show in the data. If they continue to grow without fixing this splintering, they're going to face issues with data concurrency or worse, data security. Every added link to the chain is another place the chain can break.

A.3 Goals and Objectives

Our CRM solution aims to solve AVGC's problems by being

- Scalable, allowing for maximum growth
- Secure, ensuring data privacy
- Centralized, smooth usage for clients, employees, and execs
- Intuitive, no training courses necessary
- Minimal, low overhead cost and no wasted processing

A.4 Prerequisites

Number	Prerequisite	Description	Completion Date
1	Database migration	AVGC has prior client and employee data that needs to be migrated to our servers, or otherwise allowed access to	May 2022
2	Web Server Loan	Due to our web hosting, payment is required to secure servers prior to launch so we have time to upload and test code	June 2022

A.5 Scope

These items are in scope:

- Taking orders
- Convert quotes to orders
- Reordering
- Part ordering
- Customer self-service
- soft delete (archive)
- hard delete

These items are out of scope:

- Opportunity Management
- Financial Forecasting
- Hosting (outsourced)

For more details, see Requirements Section

A.6 Environment

Our CRM proposal is web-based, and as such is designed to work on all modern browsers on their latest versions:

- Chrome; Chromium
- Firefox
- Edge
- Safari
- iOS Safari
- Android Chrome

Our tech stack is LAMP-base (Linux Apache MySQL PHP) and to enable our scaling, we're using Amazon Web Services as they're the most bandwidth per dollar

2. Requirements

AVGC has specified their need for a variety of systems and subsystems, spanning from content management to sales tracking. We believe that while some features may benefit from being outsourced (elaborated on in B.4 Nonfunctional Requirements), Order Management should be left in house, and we propose a multi-tiered system, one for both businesses and their clients:

B.1 Business Requirements:

If a Sales Team Member logs in to our order tracking application, they will be presented with a dashboard allowing them to

- Create a new order with a client
- Convert signed quotes into order requests
- Track the status of orders in progress
- Archive and remove from view completed orders
- Hard delete erroneous or duplicated entries

B.2 User Requirements

After an order has been placed with a Sales Team Member, a client is given their own login to our system. Upon login, a user is presented with

- The status of their current order
- An option to reorder completed orders
- A part order form for completed orders

B.3 Functional Requirements

Due to its cloud-based nature, our system should be able to handle more than the 500 predicted concurrent users, with headroom left for scalability in the future. We aim to design a web app, that way accessibility will be maximized, and only limited if the client is not using the latest browsers (e.g. Chrome, Firefox, Edge, Safari). Part of the reasoning behind making an Order Management system in-house is that it allows us to keep all user data on American servers.

B.4 Non-Functional Requirements

Opportunity Management: While we are able to store the data, we do not intend to analyze it in-house, at least not at current bid price. That functionality can be expanded into later should needs evolve.

Financial Forecasting: Our system does not utilize any neural networks to find overarching trends in data, at least not with the division hired. Can be discussed if prices change and we find contractors

3.C Software Development Methodology

Seeing as how AVGC has requested a waterfall design-based development cycle, we've taken the liberty of breaking down the pros and cons of that approach in case requirements change in the future. A waterfall approach involves using a linear, static, predictable, start-to-finish approach, the steps of which resemble a waterfall as they cascade into each other. Let's compare that to the popular dynamic, circuitous methodology, AGILE.

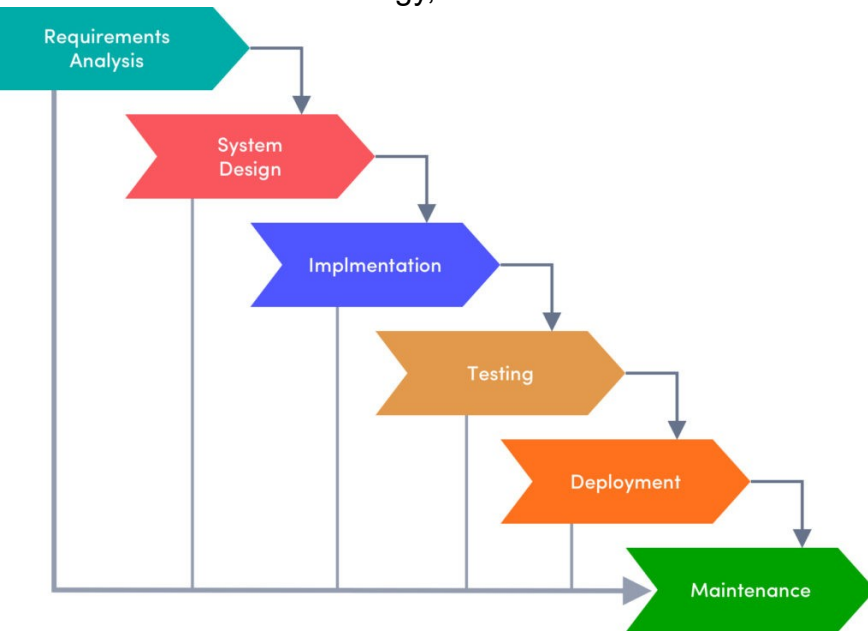


Image from blog.tara.ai [1]

C.1 Waterfall Pros:

Due to the need for one concrete deployment date, the waterfall method lends itself to projects that require a predictable launch date/window. Consistency is ensured throughout development by setting up milestones one time only, and checking in to those milestones along the way. Systems designed this way are checked and approved before they deploy, so that the end user rarely has a rough experience.

C. 2Waterfall Cons:

The waterfall system has no user feedback built into its flow, so aside from upkeep, maintenance, and small changes over time, the end product is the end product. If a lot of new information is coming in while the product is being developed, waterfall method will struggle to incorporate that new information without starting over

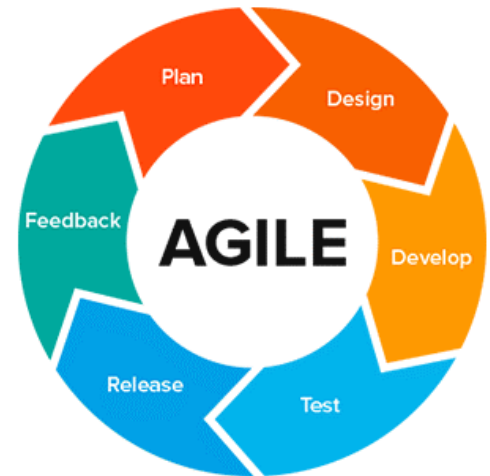


Image from timhunold.medium.com [2]

C.3 AGILE Pros:

When the goal is more of a service than a singular, static project, one would want a service that can update and change and evolve overtime. If growth becomes so advanced that a system needs to adapt weekly or sometimes even daily, then the process can no longer resemble a waterfall with an end, but must become a never-ending whirlpool of updates, feedback, and more updates. AGILE shines when flexibility and high user feedback are valued

C.4 AGILE Cons:

Requires constant energy put back into the system by the way of a fully staffed dev team, a never-ending production cycle, and a reliance on user feedback and product updates. If nothing substantial is going to change about how users use the service, why bother aiming for often updates, or anything beyond bug patches and small maintenance?

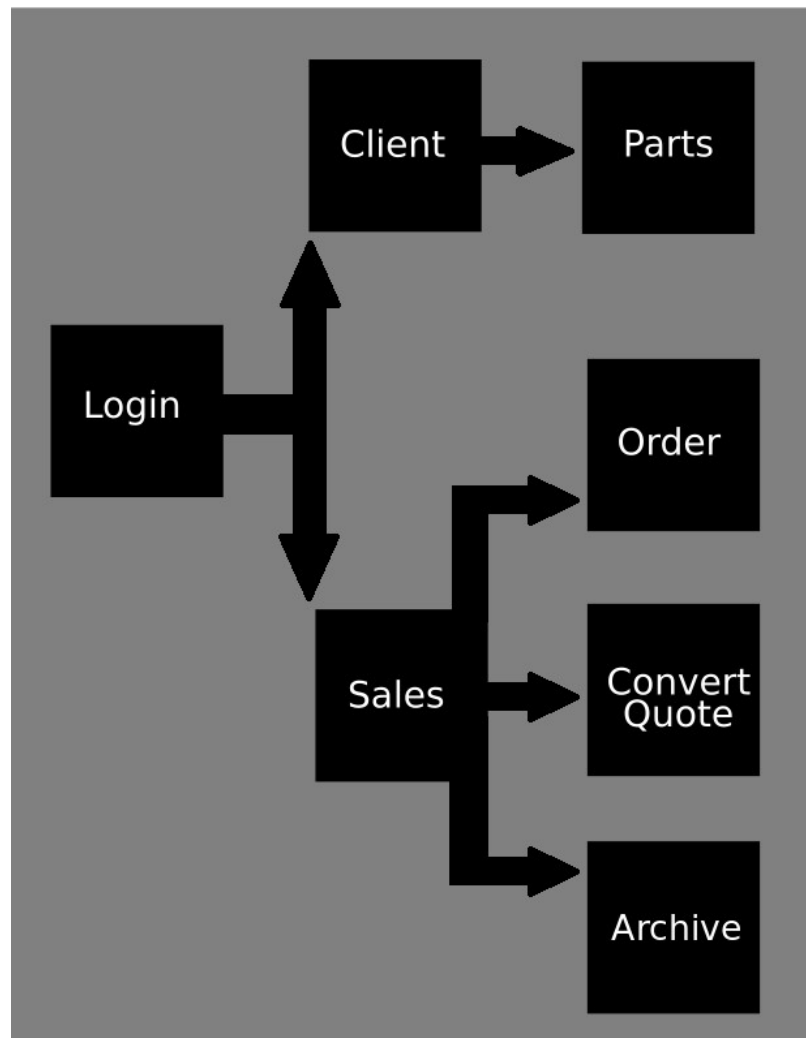
C.5 Best Suited – Closing Thoughts

Waterfall method is a movie, you have to wait till the sequel to get the next one, but it's way higher quality than anything else you can watch. AGILE method is a TV show, a new episode comes out every week because people tune in expecting something new every week. It's lower quality and therefore cheaper short term than a movie, but more expensive as it goes on.

The CRM document provides a thorough guideline for what this product is supposed to be. And although the company is growing fast, there shouldn't be too much reason to update the system beyond maintenance. AVGC already uses waterfall method internally, and I think this solidifies our decision to stick with it for this project.

D. Design

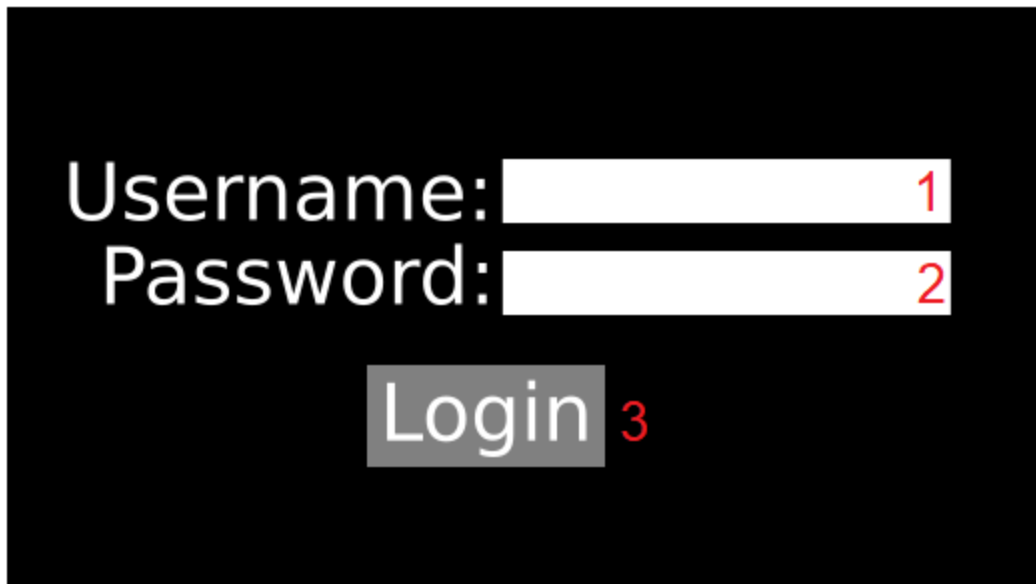
D.1 Flow Chart



As mentioned in the Requirements section, our CRM splits Order Management into two categories, the client-side tools, and the sale-side tools. From the client dashboard, the user can see the status of current orders, resubscribe to old orders, or navigate to the parts page for an old order to request new parts. From the sales/company-side, a view of all their current clients orders can be seen, with links to a form make a new order or convert a signed quote into an order. By navigating to the archive, the user can see all order hidden from the main view, and even delete them if necessary.

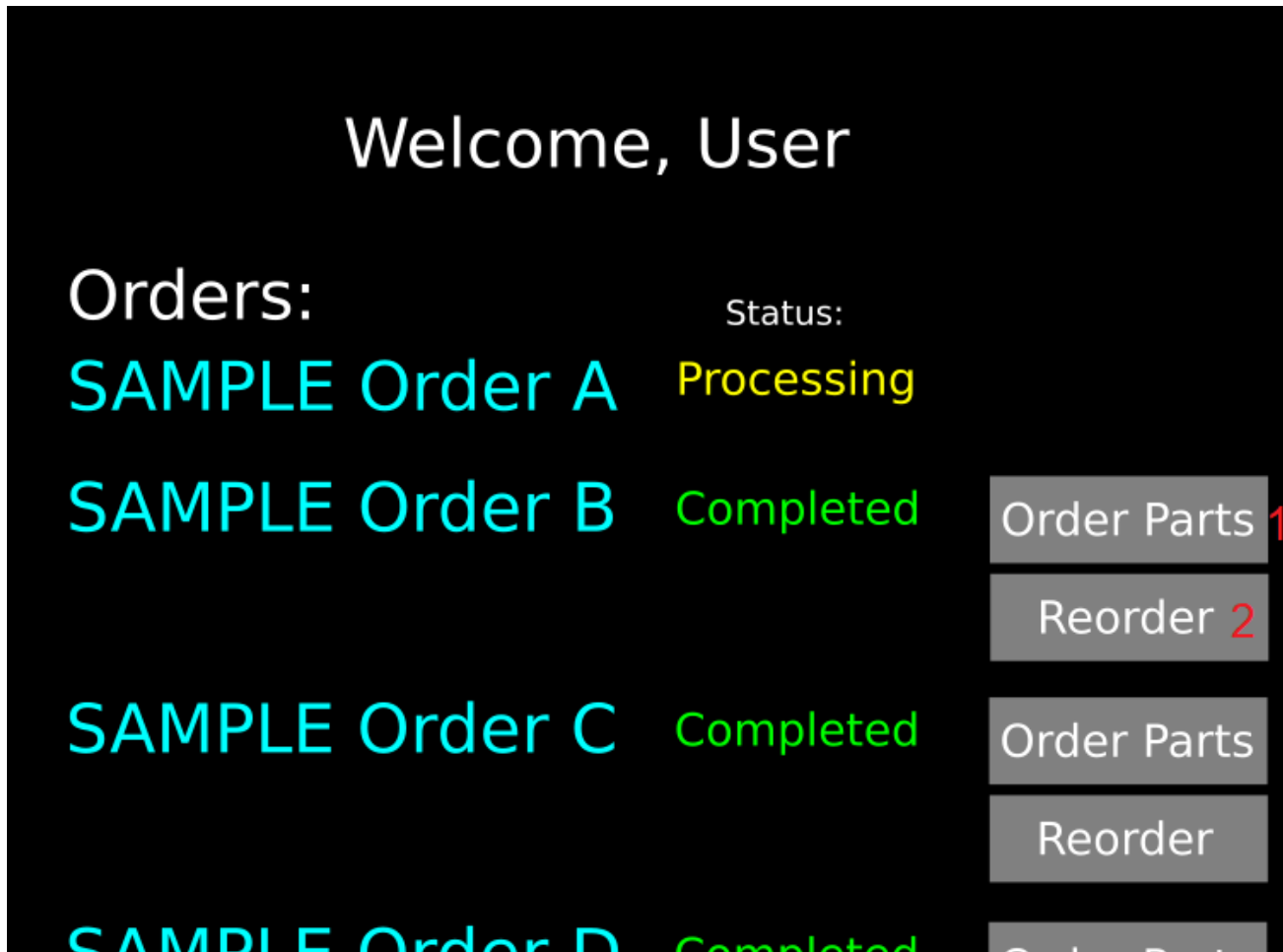
D.2 GUI

1 of 7: Login

A screenshot of a login form on a black background. It features two white text input fields. The first field is preceded by the label 'Username:' and has a red number '1' at its right end. The second field is preceded by the label 'Password:' and has a red number '2' at its right end. Below these fields is a grey rectangular button with the word 'Login' in white text, followed by a red number '3'.

Clients or Employees can login and are redirected appropriately

ID	Control	Property	Data Source
1	text box	Receive username	User text input
2	text box	Receive password	User text input
3	button	On click: Submit data, redirect to client or sales	Form - text boxes



Clients are shown their current and past orders, with option buttons to order parts or reorder

ID	Control	Property	Data Source
1	button	Link to the parts page for that order	Internal Variable
2	button	Re-submit order details as new order, confirm with pop-up	Internal Variables

Back 3

SAMPLE Order B

Parts:

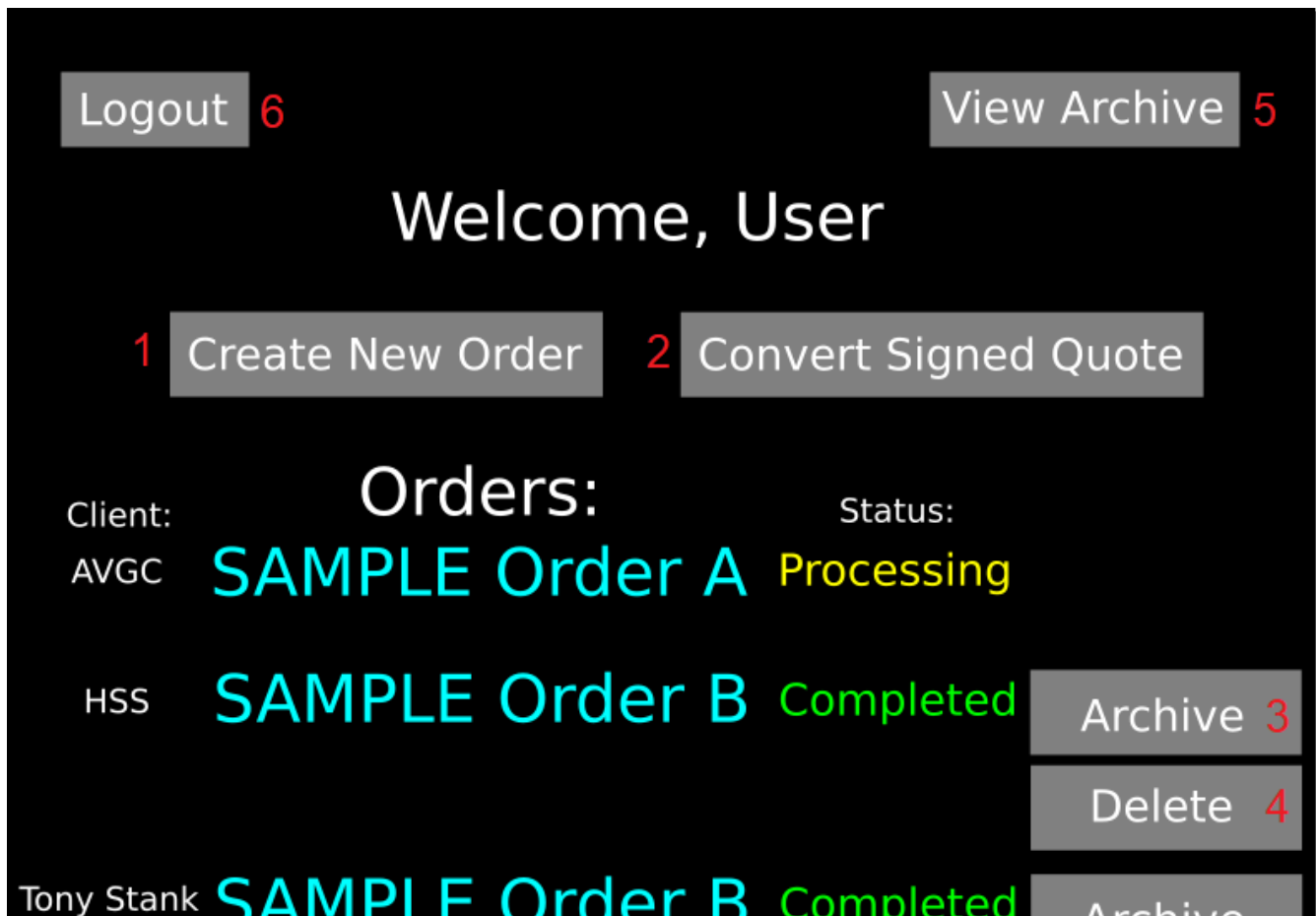
SAMPLE PART 1 Quantity: 1000x 1 Order Parts 2

SAMPLE PART 2 Order Parts

SAMPLE PART 3 Order Parts

Client can order individual parts from any past order

ID	Control	Property	Data Source
1	text box	Receives user inputted number (int)	User text input
2	button	On click: Popup confirm; If yes: submit data	Form - text box
3	button	On click: Return to client orders dashboard	N/A



If the user logging in is a Sales Representative, they are redirected to their own dashboard

ID	Control	Property	Data Source
1	button	Links to order form	N/A
2	button	Links to quote convert page	N/A
3	button	On click: Add order to archive, hide from view (soft-delete)	Internal Variable
4	Button	On click: Popup confirm, if yes (hard) delete order	Internal Variable
5	Button	Links to archive of soft-deleted orders	N/A
6	Button	On click: Popup confirm, if yes return to login	N/A

5of7: Order Form

Back 9

Label: SAMPLE Order A 1

Client: 2

Parts:

3 SAMPLE PART 1

1000x 4

\$. \$\$ 5

\$. \$\$ 6

SAMPLE PART 2

1000x

\$. \$\$

\$. \$\$

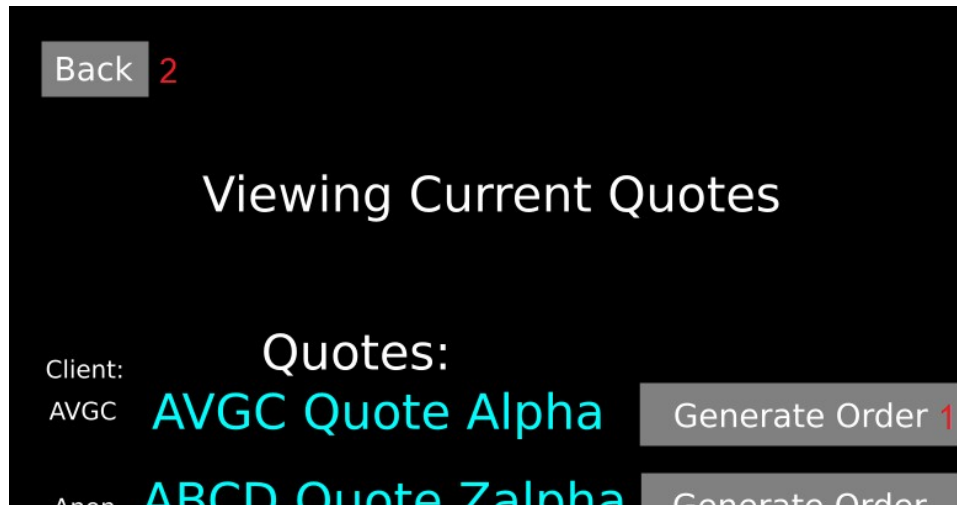
Total: \$. \$\$ 7

Place Order 8

A Sales Rep can help a user create a new order with this form

ID	Control	Property	Data Source
1	Text box	Accepts user text for order label	User input
2	Text box	Accepts user text for client	User input
3	Text box	Accepts user text for part of order	User input
4	Text box	Accepts user number for quantity	User input
5	Number	Number reflects current unit cost for selected part	Internal Variable
6	Number	Updated live by multiplying quantity and cost for part	Calculated Variable
7	Number	Updated live by adding all costs in form	Calculated Variable
8	Button	On click: Popup confirm: if yes, submit form data as order	Form
9	Button	On click: Popup confirm, if yes return to sales dashboard	N/A

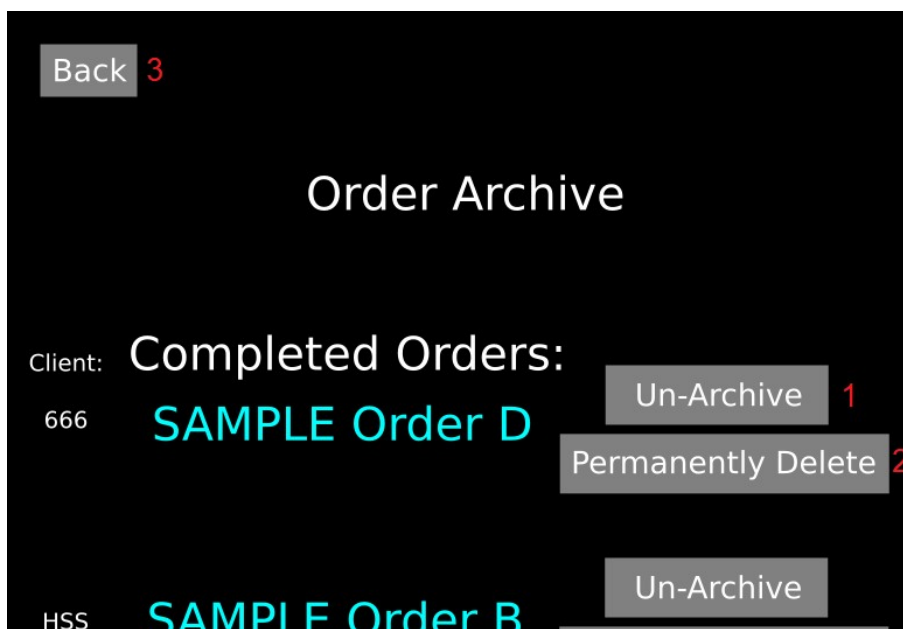
6of7: Quote Convert



Quotes are handled by a separate database, but if signed, appear here, ready for conversion

ID	Control	Property	Data Source
1	Button	On click: Links to order form and pre-fills the inputs with information from the relevant quote	Internal variables
2	button	On click: Popup confirm, if yes return to sales dashboard	N/A

7of7: Archive



Here a Sales Rep can modify, view, and drop their hidden orders

ID	Control	Property	Data Source
1	Button	On click: Undo soft-delete, add order back to view on dash	Internal variables
2	Button	On click: Hard-delete order, remove from archive	Internal variables
3	Button	On click: Popup confirm, if yes return to sales dashboard	N/A

E. Testing [3]

E.1 User Entry White Box Testing

While users are free to enter numerous values in the part quantity field on our CRM's order form, only whole number values greater than zero and less than the available count of the part should be accepted. As the software was developed in-house, we had access to the internal code, letting us test with values we specifically knew would challenge our system (negatives, fractions, mega huge values, zero, null values), a method known in-industry as white box testing

E.1.1 Entering an IN-Valid Part Quantity

Requirement to be tested:

The order form's ability to handle bad data entered in parts quantity field

Preconditions:

A valid label, client, and part all must be entered, so the quantity field can be isolated as the sole test variable and left blank before testing

Steps:

- 1-Determine an invalid yet typable number entry (0, $\frac{1}{2}$, 10^{100})
- 2-Enter the value into the quantity field on the order form
- 3-Click Submit
- 4-Click Confirm on popup

Expected Results:

The data should fail to post to the database, and the user should see a red highlight around the invalid data field, with a popup dictating the rules of the data field to the user so they know what and where to correct.

Pass/Fail?:

Test **passed**. Our system rejects any data that could cause an erroneous calculation before any damage happens to our database or to our clients wallets.

E.2 Concurrent User Exhaustive Testing

AVGC has clearly defined a business need to handle 500+ concurrent users on their system at once, as well as breathing room to make sure they're comfortable as they continue to scale. With that in mind, we knew we needed to test every possible port and switch on our system to make sure they were up to snuff. When dealing with every combination of IP address and user activity, this brute force-style method is known as exhaustive testing

E.2.1 Testing 500 Bots on Ports

Requirement to be tested:

The cloud server's ability to handle the strain of as many concurrent users as possible.

Preconditions:

As many scripts/bots as there are users to simulate, plus hardware to run off of

Steps:

- 1-Turn server live
- 2-Set bots/scripts to connect randomly, to random ports
- 3-Continue to add more bots/scripts as time goes on, tracking up-time
- 4-When server fails/Testing resources run out, record collect data

Expected Results:

The servers are cloud-hosted so as user demand grows, so too does our processing power. We've budgeted enough to maintain load well over 500 concurrent users at any activity level and port connection.

Pass/Fail?:

Test failed, after 750 concurrent users. Meaning a huge **pass** for our CRM and 50% headroom for growth over time until more resources are allocated to expansion.

E.3 Application-Database Interaction: Incremental Integration Testing

Since we developed this application as a dual-user experience, we naturally took consideration in ensuring the cross-availability of the order data at all times. E.g. If a client modified an order, a their company representative can see that change on their dashboard in real-time. Verifying interactions between modules is where incremental integration testing comes into play

E.3.1 Order Cross-Availability

Requirement to be tested:

The application's ability to properly add values to the database and display those values accurately to the appropriate users.

Preconditions:

A Sales/Company Representative needs to have an account linked with at least one active client account and one prior completed order.

Steps:

- 1-Log in as client
- 2-Reorder any completed entry
- 3-Logout
- 4-Login as Sales
- 5-View and record changes

Expected Results:

When a linked client places an order, that order should be viewable in the database, and that change should be reflected in the sales dashboard if the two accounts are linked

Pass/Fail?:

Test Pass; We had our engineers test proper data integration at every step of development. This along with the prior test results shows a favorable time to deliver.

F. Sources

[1] *Waterfall*. (2020). Tara. Retrieved April 12, 2022, from <https://blog.tara.ai/software-development-life-cycle/>.

[2] *Agile*. (2021, December 9). [Agile development cycle]. That Tim Guy. <https://timhunold.medium.com/orbital-development-cycle-8ec215da6bc0>

[3] Stephens, R. (2015). *Beginning Software Engineering*, Chapter 8 - Testing. Wrox Press.