Doll Distributing Ecommerce System

PROJECT PLAN

Team Number: SDMAY19-22
Client: Doll Distributing
Adviser: Mohamed Selim

Alec Harrison - DevOps
Caleb Olson - UI/Database
Daniel O'Neill - Testing Engineer
Mitchell Bennett - Communications
Thomas Staudt - UI/Testing
Thomas Wesolowski (TJ) - Project Manager

Team Email: sdmay19-22@iastate.edu
Team Website: https://sdmay19-22.sd.ece.iastate.edu/
GitLab: https://git.ece.iastate.edu/sd/sdmay19-22
Revised: 10/26/2018 Version 2

Table of Contents

List of Figures

List of Tables

List of Definitions

VIP: Vermont Information Processing

API: Application Programming Interface

CSV: Comma Separated Values

AWS: Amazon Web Services

DB: Database

SQL: Structured Query Language

SES: Simple Email Service

App: Application

Web: Website

UI: User Interface

Agile: Development methodology focused on incremental implementation of small tasks over short periods of time.

PHP: PHP Hypertext Preprocessor

HTML: Hypertext Markup Language

CSS: Cascading Style Sheets

IT: Information Technology

BrandFinder: A website that shows how individual products are branded

1 Introductory Material

1.1 Acknowledgement

Team 22's Client: Doll Distributing LLC

Team 22's Advisor: Mohamed Selim

1.2 Problem Statement

Currently Doll Distributing has an ordering system that requires sales reps to be in person to make sales. This can cause their clients to feel pressured into buying other products or being inconvenienced to place orders at not optional times. There is also not a way for clients to learn about new products or deals on their own time. They currently must call Doll's rep and hear about deals if their schedules work out.

Our task is to create an ecommerce system for Doll Distributing that will interact with their current inventory system. We will create a system that will allow clients to view Doll's inventory, place orders that will be sent to account reps, and be able to view their historical orders. This will be linked from their current site and not allow anyone without an authorized account with a valid liquor license to enter.

1.3 Operating Environment

The operating environment of our project will be an AWS Linux server hosting a docker container running a Laravel [8] web app. This application will also have to have at least a mySQL database with a user's table to host users and their liquor licenses. This system should be able to run in any environment a smartphone can connect to the internet and a climate it can run.

Order Placing- Launching a php job to send an email via SES to account reps, associating users and orders in DB so users can see past orders.

User Interface – Linux Operating System, Route 53 route to hit to bring up web app, User Browser (like chrome, edge, or firefox)

1.4 Intended Users and Intended Uses

Our project with Doll Distributing is a unique system that required us to do some research to come up with users for our app. This app will be designed with our core users in mind and also work well with our other users, too. We have narrowed down our users to clients, account reps, IT, and sales.

This project has four main users with varying levels of needs that need to be addressed by our app. For the IT users we must make an app that is maintainable and works with their current inventory system as real time as possible. Another role is the clients, or the people who buy the alcohol from Doll. This user is our primary use case and we must satisfy their bar minimum needs of placing an order and seeing their past orders. Next, we identified the account rep user. They would need to be able to authorize users to be active once their

liquor licenses are validated. The last user we identified would be the sales people. This system needs to be visually appealing enough that the sale department should be able to pitch this system as enough of a pro to bring clients to Doll from their competitors.

1.5 Assumptions and Limitations

Assumptions:

- We will be able to interact with VIP's inventory system better and quicker than the 20-minute manual CSV export.
- We will be able to convince VIP to give us access to an api or a testing database.
- The end product will be used in the United States.
- The website will be written in English.
- We will be able to achieve our goal using a MySQL DB.
- We will be able to extend Doll's current website style to our new system.
- An email to an account representative when an order is placed will be the end goal of our product.
- Our app will be maintained in the future (so it requires clean and well tested code).

Limitations:

- The user shall be validated before they enter the site (client requirement).
- The app shall use VIP's inventory data (client requirement).
- The app shall work on desktop and popular smartphones.
- Currently, the fastest time that a CSV exports from VIP is twenty minutes.
- The website will not allow people without liquor licenses to enter.
- The site will be dependent on Amazon Web Services.

1.6 Expected End Product and Other Deliverables

This section will go over our deliverables and the times at which we expect them to be available. This project will be divided into several major milestones. Namely the User Interface, a testing platform for a pilot group, and the final Ecommerce platform in Live Production. Documentation will be included with each of these deliverables to help Doll as well as the users be well informed about usage of each of the product's phases.

- User Interface for clients
    - The interface will display the information from the inventory file in a helpful and efficient way, by using blocks for every product.
        - Display the important details to catch the user's eye, clicking into it will display more information about the product
    - Interface with current online website provided by Doll, and match its style.
- Testing Platform for a pilot group
    - Users will be able to log in using a registered account and have a valid liquor license.
    - Users will be able to register an account with our website and send in information to the representatives to allow users to be able to authenticate to our web services

    - Users will be able to see the User Interface provided by the above section.
    - Users will be able to add items to cart and be able to check final details for the order and pricing before the user sends the confirmation order to the Doll representatives.
    - Choose a group of clients to be test users for the pilot tester and get a list of suggestions and criticisms to implement into the site.
- Final Platform in Live Production
    - The final product will be a refined version of the original prototype with documentation.
    - We will take the prior list and refine our website to meet user's demands and make the platform more pleasing for the vendors using it.
    - Put the project online and available for all the users to be able to order from.
    - The platform will be used by Doll Distributing, the repersitives in the field, and Doll's clients.

2 Proposed Approach and Statement of Work

2.1 Objective of the Task

The ending goal of this project will be as follows:

– An actual production online ordering platform

– Log in and out of the website

– Certain views based on what kind of commercial buyer you are

– View recommended beverages and past orders

– View available beverages

– Select beverage items to checkout

– View currently checked out beverages(personal cart)

– Place an order on the website

– Sending email to the account representatives who will validate and place the order internally in existing order placing system

– Updating the inventory database (Live time or hourly CSV if that is what they decide).

2.2 Functional Requirements

The functional requirements for the project are the following.

- The user shall be able to log in using their registered account and liquor license
- The user shall be able to view all available products
- The user shall be able to select products to order
- The user shall be able to place an order
- Order details shall be sent to Doll Distributing to be approved
- Database shall be appropriately updated at least once a day

2.3.1 Constraints Considerations

Non-functional requirements:

- Database must be updatable within a manageable amount of time
- Website must be properly branded
- Products must be displayed in a visually appealing manner
- Ordering Platform matches the regular webpage

Constraints:

- VIP is not allowing unfettered access to their database
- Funding for hosting services such as an AWS server

- Time to learn new technology stacks

2.3.2 Standards

This project will be implementing design standards from Laravel and Bootstrap in order to create a clean user interface. These standards can be found on each technology's documentation page. These standards are largely there to have our files match the standard formatting that other people will be expecting. We plan on following these file standards as this will make it much easier for each member of the group to contribute to the project.

2.4 Previous Work And Literature

The domain that our project seeks to inhabit is more specific than a general online store. While a typical online store akin to Amazon, or the online stores of retailers like Walmart and Target focuses on providing a wide range of products directly to consumers, our application is focused on a smaller set of products, and is restricted to existing Doll retail customers. Examining our project's domain reveals two main applications currently providing this functionality; Budweiser's online ordering website, TapWeiser [15], and VIP's [14] own online ordering tool, VTInfo.

TapWeiser offers a slick user interface, and a cohesive and orderly design. It simplifies the user experience by allowing for the reordering of past orders and frequently ordered products. It has a significant downside however, as it is available only for use with Budweiser products; and is thus problematic for Doll as a distributor working with several different brewing companies. Its use is also intended for retailers to work directly with Anheuser-Busch; which would effectively assume the distributor role that Doll provides as its core business. However, we have noted that the user-friendly UI is this application's main claim to success; we have made this a key point of focus for our project.

VTInfo is provided by VIP. As a result, it integrates natively with Doll's current inventory system. This allows the software to provide up to date inventory information in real-time. However, Doll has expressed that they as a company, as well as their retail customers, find the interface to be cluttered and difficult to navigate for those not familiar with VIP's other products. Doll is concerned with the amount of time that it would take to train not only their retail customers in the use of the software, but their own employees as well. They are also concerned with the level of support that the software receives; based on their previous experiences with their existing inventory system. Our project seeks to provide a similar level of integration, while providing an end-product that is more robust.

2.5 Proposed Design

Our currently proposed design is to build a Laravel web app that can be run within a docker container on any platform. An AWS Linux Server will be used to host the container. Customers will be able to view the web app in browser. While viewing the page, they will be able to place orders through selecting items on the webpage. Orders will be placed via PHP jobs when a customer finalized their order and notifies the web app that they wish to place it. We have two options for getting the inventory and pricing data: one way is to use CSV files from VIP, and another is to use XML files. The CSV files take 20 minutes to process after being parsed. The XML files would not require as much

processing time after parsing, but are much bigger and contain a lot of data that is not needed for our purposes. XML files would likely take a significant amount of time to parse due to their size. That data will then be used to populate the ecommerce site and allow validated users to view the items and select them for orders. User logins will be supported by a database that is also hosted on AWS. The database will store user details such as login information, liquor license information, and previous orders to allow for items to be suggested automatically to repeat users.

Our current system design has several strengths that will allow it to perform the necessary requirements for placing an online order. Most notably, our system will be containerized using a Docker Container. This allows it to be easily picked up and transferred to a new host if it ever needs to be redeployed. Having dependencies taken care of by Docker will allow development to be focused on functionality rather than meticulous management of the system whenever it must be redeployed.

The architecture of the system, shown below, is a good example of how streamlined and simple our system can be, while still containing all intended features. The initial data population is very linear. Currently, we are planning to use the CSV report, but are thinking about switching to parsing an XML report instead. In the current design, the CSV or XML routes will be interchangeable, and the project will proceed as planned regardless of which one we end up deciding to utilize. The CSV sheet will be retrieved from VIP, parsed and then sent to the website to populate available inventory and prices of available items.

The website will utilize Vue.js to turn the available inventory into a list of cards that show product details to the customer. It will be in a list format to easily accommodate for changing numbers of products over the course of each season. Ordering will then be a simple process for the customer. A strong area of this design is that the customer will be able to quickly validate themselves with just their liquor license and store information, and then quickly go to ordering products. Speed is very important in this case, since customers working at bars will often close very late at night, and want to order new inventory quickly as they close up. Our system will E-mail final orders to a Doll sales representative. This means the system does not have to validate all payment information on the spot, and further expedites the process. Overall, our design capitalizes on linearity and ease/speed of ordering for the customer. Those traits heavily factored into our design choices, and should continue to support our decisions as users start testing the product.

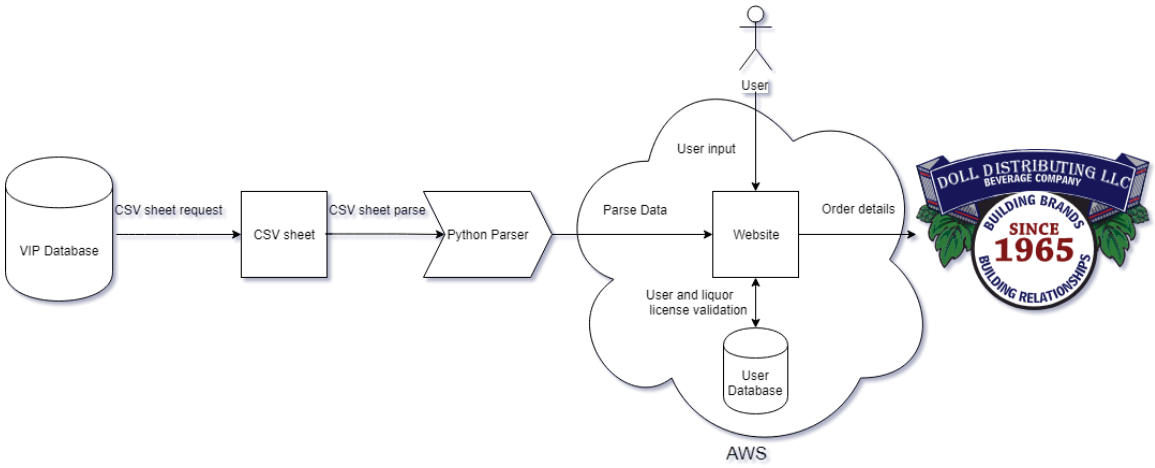

Figure I: Proposed Architecture Diagram

There are several notable design alternatives that we have not decided to pursue. One option was to replace Laravel [8] with Django [3]. This would allow us to capitalize on several of our group member's previous Python language experience. Its drawback is that we also have several group members who have worked with Laravel before, so picking between the two will always benefit a few specific people, and require others to learn a new framework. Other design alternatives involve our methods of retrieving product data. Our options are to use reports that are generated in a CSV format, or XML reports, or to use an API to interface directly with the VIP database. Using an API to retrieve and update data in real time seems like the best method, but it cannot be used because VIP will not allow us to have direct access to the database.

This left us with options for CSV and XML file parsing. For either of those options a python script is necessary. Both versions are very similar, resulting in a very difficult decision. Currently the CSV format is the route we are taking, but it may be subject to change as we get further in development of the website. We will have to switch to XML parsing if we cannot get pricing information from the CSV reports.

2.6 Technology Considerations

Laravel [8] is a very powerful web application framework. It is definitely capable of doing everything we need for this platform. It allows us to easily manage all resources in regards to CSS and Images, and can also be placed in a Docker Container [4] through the use of Laradock. However it is fairly complex for what we are building, and few group members have extensive experience with it. Over the course of the project this issue will slowly become smaller as each member improves their skills with the framework.

AWS will work fantastically for our hosting service. It allows us to keep the application very scalable while also keeping hosting prices low. It also means that Doll does not need to worry about their own uptime or hosting for the application.

VIP is our primary database. It will be difficult to utilize because we will not have full access to it. Unless we obtain an API for accessing it, it will likely be the biggest roadblock in development. This is because the alternative is to manually export and upload CSV files for inventory.

2.7 Safety Considerations

There are not any pressing safety concerns. Most concerns about the project relate to risks about data or validation of liquor licenses for users.

2.8 Task Approach

We will be using Agile methodology to develop this project. Larger tasks and deliverables will be broken down into smaller issues that can be taken on one at a time by individual team members. Each sprint will have scheduled tasks that are targeted for completion by the end of the sprint.

We will be developing a mock UI as our first deliverable for the client. This will allow us to get confirmation on what we are building before we get too far in the development process. Once we get a UI approval we will continue developing the UI and confirming out loading time is where it needs to be. Once the UI is confirmed we will build the data bridge. We are hoping to have the mock up done and delivered by the end of November.

Our application's inventory will be updated at least once a day to mirror Doll's current inventory updates. Our program will take in a .CSV or XML file at least once a day, parse it, put it in our MySQL database, and inventory information will translate to the application. Our application will load and respond in under three seconds. When we achieve this goal we plan on beta testing the service with some of Doll's clients. The demo will allow us to gather feedback from real users, to improve the system.

Our second deliverable will be an automated fetcher of the .CSV or XML data grabber. That way we will no longer have to update the system manually once a day. Due to the exports taking around 20 minutes we are planning on having our system update as fast as the average export takes. This will allow our website to have a more accurate count on inventory. We are planning on using a python script to automate this process.

2.9 Possible Risks And Risk Management

Any orders must be through a valid liquor license, since there are many regulations relating to liquor distribution. This risk can be managed by very carefully testing our methods for validating liquor licenses on our ordering platform.

2.10 Project Proposed Milestones and Evaluation Criteria

Milestone 1: Determine requirements

Milestone 2: Develop a UI mockup.

Milestone 3: Develop a testable prototype.

Milestone 4: Refine the prototype based on trials with the client's users.

2.11 Project Tracking Procedures

We will be using the issues page on our team's GitLab page. This way we can set up our milestones and place issues under those milestones. This also allows for members to take responsibility of these tasks and due dates to be placed on those issues.

2.12 Expected Results and Validation

Our desired outcome is to have a functional ordering system that can be used by Doll Distributing and their clients.

At a high level users are able to place orders, view old ordered products, recommendations, and be able to send an email verification to the approvers to validate the orders before sending them to the packagers.

The set of validation criteria that will need to be fulfilled will include the following:

1. Users should be able to log in using their liquor license and store information
2. Products will be viewable by validated customers
3. Products listed on the site will match what is given in a CSV or XML report
4. Validated customers will be able to select available items and add them to their virtual shopping cart
5. Customers will be able to modify items in their virtual shopping cart. This includes changing quantity, or removing items from the cart.
6. A properly formatted Email with order details for a customer will be sent to the appropriate Doll sales representative when the customer decides to finalize an order.
7. Returning customers will be able to correctly view their purchasing history.

2.13 Test Plan

We will be using Laravel Dusk [8] to test our platform to make sure that we have every possibility tested and not able to access things we are not supposed to. Also we will be testing as we are coding, and writing test cases before we start, so there should be a clear instruction set. Also after we are done with each section, we will be trying to PenTest/test out all of the functions of the platform to make sure everyone gets their exact permissions.

Our testing will be conducted in multiple ways. We will first run all of our automated test suites as we develop on our local machines. This will confirm that our new features don't break old tests, and also force us to keep up code quality. We will also run tests automatically on its way to production. The hope with running tests before production is that the test environment will mirror production to have a last confirmation before pushing code to production.

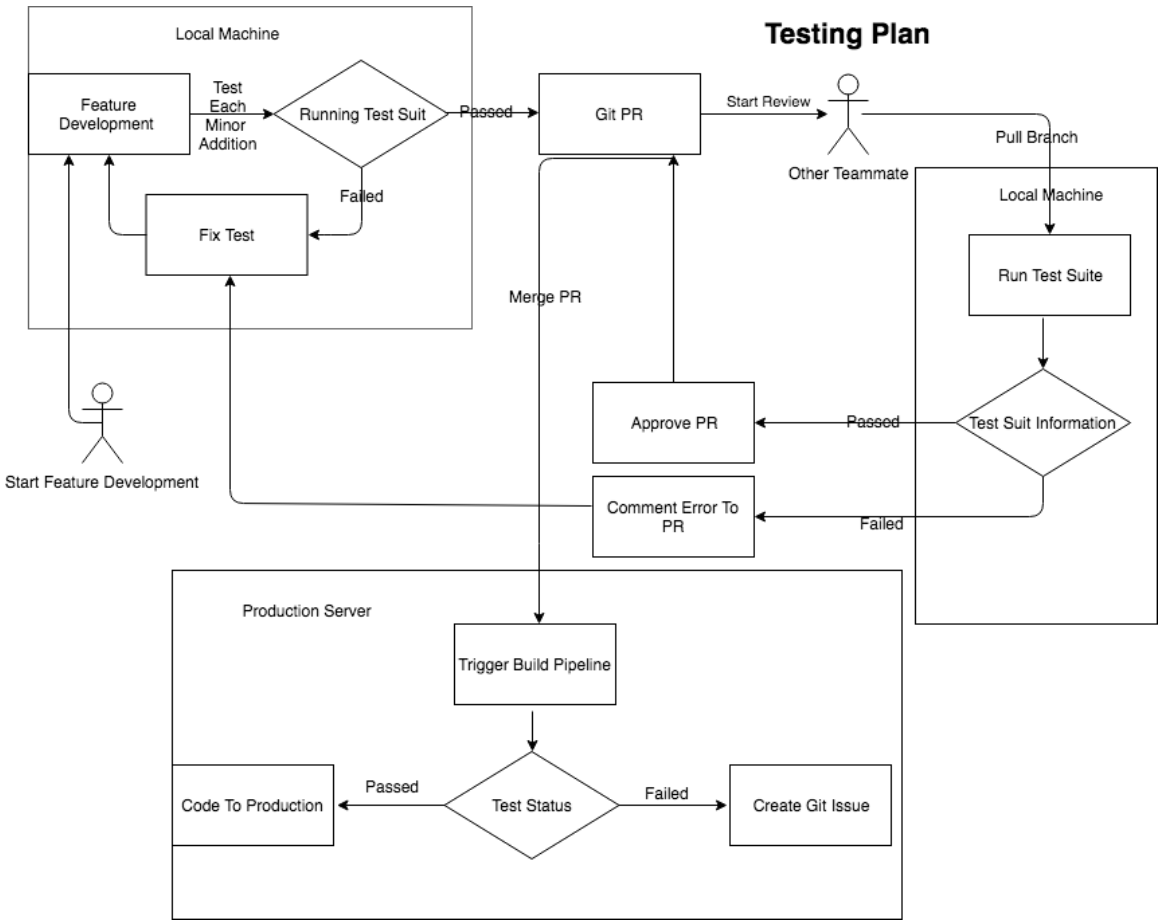

Figure II: Testing Plan

3 Project Timeline, Estimated Resources, and Challenges

3.1 Project Timeline

For further reference, see Gantt charts in appendix 4.3

Functional	Non-Functional	Weeks
Stage 1		Total: 5
Retailer ordering portal (General Base Requirement)	• Access should be through a convenient web-based portal • Access must be restricted to only registered retailers • Retailers must not be able to access information from other retailers • Web portal must integrate with current client website, using already established links on client site.	N/A
Retailer information/ orders should be stored in a database	• Database should be of some standardized format/structure • Database must be private and secured such that only client is able to access	2
Retailers should be able to view products	• Products should be shown with a status indicating availability • Products shown should include description of the product, or link to the client's BrandFinder page	2
Retailers should be able to place orders	• Retailers should be able to submit orders through the online web-based portal • Ordering screen should display quantities of each product • Ordering screen should display itemized and total cost	4
Orders should be sent to client for review	• Orders should be exported to compatible format to integrate with client's CMS and manual entry **OR** • Orders should be exported directly to the client's CMS	1
Retailers should be able to view previous orders	• Previous orders should be displayed with ability to download/view • Previous orders should include itemized product list, total cost, and order/delivery dates	3

Table I: Project Stage 1

Functional	Non-Functional	Weeks
Stage 2		Total: 5
Retailers should have the ability to reorder a previous order	• Reordering should be combined into the same view as past orders	1
Reorders should be sent to client for approval	• Orders should be exported to compatible format to integrate with client's CMS and manual entry **OR** • Orders should be exported directly to the client's CMS	1
Retailers should be able to contact client	• Web portal should provide a simple contact form • Contact form should notify (email, text) the appropriate person within client company	2
Client should be able to export retailer information/orders	• Export format should be able to integrate with VIP/RANKER	4

Table II: Project Stage 2

Functional	Non-Functional	Weeks
Stage 3		Total: 7
Retailers should be able to see recommendations based on local orders.	• Web portal supports recommendations while shopping	3
Real time updates to database	• With API to database, be able to update the product amount after Recorders approve the order.	2
Retailers should be able to see when items are on sale for a special deal	• Website emphasises on the special deals to entice the Retailers.	3
Full production	• Website passed the development stage and testing, and ready for full deployment and use	N/A

Table III: Project Stage 3

Deliverables	Date of Completion
Stage 1 Tentative Completion	11/2/2018
Stage 2 Tentative Completion	1/18/2019
Stage 3 Tentative Completion	3/22/2019

Table IV: Project Deliverable Completion Dates

3.2 Feasibility Assessment

Realistically we should be able to implement all of the requirements for this project. However, we may face several challenges. Our main concern is the availability of existing client data. Because the main VIP system used by Doll to process orders is now considered legacy software, we are limited in our interactions with existing orders and retailer data. Another challenge is access to this system's API, as well as any questions regarding interfacing with the VIP API. This may cause delays in the project timeline, while we wait on responses from the VIP vendor. Additionally, the legacy status of the VIP software may mean that we are forced to abandon the use of an API, and instead rely on manually exported data reports to interface with Doll's database.

3.3 Personnel Effort Requirements

Task	Description	Time Estimate
Elicit Requirements from Doll Distributing, VIP, and Blue Compass	Ensure that our plans and implementation are up to Doll's standards.	30 Hours
Research Laravel	Practice skills to have a baseline to work from.	50 Hours
Research AWS	Find out the optimal way to implement our hosting service.	25 Hours
UI Mockups	Perform initial drawings of the UI and develop CSS/HTML for it.	50 Hours
Database Seeding	Create seeders to test out databases and user creation/authentication	40 hours
Setup Continuous Build Pipeline	Set up an environment so that continuous integration can be used throughout the project	20 Hours
Setup Database	Create tables in a database so that it can be populated with data from VIP	40 Hours
Create CSV Parser	Create a server side script to parse CSV files and populate the database	60 Hours
Implement User Login	Create a system for users to login and validate their Liquor license and other credentials.	100 Hours
Implement basic database updates	Allow the database to be manually updated through a new CSV file	80 Hours
Display available products	Allow verified users to browse available products	150 Hours

Table V: Personnel Effort Requirements

Task	Description	Time Estimate
Implement Shopping cart	Create a page where verified users can add available items to their cart to build an order	150 Hours
Automate Email to order representative	Have an Email with order details be automatically sent to sales representatives after an order is placed	50 Hours
Implement user history	Allow users to browse their history and past orders to help reduce time to make repeated orders	60 Hours
Implement reordering	Allow users to place a repeat order from their order history	50 Hours
Implement real-time Database updates	Allow the database to update in real-time as orders are placed. Hopefully by interfacing directly with VIP	60 Hours
Implement promotional deals	Allow users to see which products have promotions or sales going on, and apply those to any relevant orders	50 Hours
Software Documentation	Continuously document code and ensure that it is easily interpretable for when maintenance is needed.	100 Hours

Table VI: Personnel Effort Requirements (cont.)

3.4 Other Resource Requirements

One set of resources that we will need are tutorials for the tools that we will use, such as Python and PHP. We will also need to access VIP in some way. We will also use the HTML and CSS code for Doll's website. These resources will be acquired from the current hosting provider, BlueCompass. [1]

3.5 Financial Requirements

We plan on hosting the project using services provided by AWS.

An initial analysis of our project determined that we would likely require the following services, and incur the associated costs.

Amazon EC2 On-Demand Pricing t2.small - $0.023 Per hour

S3 Standard Storage - $0.023 per GB

Amazon EBS - $0.10 per GB/Month

Our choice to use AWS for hosting our project means that costs can be very elastic. As a result, we estimate that during development, our costs will not exceed $2.00/month. This cost will eventually grow as the project moves in to production. However, we estimate that given the size of Doll's business, the deployment platform is more than adequate. However, deploying via AWS means that we are able to scale up the app as needed, helping Doll better manage retail orders during peak ordering times, and scaling down during less active periods. [9]

We have also incurred some additional cost as a part of working with Doll's existing hosting service. We have participated in several meetings with BlueCompass, and their hourly labor rate of $75 cost the client roughly $150, which we approved ahead of time. However, we have elected to take part in additional meetings with the company only if absolutely required.

Our only other resources include our code platform and development tools, which are free and open-source.

4 Closure Materials

4.1 Conclusion

Doll Distributing currently handles orders and interactions with its retail customers through text, email, and phone conversation. Many of their retailers operate during all hours of the day, which makes these current means of communication less than ideal. Orders submitted from bars and restaurants are often submitted in the middle of the night, and not processed until the next day. As a result, ordering is a very manual process, and retailers are not easily able to ask questions and get updated information from product representatives. Our online ordering platform will provide a web-based application accessible at all times, from any electronic device. Our platform will provide a form through which retailers can ask questions about products, and see recommendations of new products, and other retailers in their area. Most importantly, our platform will provide an interface for Doll's retailers to create and submit orders, see past invoices, and quickly reorder common products. This platform will allow retailers to quickly order products at any time, ask questions about products, and get suggestions from Doll and other distributors about new and promotional products. Overall, this project aims at streamlining Doll's ordering process, and improving their relationships with retailers as a result.

4.2 References

[1] Blue Compass Digital, “Blue Compass | Award Winning Web Design and Digital Marketing,” *Blue Compass | Award Winning Web Design and Digital Marketing*. [Online]. Available: https://www.bluecompass.com/. [Accessed: 27-Oct-2018].

[2] D. Distributing, “Doll Distributing,” *Doll Distributing Beverage Company*. [Online]. Available: https://www.dolldistributing.com/. [Accessed: 27-Oct-2018].

[3] “Django Homepage,” *The Web framework for perfectionists with deadlines | Django*. [Online]. Available: https://www.djangoproject.com/. [Accessed: 27-Oct-2018].

[4] “Docker,” *Docker*. [Online]. Available: https://www.docker.com/. [Accessed: 27-Oct-2018].

[5] “Flask Homepage,” *Welcome | Flask (A Python Microframework)*. [Online]. Available: http://flask.pocoo.org/. [Accessed: 27-Oct-2018].

[6] “Hypertext Preprocessor,” *php*. [Online]. Available: http://php.net/. [Accessed: 27-Oct-2018].

[7] “Jinja2,” *Welcome | Jinja2 (The Python Template Engine)*. [Online]. Available: http://jinja.pocoo.org/. [Accessed: 27-Oct-2018].

[8] T. Otwell, “Laravel Homepage,” *Laravel - The PHP Framework For Web Artisans*. [Online]. Available: https://laravel.com/. [Accessed: 27-Oct-2018].

[9] “Pricing,” *Amazon*. [Online]. Available: https://aws.amazon.com/pricing/?nc2=h_ql_pr. [Accessed: 27-Oct-2018].

[10] “Python Homepage,” *Python.org*. [Online]. Available: https://www.python.org/. [Accessed: 27-Oct-2018].

[11] “Retailer Portal,” *Retailer Portal | VIP*. [Online]. Available: https://apps.vtinfo.com/retailer/user/login. [Accessed: 27-Oct-2018].

[12] “Team Repository,” *GitLab*. [Online]. Available: https://git.ece.iastate.edu/sd/sdmay19-22. [Accessed: 27-Oct-2018].

[13] “Team Website,” *sdmay19-22 • Online ordering platform*. [Online]. Available: https://sdmay19-22.sd.ece.iastate.edu/. [Accessed: 27-Oct-2018].

[14] “The drinks are on VIP.,” *Route Accounting for Beverage Distributors - VIP*. [Online]. Available: https://public.vtinfo.com/. [Accessed: 27-Oct-2018].

[15] “Welcome ToTapWiser,” *TapWiser*. [Online]. Available: https://store.tapwiser.com/. [Accessed: 27-Oct-2018].

4.3 Appendices

Figures 3: Gantt chart for project stages and the tentative schedules on when they will be finished

Figure III: Gantt Chart for Semester 1 and 2 (Tentative)