Doll Distributing Ecommerce System

FINAL REPORT

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

Alec Harrison - DevOps
Caleb Olson - UI/Database
Daniel O'Neill - Email Guru/Presentations
Mitchell Bennett - Communications
Thomas Staudt - UI
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

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 VIP-provided website that shows additional product and brand information

0 Executive Summary

Currently, Doll Distributing has an ordering system that requires sales representatives to meet clients in person to take orders. This can cause their clients to feel pressured into buying other products or being inconvenienced to place orders immediately. There is no way for clients to learn about new products or deals on their own time. They currently must call Doll's representative and hear about current/upcoming deals.

The task was to create an e-commerce system for Doll Distributing LLC that will interact with their current inventory system. A system will be created to allow clients to view Doll's inventory, place orders that will be sent to account representatives, and be able to view their historical orders. This will be linked from their current site and will not allow anyone to access the website without an authorized account and with a valid liquor license.

1 Requirements specification

1.1 Functional requirements

Functional Stage 1:

- ➢ The user shall be able to register an account using a valid liquor license.
- ➢ The user shall be able to log in with their registered account.
- ➢ The user shall be able to view available products.
- ➢ The user shall be able to send an order request to Doll.
- ➢ The site shall have a database to store user data and product data.
- ➢ The user shall be able to view previous orders.

Functional Stage 2:

- ➢ The user shall be able to reorder a previous order.
- ➢ The user shall be able to contact a Doll representative.
- ➢ The client shall be able to export user information and orders

Functional Stage 3:

- ➢ The site shall provide a solution for daily database updates.

1.2 high-level requirements & Use-cases

The use-cases for the project are defined as follows:

- Retailer creates account with valid email and liquor license information
  - Email is sent to Doll admin
  - Account is validated by Doll admin
  - Retailer's pricing group and account representative email are entered by Doll admin
- Retailer logs in with validated email and password
- Retailer creates new order
  - Views available products
  - Adds products to their cart
  - Checks out their cart
  - Email is sent to Doll order representative for verification and approval
- Doll admin updates product and pricing information
  - Doll admin extracts CSV data from Doll CMS
  - Doll admin logs in
  - Doll admin uses admin page to upload CSVs
  - The website replaces existing product and pricing information with data from each CSV
- Retailer updates account information
  - Retailer logs in

        - Retailer uses “settings” page to change account password
- Doll admin manages retailer account
    - Doll admin logs in
    - Doll admin uses admin page to remove/modify retailer accounts
    - Database is updated with account changes

1.3 Non-functional requirements

Non-functional Stage 1:

- ➢ User account registration shall require a valid liquor license.
- ➢ User account registration shall require approval from a site administrator.
- ➢ User accounts shall not be able to view any information of other accounts.
- ➢ The site shall integrate with the current Doll Distributing site.
- ➢ The site database shall be of a standardized format.
- ➢ The site shall show all available products in a users region.
- ➢ The site shall incorporate the BrandFinder website.

Non-functional Stage 2:

- ➢ Reordering shall be combined into the same view as past orders
- ➢ Orders shall notify (email) the appropriate person
- ➢ Updates shall be as quick and painless as possible

Non-functional Stage 3:

- ➢ The Website’s style shall match Doll’s site

2 System Design & Development

2.1 Design plan

The table shown in Figure 1 outlines a list of functional and non-functional requirements derived from discussions with the client, and the team's understanding of the frameworks and services available for use within the scope of this project.

Requirements

Functional (What functions the project needs to perform)	Non-Functional (How the project's functions must perform, constraints)
Stage 1	
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.
Retailer information/orders shall be stored in a database	• Database should be of standardized format/structure • Database must be private and secured such that only client is able to access
Retailers shall 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
Retailers shall 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
Orders shall be sent to client for review	• Orders shall be sent to the correct

	Doll representative
Retailers shall 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
Stage 2	
Retailers shall have the ability to reorder a previous order	• Reordering should be combined into the same view as past orders
Reorders shall be sent to client for approval	• Past orders shall be saved and easily reordered and resent to Doll representative
Retailers shall 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
Client shall be able to export retailer information/orders	• Export format should be able to integrate with VIP/RANKER
Stage 3	
Full production	• Website passed the development stage and testing, and ready for full deployment and use

Table I: Project Requirements

2.2 Design Objectives, System Constraints, Design Trade-offs

To fulfill these requirements,

- The team reached out to the client's website hosting company to gain knowledge of how the existing website is deployed, and the challenges associated with integrating the proposed requirements with the client's existing website.
- Research was done to determine which web development framework would be most efficient for the project. Factors such as team experience, flexibility, and documentation of the framework were all considered.
- Additionally, the team also reached out to the software company behind VIP, the client's order management and inventory system.

2.3 Architectural Diagram, Design Block Diagram -- Modules, Interfaces

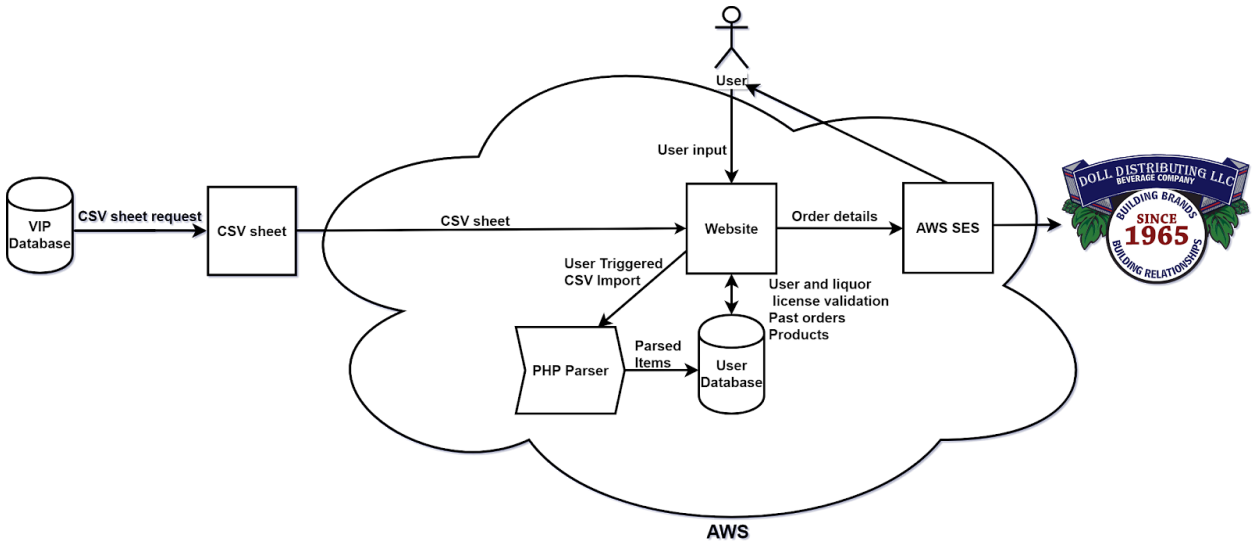

Figure I: Architecture Diagram

2.4 Description of Modules, Constraints, and Interfaces

Since the VIP database cannot be directly accessed, the data must be obtained using exported CSV files. Administrators on our site can import data from these CSV files using the import page. These imports use our PHP parser, which imports the data to our database. After the import, retailers can view the imported data and order the products that are listed using the items page and shopping cart feature. When an order takes place, the order request email is sent to a Doll Distributing representative. The rep will then be able to approve the order and notify the retailer of the approval.

3 Implementation

3.1 Implementation Diagram, Technologies, Software Used

Our current design is building a Laravel web app that can be run within a docker container on any platform. An AWS Linux Server is used to host the container. Customers can view the web app in browser. While viewing the page, they are able to place orders through selecting items on the webpage. Orders are placed via PHP jobs when a customer finalizes their order and notifies the web app that they wish to place it. We retrieve product and pricing data from CSV files exported from Doll's VIP account. That data is then used to populate the ecommerce site and allow validated users to view the items and select them for orders. User logins are supported by a database that is also hosted on AWS. The database stores user details such as login information, liquor license information, and previous orders.

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 in Figure I, is a good example of how streamlined and simple our system is, while still containing all intended features. The initial data population is very linear. We use CSV sheets to quickly add inventory and pricing data. The CSV sheet is retrieved from VIP, parsed and then sent to the website to populate available inventory and prices of available items.

The website utilizes Vue.js to turn the available inventory into a list of cards that show product details to the customer. It is in a list format to easily accommodate for changing numbers of products over the course of each season. Ordering is then a simple process for the customer. A strong area of this design is that the customer can 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 E-mails 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 resulted in an efficient product.

3.2 Rationale for Technology/Software Choices

Laravel [3] is a very powerful web application framework. It can do everything we need for this platform. It allows us to easily manage all resources regarding CSS and Images, and can also be placed in a Docker container [1].

AWS works 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 clients primary database that stores all of their inventory and pricing data. It is difficult to utilize because we do not have full access to it. The solution to this access issue is to manually export and upload CSV files for inventory data.

Docker[1] is a great tool for deployment of applications. We decided to use it because it allows everything to be placed in one “container”. This container holds all necessary dependencies and can be easily moved between hosting services. This portability enhances the ease with which the project can be handed off or redeployed. Having all of the dependencies included also allows the project to run anywhere that Docker can without the need to install any other technologies.

3.3 Applicable Standards and Best Practices

Maintainable - Able to be passed to another team that will be able to manage it.

Usable - Users of the site will be able to use its features without needing a large amount of training.

Rapid development- Development environment can be quickly deployed in a consistent manner

PSR-1 Basic Coding Standard [5] - This standard specifies the general structure of a file, as well as the naming convention for classes and methods.

PSR-2 Coding Style Guide [6] - This standard specifies a coding style for developers to follow. This includes details such as line indenting and length, and the position of opening and closing braces for methods, classes, and control flow statements.

IEEE 1028-2008 Standard for Software Reviews and Audits [4] - This standard describes various software review processes that are performed by developers and other project personnel.

4 Testing, Validation, and Evaluation

4.1 Test plan

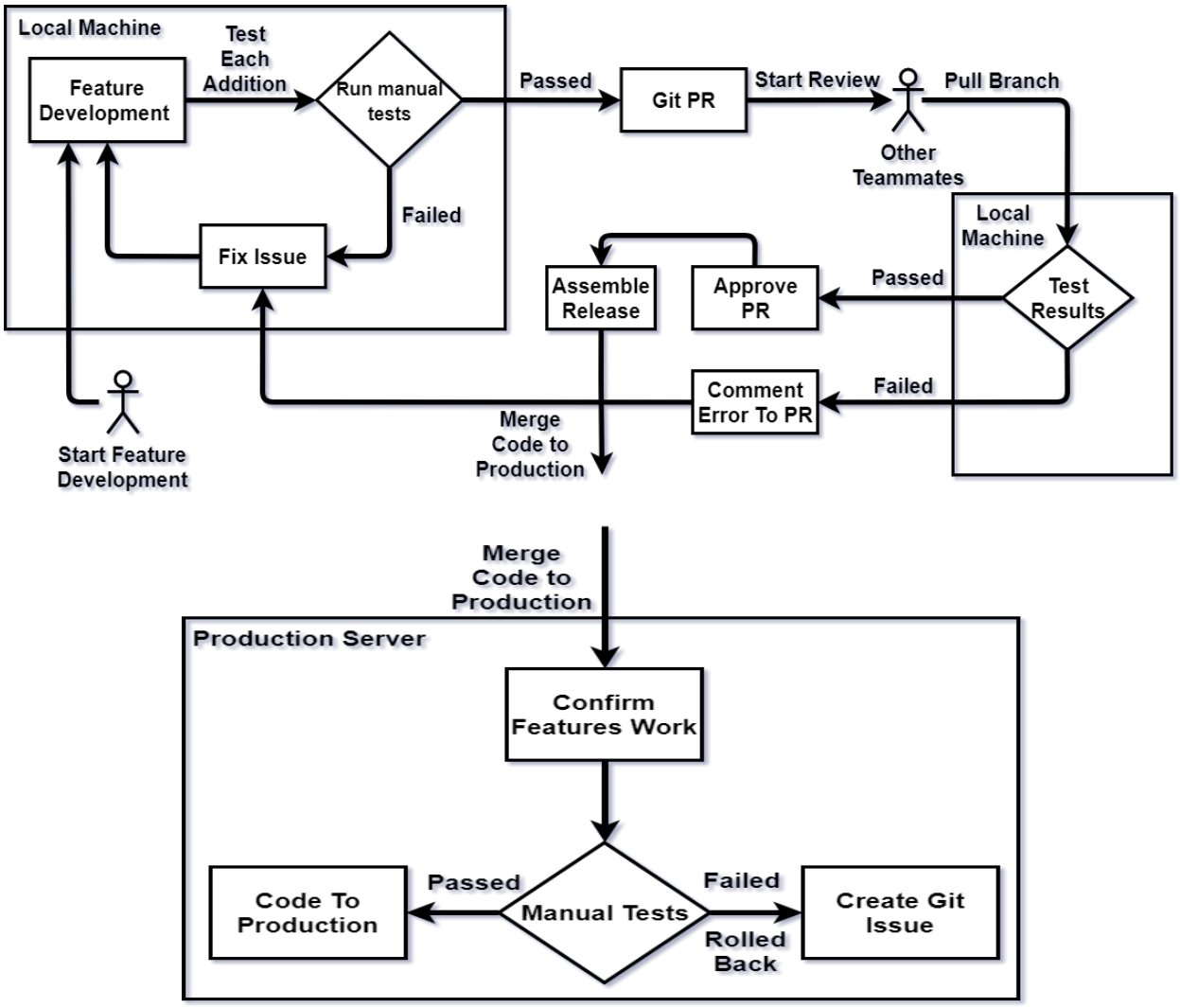

Figure II: Testing Plan

4.2 Unit testing

PHPUnit [2] uses assertions to verify that the behavior of the specific component being tested behaves as it is expected to behave. After code is created, it should go through tests that are made to be sure that any changes that was made to the PHP code will function as expected and not do anything that isn't expected.

4.3 Interface testing

This type of testing will be from a user's standpoint. The test will run in a browser-type environment and

test the app's links to make sure that they go where is expected of them. This plan will allow us to follow a set of instructions in order to insure our interfaces operate as expected.

4.4 System integration testing

Since the application will be a separate instance from where Doll currently has their website, testing for integrations will not be necessary except for manual/automated testing to see if links go back correctly to their main page.

4.5 User-level testing

User-Level testing will be done on a feature by feature basis. This will be done prior to merge requests being approved. This will allow a user to manually test a feature completely to try and break it. This can help uncover unexpected user behavior, and also that the feature is implemented in such a way that it is intuitive for the user to use.

4.6 Validation and Verification

Acceptance testing will be getting approval from Doll.

Doll will check over the website on new features that are implemented to make sure it is what they expected it to look like and agree to go ahead with the implementation. This also includes after approval to go live, Doll will be able to receive suggestions from their clients so that the application can be implemented/improved on.

4.7 Evaluation

Our manual testing was very successful throughout the semester. It was helpful to have code reviewed by multiple group members whenever a feature was added. This encompasses user testing, interface testing, and system integration testing.

5 Project and Risk Management

5.1 Task Decomposition & Roles and Responsibilities

Alec Harrison - Infrastructure, UI/UX, and Project Management
Caleb Olson - UI Design, User registration, User modification
Daniel O'Neill - Amazon Email Service, Email templates, Presentations, and Documents/Poster
Mitchell Bennett - Communication/Inventory Scripting
Thomas Staudt - UI Design, User modification, Manual feature testing
Thomas Wesolowski (TJ) - Backend order processing, Doll CMS -> database integration

5.2 Project Schedule

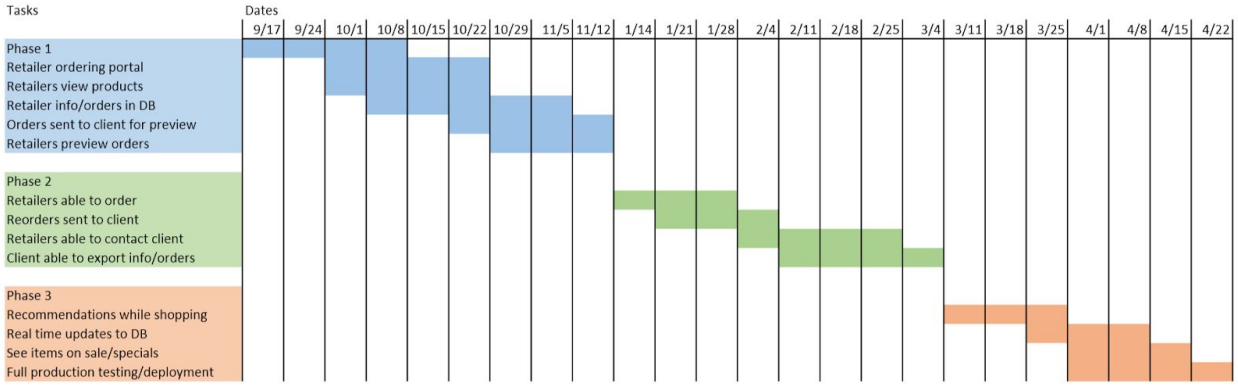

Figure III: Proposed Project Schedule

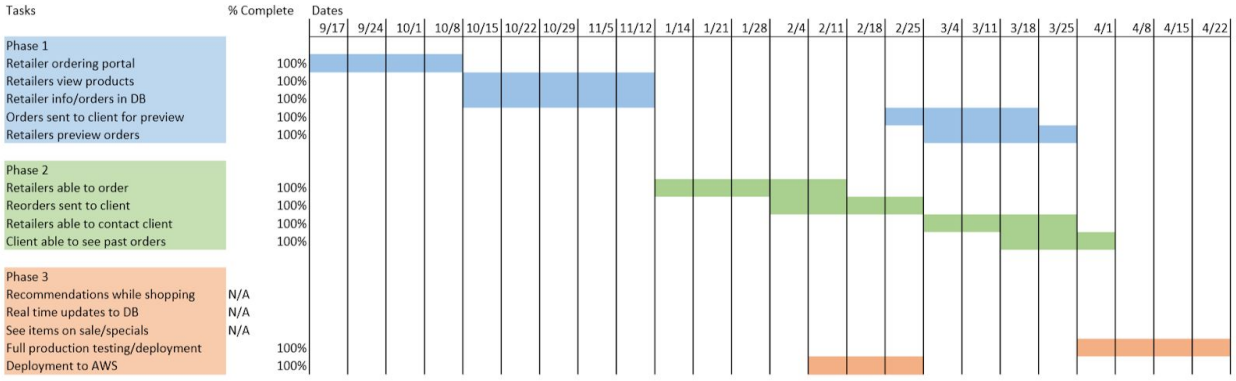

Figure IV: Actual Adjusted Project Schedule

5.3 Risks and Mitigation: Potential vs. Actual and how they were mitigated

The team identified potential risks including the security of user information, validation of liquor licenses, and regulation around the sale of liquor. We also identified a more serious risk resulting from the possibility of not being able to get real-time data from VIP.

We found that some of these potential risks did become issues during project development; however, many were easily mitigated. Our use of the Docker platform allowed for the creation of a database connection that did not require any external ports or connections; this allowed us to better restrict database access to only the web application. The risks surrounding validation of liquor licenses was addressed by the client's desire to have their existing order representatives remain as part of the order process. The team then designed the process of validating users and approving orders around sending emails to Doll representatives for manual review. This design decision also gives Doll the ability to handle liquor regulations under their own discretion.

While the team was able to address many of the potential risks identified, one risk did cause complications during development. During a meeting with VIP, which was meant to discuss the possibility of access to Doll's CMS data, we learned that VIP was creating their own online platform and thus would allow real-time API access to their platform. The team evaluated several possible mitigations, eventually settling on a manual process for database updates. While unfortunately a manual process, our team created custom reports within Doll's CMS platform that could be easily exported as CSVs. This allows Doll admins to upload pricing and product availability to the application at any time. This solution was discussed with the client, and agreed upon on the basis that order representatives manually update this information already, so a simple step to update the website would not introduce significant additional work.

5.4 Lessons learned

Throughout this project, the team learned a significant amount about the importance of communication. Maintaining contact with Doll was a key part to progress in the project. When our communications slowed down, so did progress, the opposite was also true. The team also learned about how progress can be impeded by unfamiliar technologies. A significant portion of the effort put into the project was because of several group members needing to learn how to effectively utilize the technologies involved.

6 Conclusions

6.1 Closing remarks for the project

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. This online ordering platform provides a web-based application accessible at all times from any electronic device. Most importantly, the platform enables Doll's retailers to create and submit orders, see past invoices, and quickly reorder common products. Overall, this project aims at streamlining Doll's ordering process, and improving their relationships with retailers as a result.

6.2 Future work

Future work on this project could include many feature updates. The biggest part that could be improved upon would be the overall look and feel of the website. All the functionality is there, but the site is not extremely colorful or appealing to the user. Other things that could be improved upon would be automated population of the database. The current method of manually importing inventory and pricing is very inefficient. More advanced features such as order suggestions could also be added in the future, as well as combining the brandfinder pages with the items page to allow for users to easily see images of the products that they are ordering without switching between sections of our website. There could also be added functionality for admins that are more non-technical and more user friendly. We could also add metrics to see the flow of purchases through the site.

References

[1] "Docker," Docker. [Online]. Available: https://www.docker.com/. [Accessed: 27-Oct-2018].

[2] S. Bergmann,"PHPUnit Manual" *PHPUnit - A programmer oriented testing framework for PHP.* [Online]. Available: https://phpunit.readthedocs.io/en/7.4/ [Accessed: 2-Dec-2018].

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

[4] IEEE Standards Association, "IEEE 1028-2008 Standard for Software Reviews and Audits", [Online]. Available: https://standards.ieee.org/standard/1028-2008.html. [Accessed:27-Oct-2018].

[5] P Jones, "PSR-1: Basic Coding Standard", [Online]. Available: https://www.php-fig.org/psr/psr-1/ [Accessed - 21-Apr-2019]

[6] P Jones, "PSR-2: Coding Style Guide", [Online]. Available: https://www.php-fig.org/psr/psr-2/ [Accessed - 21-Apr-2019]