

University of Victoria
Computer Science Co-op
Work Term Report
Spring 2019

Implementation of Technologies and Design
Related to the Development of the
XXXXXX E-Commerce Platform

XXXXXX
Web Development
Victoria, BC, Canada

Nathan Jenkins-Boale
V00851969


1
Computer Science
njboale@uvic.ca

April 26th 2019

Supervisor's Approval: To be completed by Co-op Employer

This report will be handled by UVic Co-op staff and will be read by one assigned report marker who may be a co-op staff member within the Engineering and Computer Science/Math Co-operative Education Program, or a UVic faculty member or teaching assistant. The report will be either returned to the student or, subject to the student's right to appeal a grade, held for one year after which it will be destroyed.

I approve the release of this report to the University of Victoria for evaluation purposes only.

Signature:  Position: R&D Engineer Date: 23 April 2019

Name (print): Sergio Perez E-Mail: sergio@ergonomyx.com

For (Company Name) Ergonomyx Technologies Canada Inc.

Abstract

The following report outlines core software technologies incorporated into the XXXXXXXX e-commerce site and illustrates how these different areas interact fluidly with each to create a seamless user experience. This document will discuss the overall database design and implementation, along with outlining why the design structures chosen best suit the needs of the project. It will address how the database interacts with currently active areas, discuss what problems may arise from the current design and what procedures were taken to minimize these risks. The report will discuss the interactions between the micro framework flask, front end, database design, and information sharing to promote user flow through the system.

Report Specification

Audience

The following report is intended for individuals who want an overview of the implementation and functionality of the current XXXXXXXX e-commerce platform but only have a limited knowledge of web development, database design and interactions with an application programming interface. This report is designed for a someone who wants a summary of the technologies and their interactions of the e-commerce platform but may not have an in-depth understanding of ideas related to writing code, database queries, efficiency and other technical ideas of development.

Prerequisites

The reader is expected to have an understanding of web page design and how a front end interacts with the backend. The reader should also have a good understanding of the general ideas and concepts that make up an e-commerce site. These concepts are, but not limited to, cart, checkout, store, users and how specific user information and abstract concepts are maintained and enforced in a database.

This report assumes that the reader has some experience working within the software development life cycle, understands the development of front end and back end services to publish and host web pages and comprehends ideas relating to database design.

Purpose

The purpose of this report is to outline how several different, independent technologies are implemented into the XXXXXXXX e-commerce platform to achieve its desirable functionality and provide a seamless experience for guests. This report should give members within the

company and future co-op students a broad understanding of the implementation of features and design of the e-commerce platform.

This report may also be useful in understanding why certain design decisions relating to database, and or software tools were considered and implemented.

Table of Contents

Abstract..... i

Report Specification..... ii

Table or Figures and Tables..... v

Introduction..... 1

Main Body..... 3

Personal reflections..... 12

Conclusion..... 12

Acknowledgements..... 14

References..... 14

Table or Figures and Tables

Figures:	Page
Figure 1:	FRONT END USER WEB PAGE FLOW.....4
Figure 2:	INSERT PROCESS OF NEW TABLETOP COMPONENT.....4
Figure 3:	INTERACTIONS WITH API AND DATABASE.....7
Figure 4:	ASYNCHRONOUS CODE FLOW.....8
Figure 5:	FRONT END USER WEB PAGE FLOW.....11
Tables:	
Table 1:	PRODUCT COMBINATION EXAMPLE.....9

Introduction

This report outlines the development of the XXXXXXXX Ltd. (“XXXXXXX” or “the company”) e-commerce platform.

XXXXXXX is a new company based out of downtown Victoria. The company aims to create workplace fitness products that promote a healthier office lifestyle. There are currently two hardware products in development: a standing desk, and an under desk bike. Both these products should allow seamless integration with the users’ phone and computer via a web-app and a mobile-app. The company is young, and filled with many co op students actively learning from each other, working towards solutions in a lively and resourceful environment.

The current co op student was tasked with developing an e-commerce platform for selling the company’s smart standing desk and smart under desk bike solutions for the workplace. This market place was instructed to be developed from scratch, without the use of third party e-commerce development tool kits. The company felt that without using out of the box solutions they would be able to offer more functionality and customizability to the customer for less long term cost. In addition, the company would not be tied to any third party and would be able maintain and manipulate the data as required.

The co-op student was to develop a platform that would interact directly with an existing application programming interface (API). This API connects to a database, insuring all database bound traffic will pass through a secure tunnel [Figure 2]. The e-commerce platform would also need to incorporate existing XXXXXXXX accounts for existing services. The user should have one account for all aspects related to XXXXXXXX software.

The purpose of this report is to give an overview of the technologies implemented and functionality of the current XXXXXXXX e-commerce site. A company’s online presence is

necessary in today's retail market. Approximately 14 % of all retail sales in the US occur through e-commerce platforms with sales expected to continue to grow in the following years [1]. XXXXXXXX online market presence is necessary to broaden their product reach and increase the possibility the company's success in the long term.

The Eronomyx e-commerce platform should provide the customer with a fluid, seamless experience. They would be able to explore, customize and securely purchase product solutions. This process should be familiar to the consumer with few barriers between exploration and purchasing; operating in a similar fashion to currently active e-commerce designs.

The two primary product designers for the e-commerce platforms are co op students. The report will serve as an overview for current XXXXXXXX employees of the work completed on the project through the 2019 Spring work term. It will also identify the designs and technologies implemented, their interactions with each other and give a brief outline to why and how these decisions address the platforms requirements.

Main Body

The following section will be divided into several sections: Database design, routes and request handling, API, Braintree, connections to the api, technical reflections, and user front end flow. The sections illustrate how these independent areas operate, and function together to provide the site's requested functionality.

Database Design

The design of the database includes several familiar concepts: users, products and supply. Other sensitive data including shipping and credit card information is handled by a third party known as Braintree; a PayPal company.

Products:

The design of the products table requires that it be scalable for the new additions of products in the future. It demands that each product may be formed from multiple components, where each component may be sourced from multiple suppliers.

To address the requirements, a database structure known as bill of materials was employed.

Bill of materials is a design strategy which allows for scalable product creation from a subset of components, where each component may be a subset of other components and so forth.

Each fully formed product belongs to the highest inheritable table. Each specific product inherits a general product's properties, including a type and price along with adding some product specific characteristics. Each product is composed of one or more components each with some subset of attributes. For example, a desk is composed of two components, a leg and a tabletop where a leg has a price and may be two colours, black or white and a table top is composed of three attributes, price, colour and size [Figure 1].

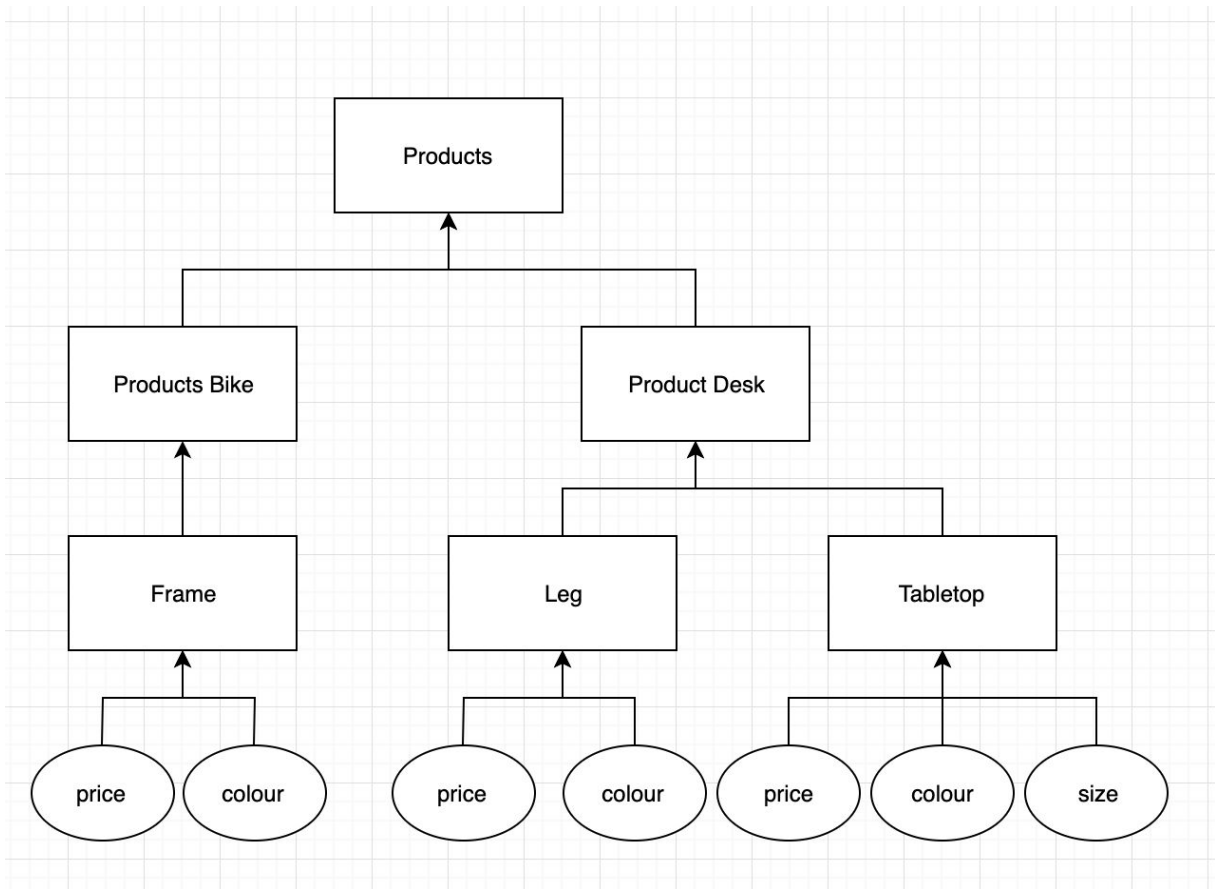


FIGURE 1 : BILLING OF MATERIAL DESIGN FOR PRODUCTS

When the company declares that they want to support a new type of component, for example a new tabletop with a different size. This newly formed component is joined with all other existing components that make up its respective product type. This creates one or more new unique product IDS to identify this unique combination.

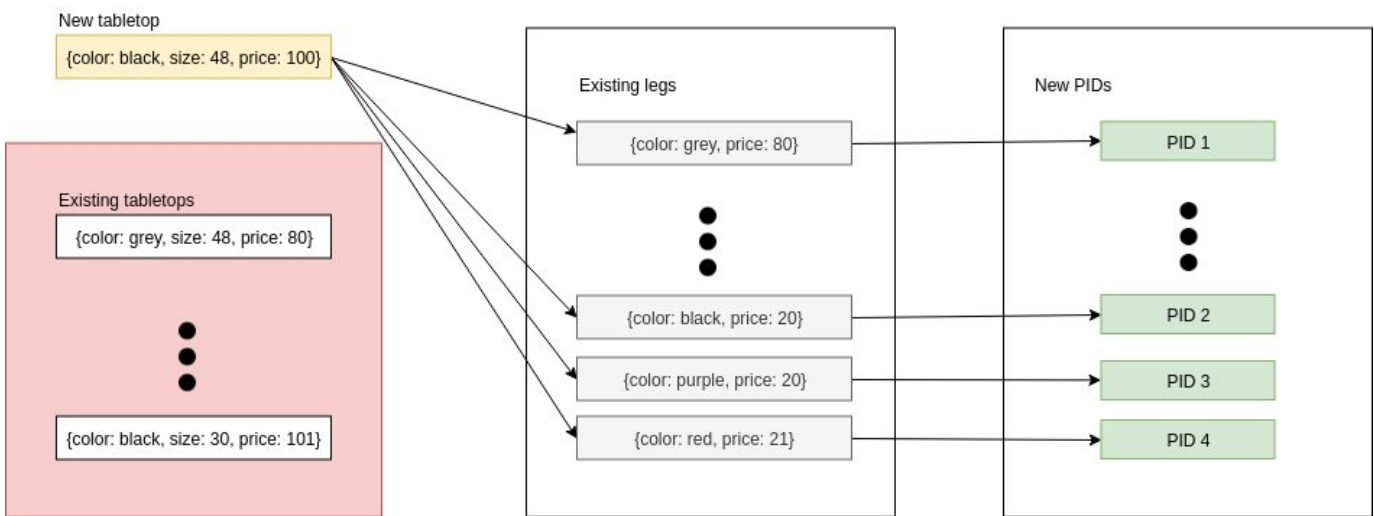


FIGURE 2 : INSERT PROCESS OF NEW TABLETOP COMPONENT

These joins are maintained in terms of database triggers. A database trigger insures that when an operation is performed on a table (insert, delete, update) a specific function is executed. In terms of products, when a new component is inserted the proper joins will take place to insure that the appropriate amount of unique products are created and stored in the database [Figure 2].

The price of a product is calculated by using the accounting function design of a bill of materials. The cost of a product is determined by the sum of the costs of its children. Unless an alternative price is updated.

Inventory / supply

Supply is considered in terms of the most atomic part of each product, it's components. When a supply component is inserted into the database, it is given a timestamp of its creation time, unique id to identify this specific component and the user that created it. These supply components may be used to create fully formed products at the users request. Since each supply component has a one to many relationship with products, the database considers supply only when checking if a product is in stock. To check if a product is in stock, we check if the individual components that make up the product are available to add to the customer's cart.

Orders

The cart table has a product id to identify the product, a user id to identify which users' cart the product is in, quantity field for the amount of this specific product they are buying and a price. This price column insures that when the product price is updated on the administrative side, that the product within their cart does not change.

Routes and Request Handling. (Flask)

Flask is used to process and handle web page requests. This is a python micro framework that was also used to implement the web application. It was chosen at the beginning of the site's creation because it uses python; a language familiar to both co op students working on the project. In addition, Flask is also used in the existing web application. It was decided that it would be helpful to not introduce another technology for future co op students to learn.

Flask renders the html web pages by using built in functions and a templating format known as Jinja2. It's syntax is similar to python and allows for programmatic handling of the html, including loops, variables and conditional on page load.

Limitations of flask

Flask renders html documents using a templating service known as jinja2. This means that to load a user specific page variables must be passed in from flask before the page loads. These variables are created in Flask are not available outside of the html document they are passed into or Flask. If the javascript document requires access to the variables it must either request them from flask or passed through the html itself. For example, when the cart page is loaded, the specific cart object must be passed in by flask. If the cart needs to dynamically change after the page load, the javascript file does not have access to this exact object and must read in the value specific variables through the html. This limitation creates a barrier of fluid information sharing between documents.

Application Programming Interface (API)

The API is the only technology that connects directly to the database [Figure 3]. This was in hopes of limiting the end points and protecting user information. The API is written as an

Express server and connects to a Postgres database. The API will also handle some security. The queries are written inside of the API, they are validated before fetching information from the database. This information is returned in the form of a JSON object with is either interpreted directly on the web page via javascript and jQuery. Or is handled in Flask by converted the JSON object to a dictionary.

Braintree

Braintree is a PayPal company that allow seamless and secure handling of customer funds. For liability and security the commerce website and database does not store user billing address, shipping address or credit card information directly. It uses a Braintree extension of Flask to process this information. Using the database created user id, Braintree stores the users information remotely. This information is retrieved and handled securely by the extension. The commerce site does not store any of this information directly.

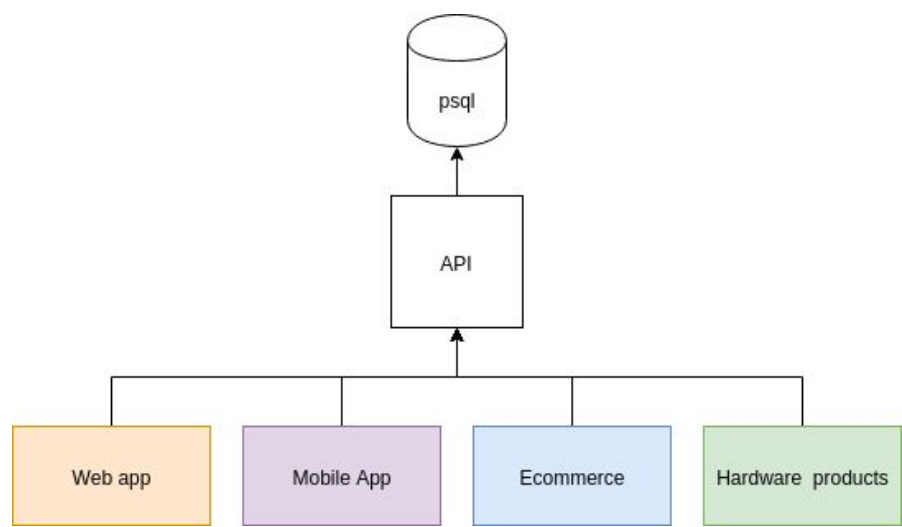


FIGURE 3 : INTERACTIONS WITH API AND DATABASE

Connections to the API and to flask from javascript files

Most accesses to the API from the javascript files are done through the Fetch API. Fetch uses the general idea of a request and response object. This works asynchronously, meaning that the code directly outside of the fetch scope may execute before the fetch response is received. When the

response object is received the code that was executing at that moment stops and the code within the response handler is executed. After the response handling finishes, the program resumes its normal flow [Figure 4]. Asynchronous calls are made to insure that the user may interact with other items on the page and the page is not ‘hung’ performing one request while they wait.

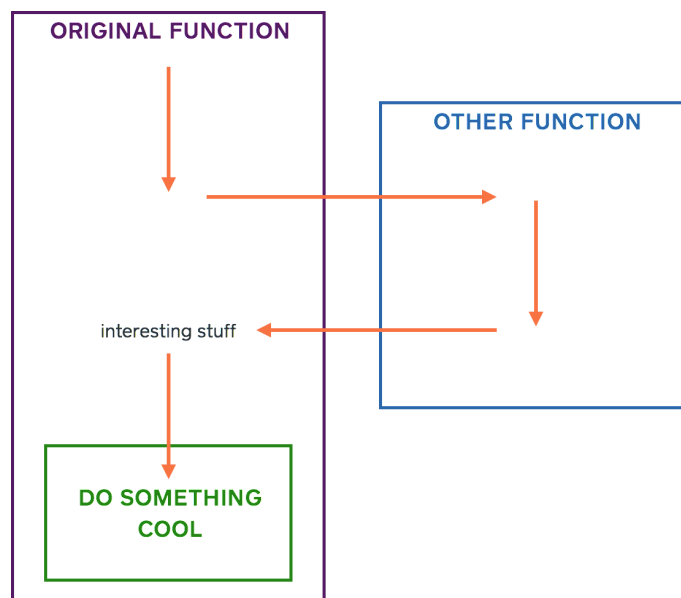


Figure 4 : Asynchronous Code Flow

Technical reflections

The choice of the database design provided some problems while implementing complicated functions such as add to cart and remove from supply or cart. Most problems arise from the fact that any one component may create many different unique product combinations that rely on it. The products in the inventory are required to be ‘built for order’ meaning that fully formed products cannot be considered. For example, a ‘desk’ is not being purchased, in fact, two unique components are being purchased, legs and a tabletop.

A single pair of desk legs may form many different product combinations.

Consider the following supply, where each item represents a unique component currently in stock.

White Leg	White Table Top
	Grey Table Top
	Black Table top

TABLE 1: PRODUCT COMBINATION EXAMPLE

Using this single white leg, there may be a total of three unique products created with it. User A and user B should both be able to add any one of these products to their cart. When either of these users checkouts, the supply will update and a message must be pushed to the other user indicating their cart has changed.

The fact that the same component may be a part of multiple different product combinations in different users cart offers challenges. To address this problem the cart considers and compares components on a users checkout and compares this request to the supply. If there is a difference, the user is warned.

Front End Flow

The flow of a user follows a common path incorporated by many production e-commerce sites [Figure 5]. The user will first start on a **landing page**. The landing page has many intriguing graphics and renders of the products. Along with information of product features and design. The user may also view the **store** page, this page contains all of the offered products. From here they may chose to customise a product or add it to their cart. **The customise page** allows the user to interact with the product by rotating it and altering its customizable features including colour and size. The user will see a live update of their currently selected product and may interact with it directly on the page.

When the user likes their current selection they may add it to their cart. From the **cart page** they will see all of the the products they selected and the product’s respective attributes. The user may change the product’s quantity or remove them entirely.

If the user is happy with the selection they continue onto the **checkout page**. The checkout page shows a summary of their current cart. The user then either selects from previous shipping information or adds new information. This information is entered into the Braintree system. After which they continue onto the payment section; handled by the Braintree extension. Finally, when everything completes successfully the user will received a confirmation email along with tracking information.

This flow is simplistic and familiar to an e-commerce experience.

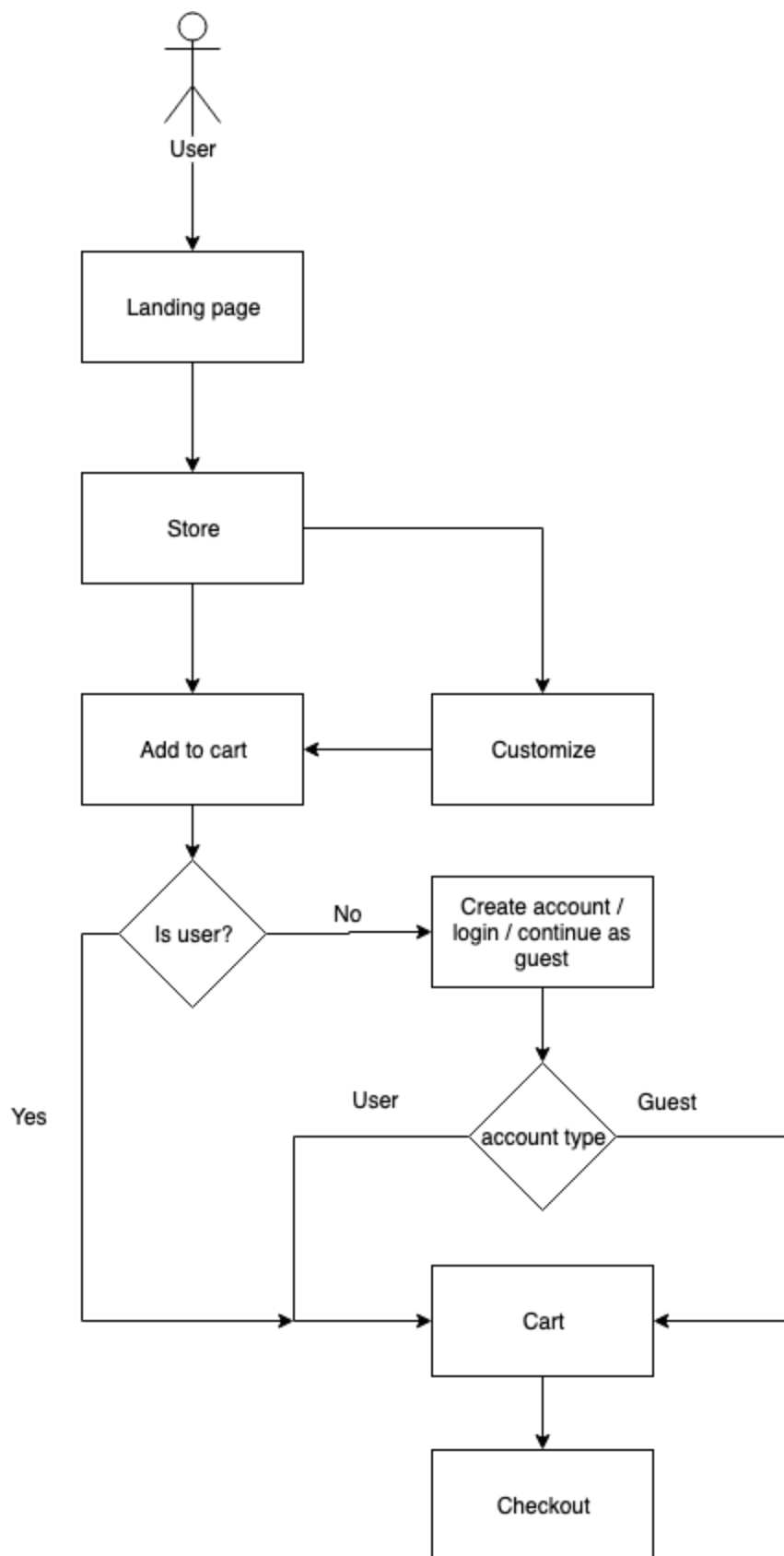


FIGURE 5 : FRONT END USER WEB PAGE FLOW

Personal reflections

I am currently nearing the end of my undergraduate. It was recommended to me to enter into the optional work experience or co op program at UVIC. So far this experience has been enlightening to see what a job within the tech industry might be like. I will graduate from UVIC with a Bachelors of Science, Computer Science focused in Networks and Communications. It has been a great help to see what how a young start up operates, and what a career in website development could offer in the future. I would like to thank my colleges and employers for a continuing interesting and impactful experience.

Conclusion

The XXXXXXXX e-commerce site combines a large number of separate, independently operating technologies to create a seamless and customizable experience for customers to explore and purchase products. The database design incorporates well know techniques including class inheritance and billing of materials. These designs will allow for easy scalability and product configuration in the future.

Flask handles all of the backend page serving and route handling. This micro framework is well documented, well tested and offers lots of extensions including security, administrative tools, along with other services that could be implemented easily in the future using the provided libraries.

The application programming interface provides a level of abstraction between interactions directly with the database. This adds another layer or security by limited the end points, providing query evaluation, and security. The API provides a secure line to modify, add and retrieve information from the database.

Payment, shipping and other user sensitive information is securely stored using the PayPal company, Braintree. It allows for secure transmission and retrieval of user specific information to complete the payment and confirmation process.

Combining all of these areas and technologies will provide users with a seamless and desirable experience to view, interact with, and purchase products.

Acknowledgements

Supervisor: Sergio Perez

Web developer partner: Nolan Kurylo (co-op student)

References

Cited References

[1] Internet Retailer, U.S. Commerce Department, 2017