Portfolio Project for Online Shopping Cart

Johnny Olmedo

Colorado State University Global

CSC480: Capstone Computer Science

Dr. Farr

11 June 2023

Portfolio Project for Online Shopping Cart

In today's digital age, e-commerce has become a crucial part of the retail industry. With more and more consumers opting to shop online, it has become imperative for businesses to have a strong online presence. A crucial aspect of any online business is its shopping cart system. The shopping cart is the backbone of an e-commerce website as it allows customers to select, add and remove items from their virtual cart and proceed to checkout seamlessly. Therefore, this proposal aims to outline the development of a shopping cart system for an online business, which will help streamline the purchasing process and enhance the customer experience.

The Inspiration

Throughout the years as I have been studying computer science and tech people, I have received many questions from, “can you fix my printer?” to “can you make a game/website?”. As a young programmer I doubted my skills and my time management as a programmer, so I declined many offers for designing projects for myself locally. But, in my senior year as an undergrad I have been more and more interested in pursuing my own projects outside of class work. Therefore, I have decided to let my final project be an online shopping cart/web design application.

Relevancy

So why is a shopping cart a relevant solution to modern business problems? Well, in a LinkedIN article DZone reported the leading software trends as of right now are cloud storage, artificial intelligence, and web applications. They also stress how it is imperative to prioritize a software's ability to address a problem, rather than selecting a program that is aesthetically pleasing or intriguing (Business, n.d.). The shopping cart also revolves around another big theme is software, automation. According to the author of “Business Problems You Can Solve With

Software", automation involves utilizing contemporary digital technologies and creative methodologies to improve customer service and decrease operational time. Automation also enables technicians to expedite their work and enhance their customer support. Additionally, automation provides managers with the ability to monitor subordinates and promptly address any potential issues or opportunities (Business, n.d.). Automating checkout processes is one of the advantages of the shopping cart for an online business.

The Project

The goal of this project will be to design a simple framework for an online shopping cart for an online store. The shopping cart will be able to add and delete items from the cart, check out, give tracking information and receipts for a given product. I expect to write the backend of this website using Python and Django. The front end using traditional JavaScript and HTML.

Conclusion for CTA 1 - Proposal

In conclusion, with the ever-growing demand for online shopping, having a robust e-commerce website with an efficient shopping cart system has become crucial for businesses to succeed. This proposal outlines the development of a shopping cart system for an online business, which aims to streamline the purchasing process and enhance the customer experience. As a computer science student, the inspiration for this project came from years of receiving requests for designing projects locally, and the goal is to design a simple framework for an online shopping cart for an online store. With the backend of the website written in Java and the front-end using traditional JavaScript and HTML, this project aims to provide a user-friendly and hassle-free shopping experience for customers. In summary, the proposed shopping cart system is an essential component of any e-commerce website, and its development will help businesses meet the demands of online shoppers and increase their online sales.

Project Plan for Online Shopping Cart

In today's digital age, e-commerce has become a crucial part of the retail industry. With more and more consumers opting to shop online, it has become imperative for businesses to have a strong online presence. A crucial aspect of any online business is its shopping cart system. The shopping cart is the backbone of an e-commerce website as it allows customers to select, add and remove items from their virtual cart and proceed to checkout seamlessly. Therefore, this proposal aims to outline the development of a shopping cart system for an online business, which will help streamline the purchasing process and enhance the customer experience.

System Overview

As we discussed in CTA 1, I will be creating a shopping cart application to aid in the production of sales for an e commerce website. The shopping cart will be able to add, delete and update items selected by the user, along with provide payment information and total amounts.

Major Software

In order to make this shopping cart work properly, a number of different major software components come together to create the cart, and handle functionality, and create a user friendly interface.

Classes

The first class is the shopping cart is the CartConfig() class where we initlize our cart app.

```
“class CartConfig(AppConfig):
    default_auto_field = 'django.db.models.BigAutoField'
    name = 'cart'”
```

The second major class we use is the Cart() class. This class includes all functionality for the cart which includes initializing the sessions, adding, deleting, updating, retrieving total.

```
“class Cart():

    def __init__(self, request):

        self.session = request.session

        # Returning user - obtain his/her existing session

        cart = self.session.get('session_key')



        # New user - generate a new session

        if 'session_key' not in request.session:

            cart = self.session['session_key'] = {}



        self.cart = cart
```

```
def add(self, product, product_qty):

    product_id = str(product.id)



    if product_id in self.cart:

        self.cart[product_id]['qty'] = product_qty

    else:

        self.cart[product_id] = {'price': str(product.price), 'qty': product_qty}



    self.session.modified = True




def delete(self, product):

    product_id = str(product)

    if product_id in self.cart:
        del self.cart[product_id]

    self.session.modified = True



def update(self, product, qty):

    product_id = str(product)
    product_quantity = qty

    if product_id in self.cart:

        self.cart[product_id]['qty'] = product_quantity

    self.session.modified = True



def __len__(self):

    return sum(item['qty'] for item in self.cart.values())
```

```
def __iter__(self):

    all_product_ids = self.cart.keys()

    products = Product.objects.filter(id__in=all_product_ids)

    cart = self.cart.copy()

    for product in products:

        cart[str(product.id)]['product'] = product

    for item in cart.values():

        item['price'] = Decimal(item['price'])

        item['total'] = item['price'] * item['qty']

        yield item
def get_total(self):

    return sum(Decimal(item['price']) * item['qty'] for item in self.cart.values())”
```

Models

It is also essential to define the necessary models to represent the cart and its related entities, such as products and cart items. These models should include fields to store relevant information like product details, quantity, and user associations. This will be enclosed in models.py (Develop, n.d).

URL

Next, a set of views and corresponding URL patterns need to be implemented to handle cart functionality, such as adding items, updating quantities, and removing items. These views should interact with the cart models and provide appropriate responses to user actions (Develop, n.d). Additionally, integrating the cart with user authentication and session management ensures that the cart is tied to the correct user and persists across multiple sessions. This can be achieved using Django's built-in authentication system and session management functionalities. The URL set up can be seen in urls.py.

Templates

To create a seamless user experience, templates should be designed to display the cart contents and allow users to interact with it. These templates utilize Django's template language to dynamically render cart data and provide intuitive user interfaces for adding, updating, and removing items. These templates will come in the form of HTML files, using JavaScript, aJax, Json exensions (Develop, n.d.). By following these steps, the shopping cart will be user-friendly and can seamlessly integrate with their e-commerce site.

Timeline

A wise leader never goes to battle without planning and weighing out the costs. The expected timeline is as follows:

Week 1 Planning and Requirements Gathering

This stage involves understanding the project requirements, identifying the desired features, and defining the scope of the shopping cart app. It typically takes around 1-2 weeks to complete, depending on the complexity of the project (Keeling, 2017).

Week 2 Design and Wireframing

During this stage, the user interface (UI) and user experience (UX) of the shopping cart app are designed. Wireframes and mockups are created to visualize the app's layout and interactions.

Week 3-6 Backend and Frontend Dev

The development process for creating a shopping cart app involves both frontend and backend development. Backend development encompasses setting up the infrastructure, and implementing the core functionalities of the shopping cart app. This phase usually takes around 2-4 weeks, But I expect my project to only take 3-4 weeks. On the other hand, frontend development focuses on building the client-side components, such as the user interface, interactions, and integration with the backend APIs. The frontend development phase typically spans 3-6 weeks, depending on the complexity of the UI and the number of screens/pages, for this project front end should only take 1-2 weeks.

Week 7 Testing Assurance

Testing is a critical phase to ensure the functionality, performance, and reliability of the shopping cart app. It involves various types of testing, including functional testing, usability testing, performance testing, and security testing (Keeling, 2017). The duration for testing can range from 1-2 weeks or more, I expect.

Week 8 Deployment and Launch

Once the shopping cart app has undergone thorough testing and quality assurance, it is ready for deployment to a production environment. For This I plan on launching throught AWS. I expect for this to last 1-2 weeks.

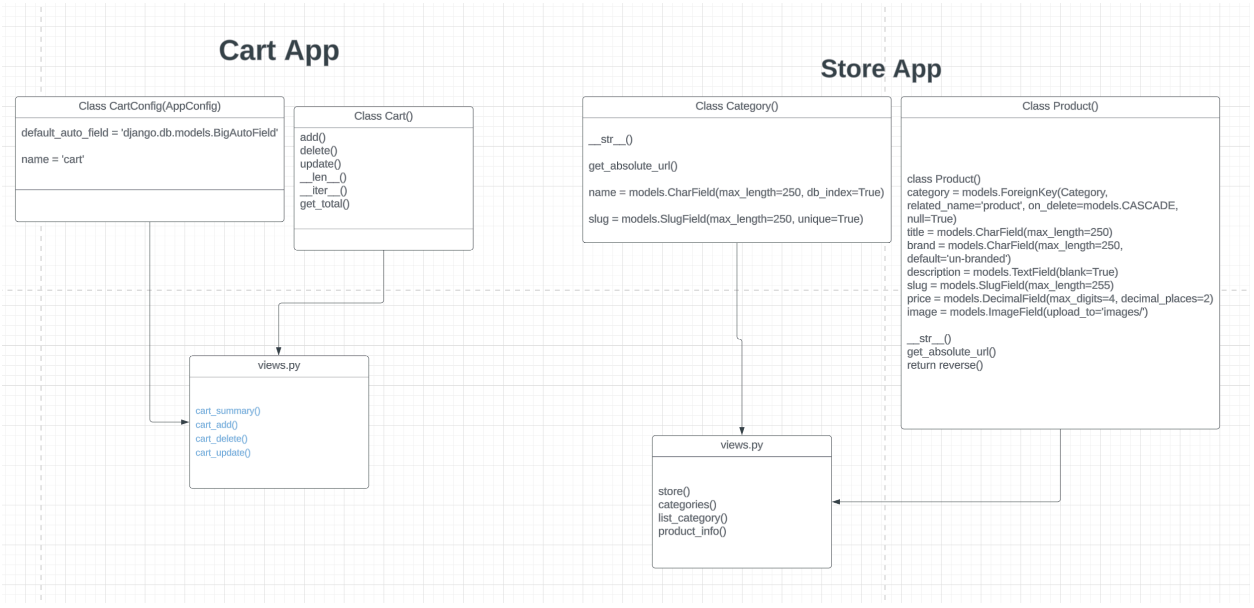
Conclusion for CTA 2 – Project Plan

In conclusion, the development of an online shopping cart system is crucial for enhancing the customer experience and streamlining the purchasing process in the e-commerce industry. By following a well-defined project plan, creating a shopping cart app that meets the requirements of any online business shouldn't be any hassle. Through backend and frontend development a visually appealing and user-friendly interface will allow customers to interact with the shopping cart seamlessly. By adhering to a realistic timeline, which involves planning, design, backend and frontend development, testing, and deployment, we can ensure the successful completion of the project within the expected timeframe. With the integration of testing and quality assurance measures, any potential issues can be identified and resolved, guaranteeing the functionality, performance, and reliability of the shopping cart app. Finally, the deployment and launch phase enables the app to be accessible to customers, and with the use of AWS, we can leverage a robust infrastructure for reliable hosting.

Project Plan UML Online Shopping Cart – CTA 4 Diagram

In the realm of software development, designing and implementing efficient and well-structured applications is paramount. To achieve this, utilizing proper planning and visual representation of the system's components becomes crucial. This paper provides an overview of the UML diagram representing the classes used in a project, along with a detailed state machine diagram for a shopping cart application.

UML Diagram



```
Class CartConfig(AppConfig)
default_auto_field = 'django.db.models.BigAutoField'

name = 'cart'
```

```
Class Cart()
add()
delete()
update()
__len__()
__iter__()
get_total()
```

```
views.py
cart_summary()
cart_add()
cart_delete()
cart_update()
```

```
Class Category()
__str__()

get_absolute_url()

name = models.CharField(max_length=250, db_index=True)

slug = models.SlugField(max_length=250, unique=True)
```

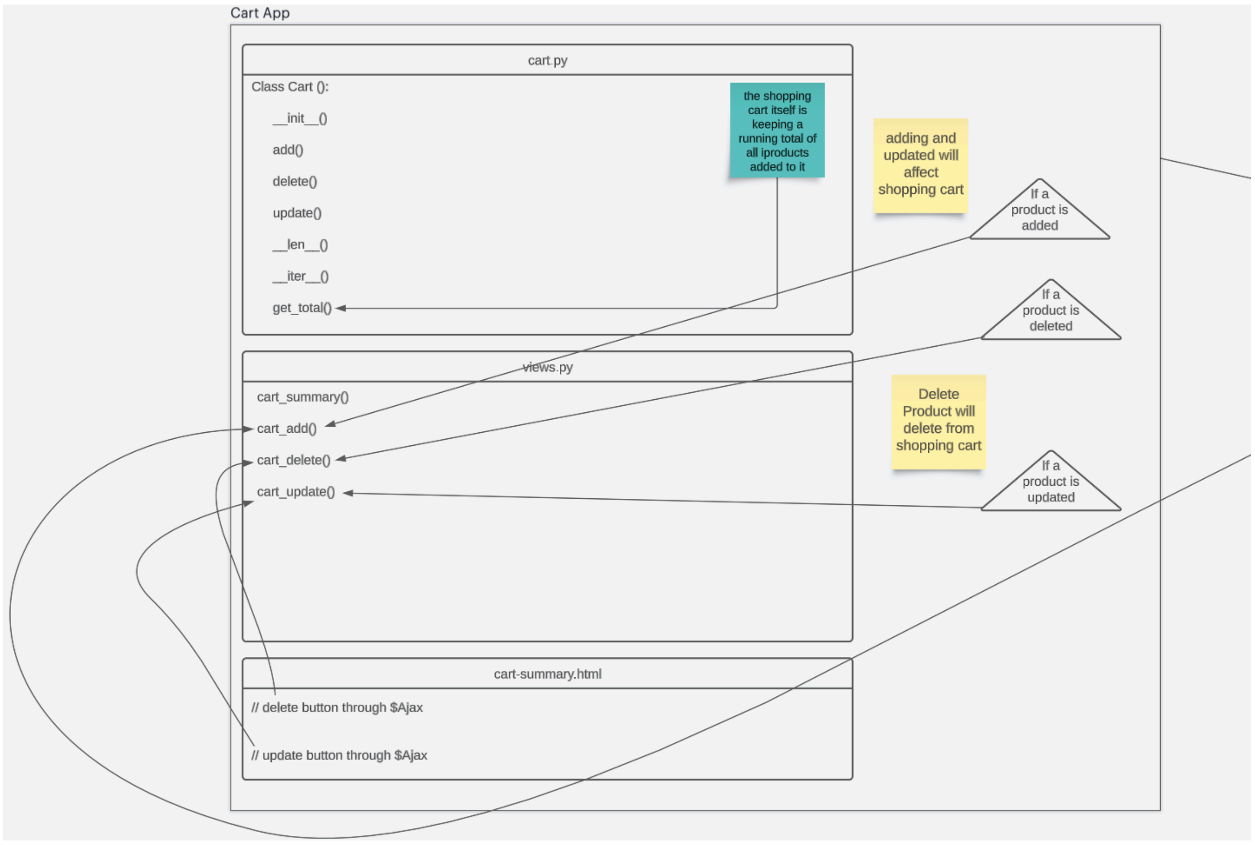
```
Class Product()
class Product()
category = models.ForeignKey(Category,
related_name='product', on_delete=models.CASCADE,
null=True)
title = models.CharField(max_length=250)
brand = models.CharField(max_length=250,
default='un-branded')
description = models.TextField(blank=True)
slug = models.SlugField(max_length=255)
price = models.DecimalField(max_digits=4, decimal_places=2)
image = models.ImageField(upload_to='images/')

__str__()
get_absolute_url()
return reverse()
```

```
views.py
store()
categories()
list_category()
product_info()
```

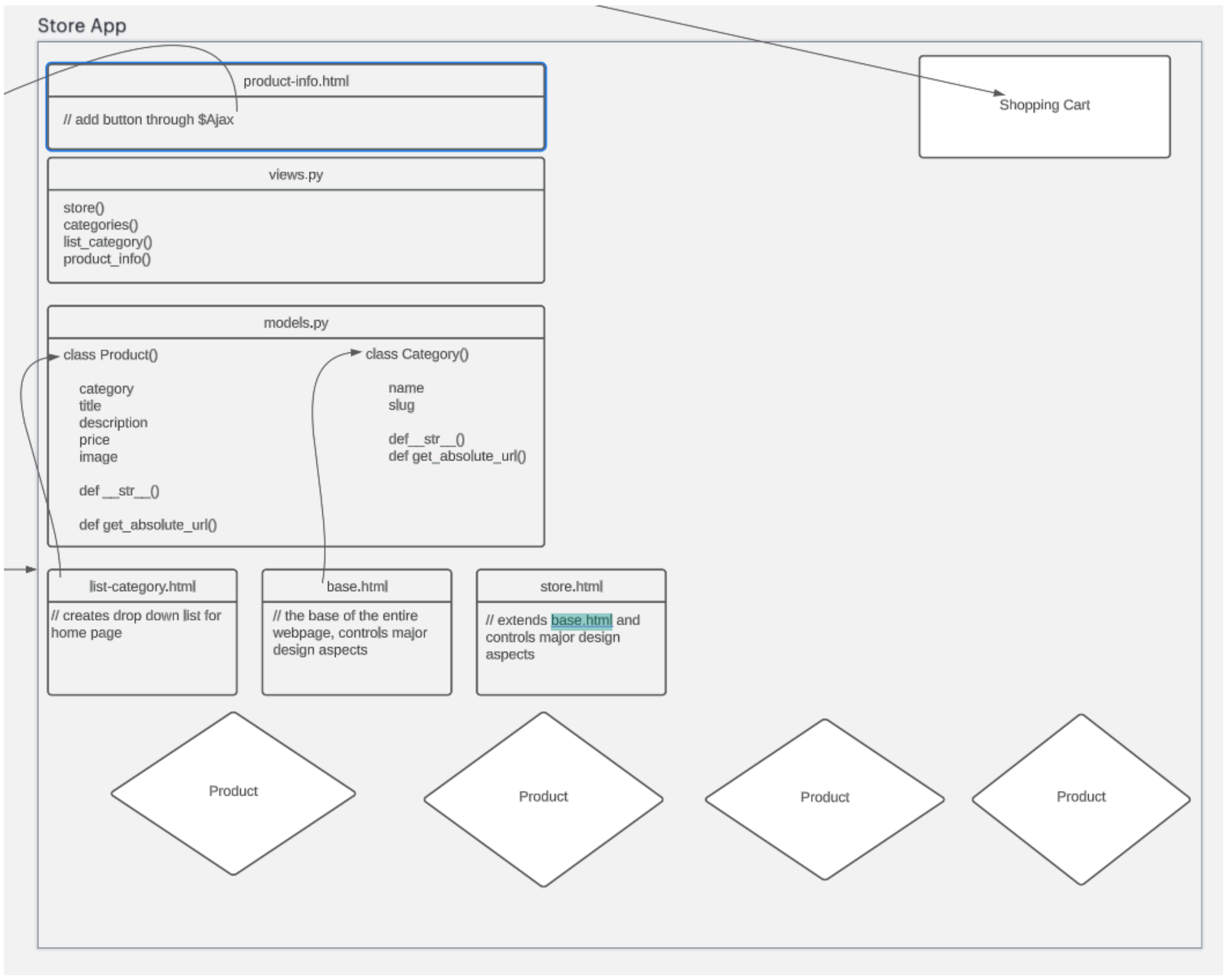
Above, we can see the UML diagram from the classes that will be used for the project. First we have the Cart app, which has the CartConfig() class and the Cart() class. Both of which are interacting with the views.py file, making it possible to update, delete, and add to the cart. On the other side we have the Category() and Product() classes that create the products themselves. The Product() class holds key attributes of the product, while Category() organizes these products into their own category of products, for example shoes, and T shirts.

State Machine Diagram – Cart App

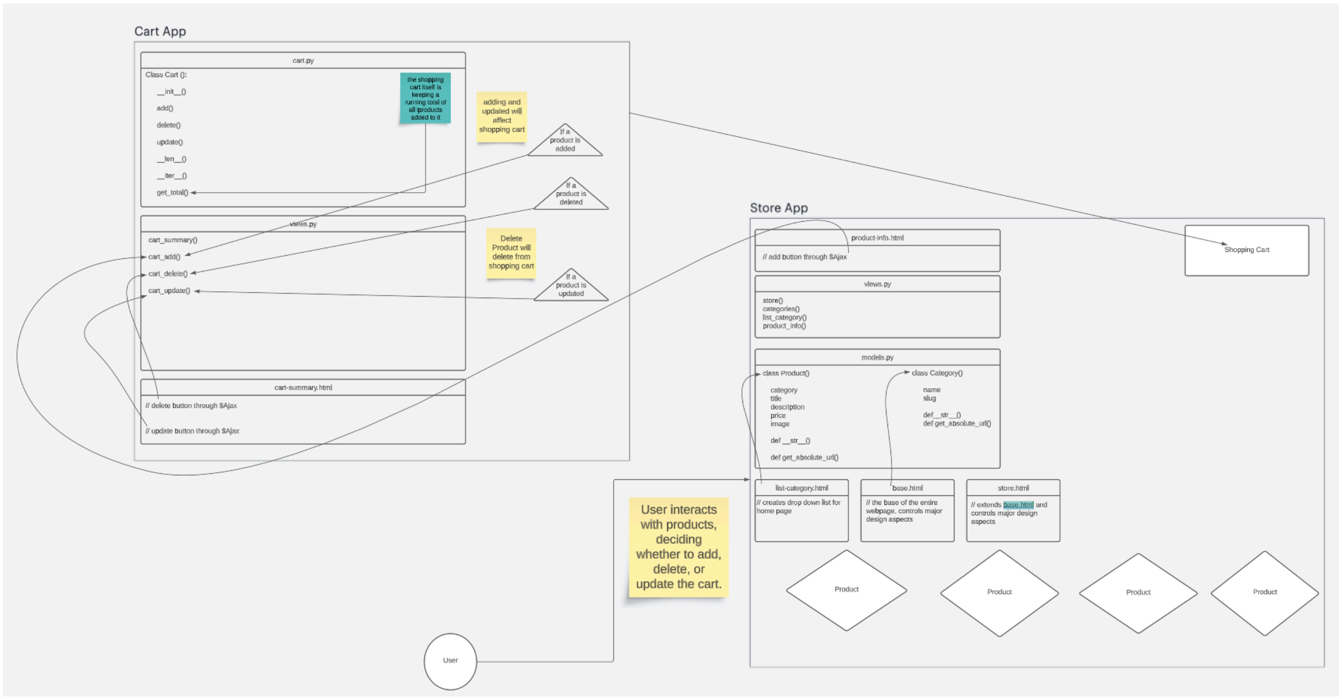


This section of the state machine diagram depicts a visual representation of the data flow between the functions and variables within the cart app. Cart.py is the class that creates, deletes, counts, and updates products site wide. Views.py is responsible for facilitating the change updates to the cart on the backend of everything, also serving as a bridge for data to travel between the frontend and the backend. Cart-summary.html is the front-end web file that creates the graphical appearance of the shopping cart summary page, just prior to check out. This is done through the use of AJAX (Asynchronous JavaScript And XML). AJAX utilizes a combination of browser built-in XMLHttpRequest object, this handles request data from a web server. JavaScript and HTML DOM handle display and of use the data (What, n.d.). The cart app makes use of many different moving parts when handling the data.

Store App



In the state machine diagram, the store app is depicted showing where different data is being allocated. In product-info.html, lies the add product button which utilizes AJAX as well. View.py in this app we have the ability to create new categories and products in our store. Usually, this is done via admin GUI, but the functions within this file are communicating to the html files as well. Models.py are the physical items themselves, along with their corresponding attributes. We have the two classes, and we see that they both interact with the other front end html files of the app. Allowing us to design certain features such as the drop down and the major design features of the webpage.



We can see here a full view of all the moving parts and pieces of the diagram. We can see how every part of this application will come together.

Conclusion Project Plan UML Online Shopping Cart – CTA 4 Diagram

In conclusion, this paper has provided an overview of the UML diagram and state machine diagram for a shopping cart project. The UML diagram illustrates the classes involved in the project, such as the CartApp, Category, Product, and their interactions. The Cart app handles cart-related functionalities, while Category and Product classes organize and define the products. The state machine diagram focuses on the data flow and interactions within the Cart app and Store app. It highlights the roles of Cart.py, Views.py, and front-end files in handling data updates and displaying the shopping cart summary page.These diagrams provide a comprehensive understanding of the project's structure and functionality, showcasing how the different components come together to deliver a seamless shopping experience. By visualizing the various moving parts, I hope you gain insights into the system's behavior and ensure effective coordination among different modules for successful project implementation.

Testing for Online Shopping Cart – CTA 6 Systems Test

So, if I'm going to create a shopping cart app, how am I going to test it? Great question, the truth is effective software testing is crucial to ensuring the functionality, reliability, and security of an online shopping cart. Testing plays a significant role in identifying defects, verifying system behavior, and validating the cart's features and functionalities. To ensure a comprehensive and systematic approach to testing, it is essential to explore different types of software testing methodologies. This paper aims to draft a document that specifies how to test the online shopping cart, considering various testing methodologies. By investigating different testing approaches, we can develop a comprehensive testing strategy that covers functional testing, performance testing, security testing, and usability testing. Through this document, we aim to establish a solid foundation for testing the shopping cart, enabling us to deliver a high-quality and user-friendly e-commerce site.

Initial Build-Up

During the initial build-up of an online shopping cart, it is crucial to ensure that the core functionality of the cart is implemented correctly and thoroughly tested. The shopping cart serves as a critical component of the e-commerce system, enabling users to add, remove, and manage their selected items before proceeding to checkout. In the creation of the shopping cart, a big focus must be on implementing features such as item selection, quantity management, and total calculation accurately. One approach to testing the functionality of the shopping cart is through modularized testing, which involves testing individual components or units of code in isolation (Author, Year). For the initial build of the shopping cart, we'll want to create a folder named cart, this folder should hold key elements to the app.

Sessions

Creating sessions in a shopping cart is a critical aspect of developing an online e-commerce platform. Sessions play a crucial role in maintaining the state of the cart and tracking user interactions (Develop, n.d). To ensure the reliability and functionality of session management in the shopping cart, thorough testing is essential. One approach to testing sessions in the cart is manual testing, which can be done to verify the behavior of sessions in real-time user interactions, ensuring that the cart maintains the correct state and user data throughout the shopping process. I as a user, you should be able to test this function by clicking on the shopping cart within the UI, and you'll be brought to a new session with everything you've added to your cart up to that point.

Adding

Adding items to a shopping cart is a fundamental functionality of an e-commerce system that requires thorough testing to ensure its reliability and seamless user experience. Testing the process of adding items to the cart involves validating the functionality, data integrity, and responsiveness of the system (Django, n.d.). One approach to test this feature is through functional testing, where different scenarios are executed to verify that items are successfully added to the cart and reflect accurately. This testing can include verifying the addition of single items, multiple items, and verifying that the correct item details, such as name, price, and quantity, are accurately captured.

Deleting and Updating

Creating a shopping cart involves implementing essential functionalities such as adding items, updating quantities, deleting items, and updating the cart's total. Testing these functionalities is crucial to ensure the cart operates as intended, providing a seamless shopping experience for users. To test the delete and update functionalities of the shopping cart, a combination of manual and automated testing techniques can be used. Manual testing involves systematically executing test scenarios where items are added to the cart and then deleted or updated. Testers should verify that the cart reflects the changes accurately and that the total is recalculated correctly. They should also validate that the cart maintains its state even when the page is refreshed or when users navigate between different pages (Django-SHOP, n.d). Automated testing can complement manual testing by providing quicker and more extensive coverage of test cases. Test automation frameworks, such as Selenium or Cypress, can be used to simulate user interactions and validate the behavior of the delete and update functionalities (Overview, 2021). Automated tests can be designed to handle various scenarios, including deleting multiple items simultaneously, updating quantities to both valid and invalid values, and verifying that the cart remains in a consistent state throughout the process (Overview, 2021). Testing the delete and update functionalities of the shopping cart is vital to ensure the cart operates flawlessly and provides a positive user experience.

Conclusion

In conclusion, the development of an online shopping cart system is crucial for enhancing the customer experience and streamlining the purchasing process in the e-commerce industry. By following a well-defined project plan, creating a shopping cart app that meets the requirements of any online business shouldn’t be any hassle. Through backend and frontend development, a visually appealing and user-friendly interface will allow customers to interact with the shopping cart seamlessly. By adhering to a realistic timeline, which involves planning, design, backend and frontend development, testing, and deployment, we can ensure the successful completion of the project within the expected timeframe. With the integration of testing and quality assurance measures, any potential issues can be identified and resolved, guaranteeing the functionality, performance, and reliability of the shopping cart app. Finally, the deployment and launch phase enables the app to be accessible to customers, and with the use of AWS, we can leverage a robust infrastructure for reliable hosting. The proposed shopping cart system is poised to meet the demands of online shoppers and contribute to increased online sales for businesses.

References

Business problems you can solve with software. (n.d.). LinkedIn. https://www.linkedin.com/pulse/business-problems-you-can-solve-software-powercodecouk?trk=pulse-article

Develop a cart application with Django. (n.d.). *Alibaba Cloud Community*. https://www.alibabacloud.com/blog/develop-a-cart-application-with-django_594697

Django-SHOP 1.2.4 documentation. (n.d.). *Django-SHOP documentation — django-SHOP 1.2.4 documentation*. https://django-shop.readthedocs.io/en/latest/reference/cart-checkout.html

Keeling, M. (2017). *Design it!: From programmer to software architect*.

Overview of test automation. (2021, December 7). Selenium. https://www.selenium.dev/documentation/test_practices/overview/

What is AJAX. (n.d.). W3Schools Online Web Tutorials. https://www.w3schools.com/whatis/whatis_ajax.asp