Estore Documentation

Ebarle, Roselle
Maglasang, Catherine
Yee, Mar Rynner
Esin, Dexter

May 18, 2016

Contents

Chapter 1

Charter

1.1 Abstract

e-store is an open-source, python-based ecommerce platform for online merchants. It provides the power to grow your web business, reach more customers and sell more products and services. It enables businesses to experience an integrated workflow for their business - Accounting, Inventory and Order Management and Customer Service under one platform.

1.2 Vision

To be the most trusted integrated e-commerce platform for merchants

1.3 Mission

To develop a platform with utmost regards in security and customer satisfaction in the e-commerce industry

1.4 Objectives

1.5 Principles

Secure Transactions Scalability and Speed

1.6 Features

1. Admin Logout
2. Admin Login
3. Customer Login
4. Customer Logout

   Customer Account

5. View my Orders

6. Re-order items
7. Update Settings

Admin Dashboard

8. Add Products
9. View Products
10. View Sales Report

Store Catalog / Store Page

11. Filter Products
12. Product Social Sharing
13. Search Product
14. View Related Products
15. Checkout Payment

1.7 Business Rules

1. When signing up, the customer can only use the alphanumeric characters and the password has a required field of 6-22 valid characters
2. The email/username is unique to every account that is recorded in the database
3. When signing up, the information inputted must be complete such as the email/username, password, contact info, and confirm password
4. When filling up the form, the password and the confirm password must be the same
5. The username and password must be recorded on the database for the customer to be logged in
6. The customer and admin has different access rights
7. The customers not logged in will not be able to checkout their cart
8. When the admin adds a product all the information of the product must be complete
9. feedback for a product is limited to only 500 characters
10. One account per store - that is the store owner
11. Admin is able to see all stores and all orders made
12. Seller is only able to see his own products and sales in dashboard
13. Seller has account balance / money obtained from sales
14. Products are shipped by merchants themselves
15. Stock qty per product can be updated
16. Once a checkout is made, the item sold appears on the sales page of the merchant, and paid status is T or F.

(a) status is set to pending

(b) merchant updates status of order (or sale) to shipped once shipped (tracking no will be placed)

(c) order status is updated to completed once item is received

(d) No returns / cancels yet

17. Checkout is an order, with unfinished status (unpaid or pending), the order items become sales.

Chapter 2

Stories

2.1 Roles

Customer - The customer browses and interacts within the store page

Admin - The admin is responsible in handling and managing orders and inventory in the admin dashboard.

2.2 Role Attributes

Customer:
Frequency of Use: Everyday
Domain Expertise: Moderate
Computer Expertise: Good
General Goals: Browse and buy products he need in the site.

Admin:
Frequency of Use: Everyday
Domain Expertise: Excellent
Computer Expertise: Excellent
General Goals: Manage orders and inventory

2.3 Persona

2.3.1 User

Akira loves shopping. Akira loves buying some rock magazines, manga, novels, gadgets, accessories and even the latest ones. Sometimes Akira is having a hard time dealing laziness because Akira doesn't like to go to a mall for shopping but he wants to buy some products from a mall. Akira found S-List. After finding S-List, Akira found out that buying products online is easier and Akira found out that there is no need to go to a mall for shopping.

2.3.2 Admin

Jane was a blogger who likes shopping. The owner of the website hired her to manage the website. Her job is to add a product, make a product be featured in the home page, add new categories for the

products, update the information of a product, and delete a product

2.4 User Stories

1. As an admin, I want to login to the website
2. As an admin, I want to be able to logout of the website
3. As a customer, I want to login to the website
4. As a customer, I want to be able to logout of the website
5. As a customer, I want to register to the website
6. As the customer, I want to be able to view the orders i've made so that I could track the status of my order.
7. As the customer, I want to view the items in my wishlist
8. As the customer, I want to view the reviews I submitted for the products I bought
9. As the customer, I want to be able to re-order products i've ordered in the past
10. As the customer, I want to update my profile settings
11. As an admin, I want to view the orders of my customers
12. As an admin, I want to add products
13. As an admin, I want to view all the products
14. As an admin, I want to view all my customers
15. As an admin, I want to view my sales report
16. As a customer, I want to filter the product by its brand and its price
17. As a customer, I want to share the product to social media sites
18. As a customer, I want to add the product to my wishlist
19. As a customer I want to search a particular product
20. As a customer, I want to add a product feedback
21. As a customer, I want to view the related products

Chapter 3

Use Cases

Use Case Element	**Description**
ID	1
Name	Get all products
Description	Get all product details from database
Primary Actors	Admin
Pre - Condition	User is logged in the system and has administrator privileges
Post - Condition	The product details are outputted in the product table in admin dashboard
Main Course	1. The user accesses the product page in admin dashboard 2. The client fetches the api resource url '/api/v1/products/' 3. The api backend returns a json to the client containing the product details 4. The client processes the json result from the api backend and outputs it in 5. the product table 6. The user sees the product and product details in the product table
Alternate Flows	
Exceptions	3a. The api returns an empty array of products 1. The client will output a message saying 'No products found' 2. The user sees the message that no products are found

ID	2
Name	Get Single Product
Description	Get the details of a single product
Primary Actors	Admin
Pre - Condition	The user is logged in and the user has administrator privileges
Post - Condition	The user should see the details of the product in the product details page
Main Course	1. The user accesses the product page 2. The user clicks on the product id he wants to view 3. The client fetches the api resource url '/api/v1/products/2' 4. The api backend returns a json to the client containing the product details 5. The client processes the json result from the api backend and outputs it in the table 6. The user sees the product details in the product detail page
Alternate Flows	
Exceptions	4a. The api backend returns no existing product 1. The client outputs a message saying that the product doesn't exist

ID	3
Name	Create a Product
Description	Create a product in inventory / catalogue
Primary Actors	Admin
Pre - Condition	The user is logged in and the user has administrator privileges
Post - Condition	The product should appear in the product table
Main Course	1. The user accesses the product page 2. The user clicks on the add product button 3. The user fills up the form in add new product page 4. The user clicks on submit 5. The client processes the form input and values 6. The client posts to the product resource url '/api/v1/products/' and adds the form inputs as a data parameter in json form 7. The api backend processes the post request from the client 8. The api saves the new product in database 9. The client redirects to the product page 10. The user sees the new product in the product table
Alternate Flows	
Exceptions	9a. The product already exists in the database 1. The api returns the error to the client 2. A message in the form should appear indicating that the product already exists

ID	4
Name	Update a Product
Description	Update a product's details in inventory / catalogue
Primary Actors	Admin
Pre - Condition	The user is logged in and the user has administrator privileges
Post - Condition	The new product details should appear in the product table
Main Course	1. The user accesses the product page 2. The user clicks on the edit button in the product row 3. The system redirects the user to the Edit Product Page 3. The user re-fills up the form and update the details as needed 4. The user clicks on submit 5. The client processes the form input and values 6. The client sends a put request to the product resource url '/api/v1/products/' and adds the form inputs as a data parameter in json form 7. The api backend processes the 'put' request from the client 8. The api updates the product in database 9. The user is redirected to the product page 10. The user sees the new product details in the product table
Alternate Flows	
Exceptions	

ID	5
Name	Add a Supplier
Description	Add a supplier for the product
Primary Actors	Admin
Pre - Condition	The supplier to be added has a name, address, phone number, fax and email.
Post - Condition	The supplier is added in the database and can be viewed on the supplier list page
Main Course	1. The admin is on the supplier page. 2. The admin Clicks on the add Supplier tab and is redirected to the add supplier form 3. The admin enters the name, address, phone number, fax, and email of the supplier 4. The Admin clicks on the submit button 5. The supplier is added and is saved in the database 6. There is a pop up message that says “the supplier was successfully added” 7. Exit
Alternate Flows	
Exceptions	4.a There is a blank input box. 1.1 The admin will redirected to the add supplier form 1.2 There is a message that tells the admin to fill in all input boxes 1.3 Back to step 4 4b: Supplier already exists 2.1 There is a pop up message that says “Supplier already exists” 2.1 Back to step 3.

ID	6
Name	Update Supplier
Description	Update Supplier Information
Primary Actors	Admin
Pre - Condition	
Post - Condition	
Main Course	1. The admin is on the supplier page. 2. The admin Clicks on the update Supplier button and is redirected to the update supplier form 3. The admin enters the name, address, phone number, fax, and email of the supplier 4. The Admin clicks on the submit button 5. The supplier is updated and is saved in the database 6. There is a pop up message that says “the supplier was successfully updated” 7. Exit
Alternate Flows	
Exceptions	4a There is a blank input box. 1.1 The admin will redirected to the add supplier form 1.2 There is a message that tells the admin to fill in all input boxes 1.3 Back to step 4

ID	7
Name	View all Supplier
Description	View all supplier on the list
Primary Actors	Admin
Pre - Condition	The admin is on the Admin dashboard page
Post - Condition	The list of all suppliers is shown
Main Course	1. The admin clicks on the supplier tab and will then be redirected to the supplier page. 2. The admin Clicks on the view Supplier tab and is redirected to the supplier list page 3. The Supplier list is shown 4. Exit
Alternate Flows	
Exceptions	

ID	8
Name	Add Order
Description	Add Customer Order
Primary Actors	Admin
Pre - Condition	The order to be added has a customer id, payment id, transaction date, shipping date and time stamp.
Post - Condition	The order is added in the database and can be viewed on the order list page
Main Course	1. The admin is on the order page. 2. The admin Clicks on the add order tab and is redirected to the add order form 3. The admin enters the customer id, payment id, transaction date, shipping date and time stamp of the order 4. The Admin clicks on the submit button 5. The order is added and is saved in the database 6. There is a pop up message that says “the order was successfully added” 7. Exit
Alternate Flows	
Exceptions	4a There is a blank input box. 1.1 The admin will redirected to the add order form 1.2 There is a message that tells the admin to fill in all input boxes 1.3 Back to step 4 4b: Order already exists 2.1 There is a pop up message that says “Order already exists” 2.1 Back to step 3.

ID	9
Name	View all Orders
Description	View all orders on the list
Primary Actors	Admin
Pre - Condition	The admin is on the Admin dashboard page
Post - Condition	The list of all orders is shown
Main Course	1. The admin clicks on the order tab and will then be redirected to the order page. 2. The admin Clicks on the view orders tab and is redirected to the supplier list page 3. The Supplier list is shown 4. Exit
Alternate Flows	
Exceptions	

ID	10
Name	Add Order Item
Description	Add an Order Item
Primary Actors	Admin
Pre - Condition	The order item to be added has an order id, product id, unit price, discount and quantity.
Post - Condition	The order item is added in the database and can be viewed on the order item list page
Main Course	1. The admin is on the order page. 2. The admin Clicks on the add order item button and is redirected to the add order item form 3. The admin enters the order id, product id, unit price, discount and quantity 4. The admin clicks on the submit button 5. The order item is added and is saved in the database 6. There is a pop up message that says “the order item was successfully added” 7. Exit
Alternate Flows	
Exceptions	4a. There is a blank input box. 1.1 The admin will redirected to the add supplier form 1.2 There is a message that tells the admin to fill in all input boxes 1.3 Back to step 4 4b: Order Item is already exists 2.1 There is a pop up message that says “Order item already exists” 2.1 Back to step 3.

ID	11
Name	Add Cart
Description	Add new Cart
Primary Actors	Admin
Pre - Condition	The cart has a session id, date created and customer id
Post - Condition	The new cart is added and saved in the database and can be viewed on the cart page
Main Course	1. The admin is on the cart page 2. The admin clicks on the add cart tab and is redirected on the add cart form page 3. The admin inputs the cart session id, date created and customer id. 4. The admin clicks on the submit button 5. The new cart is added and saved in the database and can be viewed on the cart page 6. Exit
Alternate Flows	
Exceptions	4a. There is a blank input box. 1.1 The admin will redirected to the add supplier form 1.2 There is a message that tells the admin to fill in all input boxes 1.3 Back to step 4

ID	12
Name	View All Carts
Description	View All Carts on the List
Primary Actors	Admin
Pre - Condition	The admin is on the Admin dashboard page
Post - Condition	The list of all carts is shown
Main Course	1. The admin clicks on the cart tab and will then be redirected to the cart page. 2. The admin Clicks on the view carts tab and is redirected to the cart list page 3. The cart list is shown 4. Exit
Alternate Flows	
Exceptions	

ID	13
Name	View Wishlist
Description	View Products that was added in the Wishlist
Primary Actors	Customer
Pre - Condition	The customer is logged in and is in the customer dashboard
Post - Condition	The customer will be able to view the products he/she added to wishlist
Main Course	1. The use case begins when the customer logs in the customer dashboard. 2. The user clicks on “My Wishlist” link in the navigation section. 3. The system displays all items in the wishlist of the user. 4. The user sees all items he added in wishlist. 5. The use case exits.
Alternate Flows	
Exceptions	

ID	14
Name	Add to wishlist
Description	Adding a product to the customer's wishlist
Primary Actors	Customer
Pre - Condition	The customer is logged in and is in the customer dashboard
Post - Condition	The specified product will be added to the costumer's wishlist
Main Course	1. The customer clicks the heart icon on the product page. 2. The specified product will be reflected on the customer's wishlist. 3. The use case exits.
Alternate Flows	
Exceptions	2a. If the specified product is already in the customer's wishlist 1. it shows an error message 2. Exit.

ID	15
Name	View Categories
Description	View all Categories
Primary Actors	Customer
Pre - Condition	The customer is logged in and is in the customer dashboard
Post - Condition	The customer will be able to view all categories
Main Course	1. The use case begins when the customer logs in the customer dashboard. 2. The user clicks on "Categories" link in the navigation section. 3. The system displays all categories. 4. The user sees all the categories. 5. The user clicks the category button. 6. The user sees the product/s that is under a certain category. 7. The use case exits.
Alternate Flows	
Exceptions	

ID	16
Name	Add Category
Description	Add a new Category
Primary Actors	Admin
Pre - Condition	The admin is in the admin dashboard
Post - Condition	The category will be added in the category list
Main Course	1. The admin is in the admin dashboard. 2. The admin clicks the add category button and will be displayed in the category page. 3. The admin clicks on the add button. 4. The category is added.
Alternate Flows	
Exceptions	3a. If the category exists 1. it show an error message 2. Exit.

Chapter 4

Test Cases

ID	1
Scenario	Add Attribute
Given	I have the following data
	| attribute_name | validation |
	| default | default |
When	I save the data
Then	I get a "201" response
And	I get a field "status" containing "ok"
And	I get a field "message" containing "ok"

ID	2
Scenario	Update Attribute
Given	I have a resource with the id "1"
And	I want to update its data to the following data
	| attribute_name | validation |
	| default | default |
When	I update the data
Then	I get a "200" response
And	I get a field "status" containing "ok"
And	I get a field "message" containing "ok"

ID	3
Scenario	Create Cart Item
Given	I have the following data
	| id | cart_id | product_id | quantity | time_stamp |
	| 1 | 1 | 1 | 1 | 2016-03-15 11:49:17|
When	I Post the cart item to resource url '/api/v1/carts/1/items/'
Then	I should get response '200'
And	I should get "status" 'ok'
And	I should get "message" 'OK'

ID	4
Scenario	Create duplicate cart item
Given	I have the following data
	| id | cart_id | product_id | quantity | time_stamp |
	| 1 | 1 | 1 | 1 | 2016-03-15 11:49:17|
When	I Post the cart item to resource url '/api/v1/carts/1/items/'
Then	I should get response '200'
And	I should get "status" 'ok'
And	I should get "message" 'ID EXISTS'

ID	5
Scenario	Get cart item
Given	cart item '1' is in the system
When	I retrieve the cart item '1'
Then	I should get response '200'
AND	the following cart item details are returned:
	| cart_id | product_id | quantity | time_stamp |
	| 1 | 1 | 1 | 2016-03-15 11:49:17|

ID	6
Scenario	Get a cart item that doesn't exist
Given	I retrieve the cart item '2'
When	i retrieve JSON result
Then	I should get response '200'
And	I should get "status" 'ok'
And	I should get a message containing 'No entries found'
And	it should have a field "count" 0
And	it should have an empty field " entries "

ID	7
Scenario	Create cart
Given	I have the following data
	|id | session_id | date_created| customer_id| is_active |
	|1 | 1 | 2016-03-15 | 1 | True |
When	I Post the cart to resource url '/api/v1/carts/'
Then	I should get response '200'
And	I should get "status" 'ok'
And	I should get "message" 'OK'

ID	8
Scenario	Create Duplicate Cart
Given	I have the following data
	|id | session_id | date_created| customer_id| is_active |
	|1 | 1 | 2016-03-15 | 1 | True |
When	I Post the cart to resource_url '/api/v1/carts/'
Then	I should have a status code '200'
And	I should get a status 'ok'
And	I should get a message 'ID EXISTS'

ID	9
Scenario	Get cart
Given	cart '1' is in the system
When	I retrieve the cart '1'
Then	I should have a status code '200'
And	the following cart details are returned :
	| session_id | date_created| customer_id| is_active |
	| 1 | 2016-03-15 | 1 | True |

ID	10
Scenario	Get a Cart that Doesn't Exist
Given	I retrieve the cart '2'
When	i retrieve a JSON result
Then	I should have a status code '200'
And	I should get a status 'ok'
And	it should have a field "message" 'No entries found'
And	it should have a field "count" 0
And	it should have an empty field " entries "
ID	11
Scenario	Get Customer
Given	customer id '1' is in the system
When	I retrieve the customer id '1'
Then	I get the customer '200' response
And	the following customer details are shown:
	| id | first_name | last_name | address | city | state | postal_code | country | phone |
	email | user_id | billing_address | shipping_address | date_created |
	| 1 | first1 | last1 | address1 | city1 | state1 | postalcode1 | country1 | phone1 | test@estore.com |
	1 | baddress1 | saddress1 | 2016-03-11 11:49:17 |
ID	12
Scenario	Get Customer not in the Database
Given	I access the customer url '/api/v1/customers/2/'
When	I retrieve the customer JSON result
Then	I get the customer '200' response
And	it should have a customer field 'status' containing 'ok'
And	it should have a customer field 'message' containing 'No entries found'
And	it should have a customer field 'count' containing '0'
And	it should have an empty customer field 'entries'
ID	13
Scenario	Create Customer
Given	I have the following data
	| id | first_name | last_name | address | city | state | postal_code | country | phone |
	email | user_id | billing_address | shipping_address | date_created |
	| 9 | first9 | last9 | address9 | city9 | state9 | postalcode9 | country9 | phone9 |
	test9@estore.com | 9 | baddress9 | saddress9 | 2016-03-11 11:49:17 |
When	I POST to the customer url '/api/v1/customers/'
Then	I get the create customer '201' response
And	I should get a customer field 'status' containing 'ok'
And	I should get a customer field 'message' containing 'ok'

ID	14
Scenario	Create Duplicate Customer
Given	I have the following data
	| id | first_name | last_name | address | city | state | postal_code | country | phone |
	email | user_id | billing_address | shipping_address | date_created |
	| 9 | first9 | last9 | address9 | city9 | state9 | postalcode9 | country9 | phone9 |
	test9@estore.com | 9 | baddress9 | saddress9 | 2016-03-11 11:49:17 |
When	I POST to the customer url '/api/v1/customers/'
Then	I get the create customer '201' response
And	I should get a customer field 'status' containing 'ok'
And	I should get a customer field 'message' containing 'CUSTOMER EXISTS'

ID	15
Scenario	Create Customer with Missing Details
Given	I have the following data
	| id | first_name | last_name | address | city | state | postal_code | country | phone |
	email | user_id | billing_address | shipping_address | date_created |
	| 10 | | | address9 | city9 | state9 | | country9 | phone9 | | 9 | | | 2016-03-11 11:49:17 |
When	I POST to the customer url '/api/v1/customers/'
Then	I get the create customer '201' response
And	I should get a customer field 'status' containing 'ok'
And	I should get a customer field 'message' containing 'error'

ID	16
Scenario	Add Image
Given	I have the following data
	| item_id | image_url | caption |
	| 1 | google.com | hi |
When	I save the data
Then	I get a "201" response
And	I get a field "status" containing "ok"
And	I get a field "message" containing "ok"

ID	17
Scenario	Update Image
Given	I have a resource with the id "1"
And	I want to update its data to the following data
	| image_id | item_id | image_url | caption |
	| 1 | 1 | google.com | hi |
When	I update the data
Then	I get a "200" response
And	I get a field "status" containing "ok"
And	I get a field "message" containing "ok"

ID	18
Scenario	Add Item Attribute
Given	I have the following data
	| attribute_id | item_id | attribute_value |
	| 1 | 1 | Default |
When	I save the data
Then	I get a "201" response
And	I get a field "status" containing "ok"
And	I get a field "message" containing "ok"

ID	19
Scenario	Update Item Attribute
Given	I have a resource with the id "1"
And	I want to update its data to the following data
	| attribute_id | item_id | attribute_value |
	| 1 | 1 | New Default |
When	I update the data
Then	I get a "200" response
And	I get a field "status" containing "ok"
And	I get a field "message" containing "ok"

ID	20
Scenario	Add Item Variation
Given	I have the following data
	| item_id | option_id | stock_on_hand | unit_cost | re_order_level | re_order_quantity | is_active |
	| 1 | 1 | 100.00 | 10.00 | 100.00 | 100.00 | true |
When	I save the data
Then	I get a "201" response
And	I get a field "status" containing "ok"
And	I get a field "message" containing "ok"

ID	21
Scenario	Update Item Variation
Given	I have a resource with the id "1"
And	I want to update its data to the following data
	| item_id | option_id | stock_on_hand | unit_cost | re_order_level | re_order_quantity | is_active |
	| 1 | 1 | 100.00 | 10.00 | 100.00 | 100.00 | true |
When	I update the data
Then	I get a "200" response
And	I get a field "status" containing "ok"
And	I get a field "message" containing "ok"

ID	22
Scenario	Add Item
Given	I have the following data
	| name | description | date_added | date_updated | is_active |
	| name | description | 2001-1-1 1:1:1 | 2001-1-1 1:1:1 | true |
When	I save the data
Then	I get a "201" response
And	I get a field "status" containing "ok"
And	I get a field "message" containing "ok"

ID	23
Scenario	Update Item
Given	I have a resource with the id "1"
And	I want to update its data to the following data
	| name | description | date_added | date_updated | is_active |
	| name | description | 2001-1-1 1:1:1 | 2001-1-1 1:1:1 | true |
When	I update the data
Then	I get a "200" response
And	I get a field "status" containing "ok"
And	I get a field "message" containing "ok"

ID	24
Scenario	Add Location
Given	I have the following data
	| location_name |
	| name |
When	I save the data
Then	I get a "201" response
And	I get a field "status" containing "ok"
And	I get a field "message" containing "ok"

ID	25
Scenario	Update Location
Given	I have a resource with the id "1"
And	I want to update its data to the following data
	| location_name |
	| new name |
When	I update the data
Then	I get a "200" response
And	I get a field "status" containing "ok"
And	I get a field "message" containing "ok"

ID	26
Scenario	Add Option Group
Given	I have the following data
	| option_group_name |
	| default |
When	I save the data
Then	I get a "201" response
And	I get a field "status" containing "ok"
And	I get a field "message" containing "ok"

ID	27
Scenario	Update Option Group
Given	I have a resource with the id "1"
And	I want to update its data to the following data
	| option_group_name |
	| default |
When	I update the data
Then	I get a "200" response
And	I get a field "status" containing "ok"
And	I get a field "message" containing "ok"

ID	28
Scenario	Add Option
Given	I have the following data
	| option_group_id | option_value |
	| 1 | default |
When	I save the data
Then	I get a "201" response
And	I get a field "status" containing "ok"
And	I get a field "message" containing "ok"

ID	29
Scenario	Update Option
Given	I have a resource with the id "1"
And	I want to update its data to the following data
	| option_group_id | option_value |
	| 1 | default |
When	I update the data
Then	I get a "200" response
And	I get a field "status" containing "ok"
And	I get a field "message" containing "ok"

ID	30
Scenario	Create order item
Given	I have the following data
	| id | order_id | item_id | unit_price | discount | quantity |
	| 1 | 1 | 1 | 100.0 | 0.1 | 20 |
When	I Post the order item to resource_url '/api/v1/orders/1/items/'
Then	I should have a response '200'
And	I should have a "status" containing 'ok'
And	I should have a "message" containing 'OK'

ID	31
Scenario	Create a duplicate order item
Given	I have the following data
	| id | order_id | item_id | unit_price | discount | quantity |
	| 1 | 1 | 1 | 100.0 | 0.1 | 20 |
When	I Post the order item to resource_url '/api/v1/orders/1/items/'
Then	I should have a response '200'
And	I should have a "status" containing 'ok'
And	I should have a "message" containing 'ID EXISTS'

ID	32
Scenario	Create order item with incomplete details
Given	I have the following order item details
	| id | order_id | item_id | unit_price | discount | quantity |
	| 2| 1 | | 100.00 | 0.1 | 20 |
When	I Post the order item to resource_url '/api/v1/order_items/'
Then	I should have a response '200'
And	I should have a "status" containing 'ok'
And	I should have a "message" containing 'error'

ID	33
Scenario	Get an order item
Given	order item id '1' is in the system
When	I retrieve the order item '1'
Then	I should have a response '200'
And	the following order item details are returned:
	| id | order_id | item_id | unit_price | discount | quantity |
	| 1 | 1 | 1 | 100.0 | 0.1 | 20 |

ID	34
Scenario	Get an order item that doesn't exist
Given	I retrieve the order item '2'
When	I retrieve JSON result
Then	I should have a response '200'
And	I should have a "status" containing 'ok'
And	It should have a field "message " 'No entries found'
And	It should have a field "count " 0
And	It should have an empty field " entries "

ID	35
Scenario	Create order
Given	I have the following data |id| customer_id | payment_id | transaction_date | shipping_date | time_stamp | transaction_status | total | |1 | 1 | 1 | 2016-03-11 | 2016-03-11 | 2016-03-11 11:49:17 | Pending | 100.0 |
When	I Post the order to resource_url '/api/v1/orders/'
Then	I should get a status of '200'
And	I should get a "status" 'ok'
And	I should get a "message" 'OK'

ID	36
Scenario	Create a duplicate order
Given	I have the following data |id| customer_id | payment_id | transaction_date | shipping_date | time_stamp | transaction_status | total | |1 | 1 | 1 | 2016-03-11 | 2016-03-11 | 2016-03-11 11:49:17 | Pending | 100.0 |
When	I Post the order to resource_url '/api/v1/orders/'
Then	I should get a status of '200'
And	I should get a "status" 'ok'
And	I should get a "message" 'ID EXISTS'

ID	37
Scenario	Create an order with incomplete details
Given	I have the following data |id| customer_id | payment_id | transaction_date | shipping_date | time_stamp | transaction_status | total | | 2 | 2 | 2 | 2016-03-11 | 2016-03-11 | 2016-03-11 11:49:17 | | 100.0 |
When	I Post the order to resource_url '/api/v1/orders/'
Then	I should get a status of '200'
And	I should get a "status" 'ok'
And	I should get a "message" 'error'

ID	38
Scenario	Get Order
Given	Order id '1' is in the system
When	I retrieve the order '1'
Then	I should get a status of '200'
And	the following orders are returned:
	| customer_id | payment_id | transaction_date | shipping_date | time_stamp | transaction_status | total |
	| 1 | 1 | 2016-03-11 | 2016-03-11 | 2016-03-11 11:49:17 | Pending | 100.0 |

ID	39
Scenario	Get an order that doesn't exist
Given	I retrieve the order '2'
When	I retrieve a JSON result
Then	I should get a status of '200'
3 And	I should get a "status" 'ok'
And	It should have a "message" "No entries found"
And	It should have a field "count" 0
And	It should have an empty field " entries "

ID	40
Scenario	Create Supplier
Given	I have the following data
	| id | name | address | phone | fax | email | is_active |
	| 1 | supplier1 | address1 | 221-2277| 063-221-2277| supplier1@estore.com | True |
When	I Post the supplier to resource_url '/api/v1/suppliers/'
Then	I should get a response '200'
And	I should get a "status" containing 'ok'
And	I should get a "message" containing 'OK'

ID	41
Scenario	Create duplicate supplier
Given	I have the following data
	| id | name | address | phone | fax | email | is_active |
	| 1 | supplier1 | address1 | 221-2277| 063-221-2277| supplier1@estore.com | True |
When	I Post the supplier to resource_url '/api/v1/suppliers/'
Then	I should get a response '200'
And	I should get a "status" containing 'ok'
And	I should get a "message" containing 'SUPPLIER EXISTS'

ID	42
Scenario	Create supplier with incomplete details
Given	I have the following data
	| id | name | address | phone | fax | email | is_active |
	| 2 | | | | | supplier1@estore.com | True |
When	I Post the supplier to resource_url '/api/v1/suppliers/'
Then	I should get a response '200'
And	I should get a "status" containing 'ok'
And	I should get a "message" containing 'error'

ID	43
Scenario	Get a supplier
Given	supplier '1' is in the system
When	I retrieve the supplier '1'
Then	I should get a response '200'
And	the following supplier details are returned:
	| id | name | address | phone | fax | email | is_active |
	| 1 | supplier1| address1 | 221-2277| 063-221-2277| supplier1@estore.com | True |

ID	44
Scenario	Get a supplier that doesn't exist
Given	I retrieve the supplier '2'
When	I get the JSON result
Then	I should get a response '200'
And	I should get a "status" containing 'ok'
And	It should have a field "message" 'No entries found'
And	It should have a field "count" 0
And	It should have an empty field "entries"

ID	45
Scenario	Get User
Given	user id '1' is in the system
When	I retrieve the user '1'
Then	I get the '200' response
And	the following user details are shown:
	| user_id | username | email | password | date_created | is_admin |
	| 1 | user9 | user9@estore.com | user9 | 1/1/1 1:1:1 | true |

ID	46
Scenario	Get User not in the Database
Given	I access the user id '2'
When	I retrieve the user JSON result
Then	I get the '200' response
And	it should have a user field 'status' containing 'ok'
And	it should have a user field 'message' containing 'No entries found'
And	it should have a user field 'count' containing '0'
And	it should have an empty field 'entries'

ID	47
Scenario	Create User
Given	I have the following user details: | user_id | username | email | password | date_created | is_admin | | 1 | user9 | user9@estore.com | user9 | 1/1/1 1:1:1 | true |
When	I POST to the user url '/api/v1/users/'
Then	I get the create '201' response
And	I should get a user field 'status' containing 'ok'
And	I should get a user field 'message' containing 'OK'
ID	48
Scenario	Create Duplicate User
Given	I have the following user details: | user_id | username | email | password | date_created | is_admin | | 1 | user9 | user9@estore.com | user9 | 1/1/1 1:1:1 | true |
When	I POST to the user url '/api/v1/users/'
Then	I get the create '201' response
And	I should get a user field 'status' containing 'ok'
And	I should get a user field 'message' containing 'USER EXISTS'

ID	49
Scenario	Create User with missing Details
Given	I have the following user details: | user_id | username | email | password | date_created | is_admin | | 1 | | user9@estore.com | user9 | 1/1/1 1:1:1 | true |
When	I POST to the user url '/api/v1/users/'
Then	I get the create '201' response
And	I should get a user field 'status' containing 'ok'
And	I should get a user field 'message' containing 'error'

ID	50
Scenario	Create Wishlist Item
Given	I have the details of wishlist items
	| wishlist_item_id | wishlist_id | item_id | time_stamp |
	| 2 | 1 | 3 | 2016-04-14 |
When	I POST to url '/api/v1/wishlist_items/' the wishlist items
Then	I should get status code response '200'
And	I should get 'ok' status
And	I should get 'OK' message

ID	51
Scenario	Create a duplicate wishlist item
Given	I have the details of wishlist items
	| wishlist_item_id | wishlist_id | item_id | time_stamp |
	| 2 | 1 | 3 | 2016-04-14 |
When	I POST to url '/api/v1/wishlist_items/' the wishlist items
Then	I should get status code response '200'
And	I should get 'ok' status
And	I should get 'ERROR' message for duplication

ID	52
Scenario	Create an invalid wishlist item
Given	I have the details of wishlist items
	| wishlist_item_id | wishlist_id | item_id | time_stamp |
	| d | x | t | r |
When	I POST to url '/api/v1/wishlist_items/' the wishlist items
Then	I should get status code response '200'
And	I should get 'error' status for invalid details

ID	53
Scenario	Create an incomplete wishlist
Given	I have the details of wishlist items
	| wishlist_item_id | wishlist_id | item_id | time_stamp |
	| | 1 | | 2016-04-14 |
When	I POST to url '/api/v1/wishlist/' the wishlist
Then	I should get status code response '200'
And	I should get 'ok' status
And	I should get 'ERROR' message for incomplete details

ID	54
Scenario	Get Wishlist Item
Given	wishlist item ’2’ is in the system
When	I retrieve the wishlist ’2’
Then	I should have a status code response ’200’
And	the following details are returned :
	| wishlist_item_id | wishlist_id | item_id | time_stamp |
	| 2 | 1 | 3 | 2016-04-14 |

ID	55
Scenario	Get a wishlist item that doesn’t exist
Given	I retrieve a wishlist item with id ’4’
When	I retrieve the wishlist item JSON result
Then	I should have a status code response ’200’
And	I should get the status says ’ok’
And	it should have a field message saying ’No entries found’
And	it should have a field count ’0’
And	it should have an empty field ’entries’

ID	56
Scenario	Add Wishlist
Given	I have the following data
	| wishlist_id | wishlist_name |
	| 1 | default |
When	I save the data
Then	I get a "201" response
And	I get a field "status" containing "ok"
And	I get a field "message" containing "ok"

ID	57
Scenario	Update Wishlist
Given	I have a resource with the id "1"
And	I want to update its data to the following data
	| wishlist_id | wishlist_name |
	| 1 | default |
When	I update the data
Then	I get a "200" response
And	I get a field "status" containing "ok"
And	I get a field "message" containing "ok"

Chapter 5

API Model

5.1 System Architecture

The architectural style adopted by the system is the Client-Server cloud based architecture using Jenkins Server as a medium between the Developers and the system. In the Architecture, the Developers push to git repository and Jenkins will then automatically build and activate the preset commands already entered by the developers and will then update the API server, the Database, the Front End Server, and the push notification server.

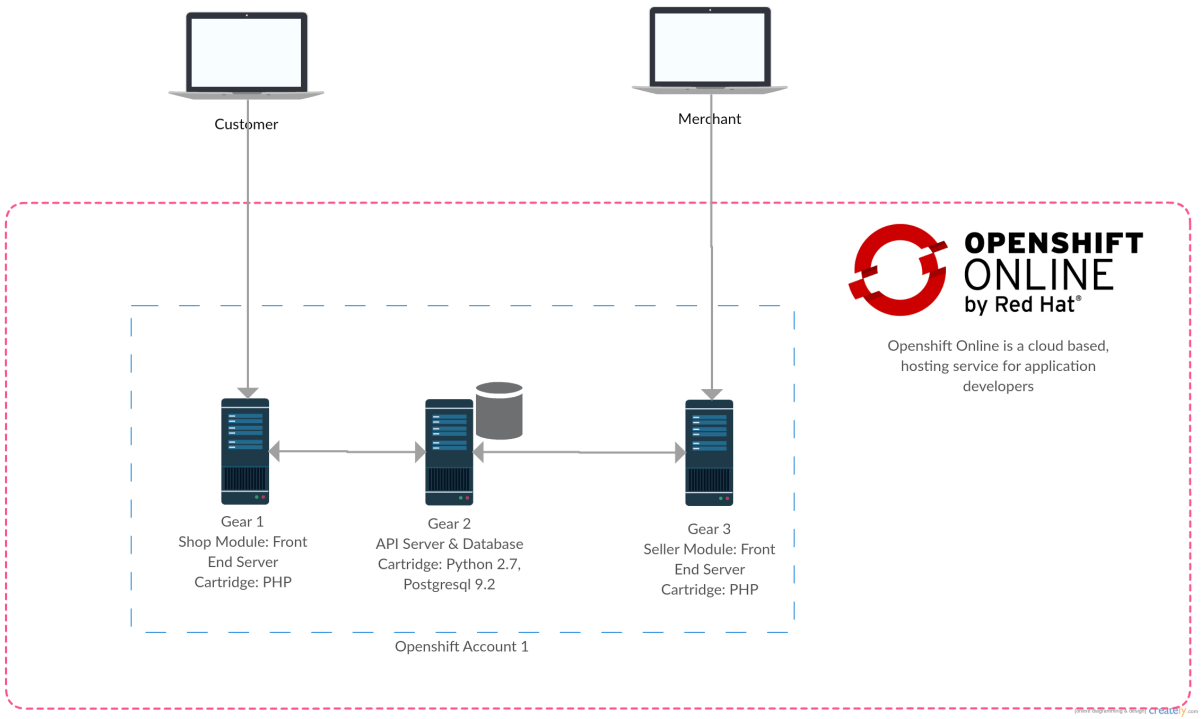

Figure 5.1: Architecture Diagram

5.2 Design Patterns

The system shall use the AngularJS framework in the client side. The design patterns used in the development of the single page application using angularjs include the following:

A Revealing Module Pattern technique is used where you expose the callable members of the service which will help you identify which members of the service are callable. An example for this is:

```
(function () {
    angular.module('estore', [
        'ui.router',                    // Routing
        'oc.lazyLoad',                  // ocLazyLoad
        'ui.bootstrap',                 // Ui Bootstrap
        'ngResource',
        'pascalprecht.translate',       // Angular Translate
        'ngIdle',                       // Idle timer
        'ngSanitize',                    // ngSanitize
        'ngCookies',
        'ngFileUpload'
    ])

})();
```

In order to export the public API and resources from the server, the module pattern will be used. The module pattern is very useful when defining services in AngularJS. Using this pattern we can simulate (and actually achieve) privacy:

```
app.factory('foo', function () {

  function privateMember() {
    //body...
  }

  function publicMember() {
    //body...
    privateMember();
    //body
  }

  return {
    publicMember: publicMember
  };
});
```

This way, once we want to inject a function inside any other component we won't be able to use the private methods, but only the public ones. This solution is extremely powerful especially when one is building a reusable library.
In implementing the REST resource access, another option would be to use the Data Mapper pattern. A data mapper is used for bidirectional transfer of data between a persistent data store and an in memory data representation. The AngularJS application will communicate with the API server, which is written in Flask(Python).
The angular $resource will help us communicate with the server and access our restful api. For example,

```
API Method:
GET /api/v1/producttypes/:id

app.factory('ProductType', function ($q) {

  function ProductType(name, description) {
    this.name = name;
    this.description = description;
  }

  ProductType.get = function (params) {
    var type = $http.get('/api/v1/producttypes/' + params.id);
```

```
    $q.all([type])
    .then(function (type) {
      return new ProductType(type.name, type.description);
    });
  };
  return ProductType;
});
```

This way we create pseudo-data mapper, which adapts our API according to the SPA requirements. We can use the User service by:

```
function MainCtrl($scope, User) {
  User.get({ id: 1 })
  .then(function (data) {
    $scope.user = data;
  });
}
```

Chapter 6

SQAP

6.1 Abstract

This document is about the Software Quality Assurance Plan (*SQAP*) of the system, e-store which is an e-commerce platform for online merchants. It provides the power to grow your web business, reach more customers and sell more products and services. It enables businesses to experience an integrated workflow for their business: Sales, Inventory and Order Management and Customer Service under one platform.

6.2 Introduction

6.2.1 Purpose

The purpose of this plan is to define the e-store Software Quality Assurance (SQA) organization, SQA tasks and responsibilities; provide reference documents and guidelines to perform the SQA activities; provide the standards, practices and conventions used in carrying out SQA activities; and provide the tools, techniques, and methodologies to support SQA activities, and SQA reporting.

6.2.2 Scope

This plan establishes the SQA activities performed throughout the life cycle of the e-store project.

This plan shall implement a project that follows the RESTful architectural style. The project shall be developed using the Flask microframework. There will be a clear separation of concerns between the client and the server for easy maintenance and scalability.

6.2.3 List of Definitions

Term	Definition
ATDD	Acceptance Test Driven Development
TDD	Test Driven Development
BDD	Behavior-Driven Development
SQA	Software Quality Assurance
UML	Unified Modeling Language
ERD	Entity Relationship Diagram
MSU-IIT	Mindanao State University - Iligan Institute of Technology
REST	Representational State Transfer
FLASK	Web Framework
Python	Programming Language
SCS	School of Computer Studies
Sales Inventory	The list of items such as the goods that are in stock
E-commerce	The buying and selling of goods over an electronic network, primarily the internet
Customer	The person who transacts in the store page of the business
Admin	The owner of the products sold in the store.
Product	The items being sold in the website
Cart	The list of items the customer is going to buy
Checkout	The process in which the customer is going to buy and pay the items inside the cart
Gherkin	Business Readable, Domain Specific Language that lets you describe software's behaviour without detailing how that behaviour is implemented.
QAM	Quality Assurance Manager
SQAP	Software Quality Assurance Plan
SQMP	Software Quality Management Plan
PM	Project Manager
CM	Configuration Manager
AD	Architectural Design
DD	Detailed Design
CI	Configuration Items
UML	Unified Modeling Language

Table 6.1: List of Definitions

6.2.4 List of References

- [*SQAP*] Software Quality Assurance Plan, SPINGRID team, TU/e, 0.1.3, June 2006
- Saleh H. (2013). Javascript Unit Testing. Packt Publishing
- Osmani A., (2012). Javascript Learning Design Patterns
- Zlobin G., (2013). Learning Python Design Patterns
- Sale D., (2014). Testing Python
- IEEE Standard for Software Quality Assurance Processes, IEEE Std 730-2014
- Clean Code Cheat Sheet
- Test-Driven Development, Dr. Christoph Steindl, Senior IT Architect and Method Exponent, Certified ScrumMaster
- Best Practices, Development Methodologies, and the Zen of Python, Valentin Haenel
- Test-Driven Development, Gary Brown
- Detailed Design, (2006). Parametric Technology Corporation (PTC)
- ESA Software Engineering Standards (ESA PSS-05-0 Issue 2), ESA Board for Software Standardization and Control (BSSC), 1991
- Configuration Items, `http://www.chambers.com.au/glossary/configuration_item.php`
- `http://flask.pocoo.org/docs/0.10/styleguide/`
- `http://explore-flask.readthedocs.org/en/latest/conventions.html`

6.3 Management

This section describes each major element of the organization that influences the quality of the software.

6.3.1 Organization

The team shall follow the agile approach, and adhere to the scrum approach in development. The team shall consist of the Scrum Master, Product Owner and the Development Team. Throughout each iteration, SQA activities should be headed by the Scrum Master who shall also serve as the Quality Assurance Manager. The team shall follow the Behaviour-driven approach in development as an extension to the Test-driven development approach. Prior to coding any functionality, the individual responsible for the feature shall create just enough acceptance tests, unit tests and code to pass the tests.

6.3.2 Tasks

The SQA team's main task is to check whether the procedures are followed and that standards are handled correctly as defined in the [SQAP]. Additionally, the SQA team inspects whether all group members fulfill their tasks according to the parts of the [SQAP] applying to their specific tasks.

Besides the described main task, the SQA team has to check the consistency and coherence between documents.

6.3.3 Responsibilities

The responsibility of Quality Assurance shall be vested in all of the members of the development team. Each one shall serve as a tester and developer at the same time. However, a Software Quality Assurance Manager (QAM) shall be the one to oversee that the BDD approach is followed, and that tests cover 100 percent coverage throughout the system, in order to avoid any bleeds or regression. The agile team shall be self-organizing individuals and take full responsibility in the feature or story assigned to them. The team shall not only be concerned with the product quality but also with the process quality and relationship between them. Should there be any major problems, the QAM shall take over and plan as needed.

6.4 Documentation

The documents to be delivered in the specific phases of the project will be based in Chapter 6, Section 5. Document standards are described in the same section. The Diagrams will be at Chapter 7 with the Storyboard.

6.5 Standards, Practices, Conventions And Metrics

6.5.1 Documentation Standards

Documentations may be in the form of a test, a docstring, or any formal document. If possible, the code should serve as enough documentation for the system. Throughout the project, PEP 8 style guide convention shall be used.

PEP 8 basically commands developers the following practices:

- Indentation: Indent with 4 real spaces (no tabs)
- Maximum line lenght: 79 characters with a soft limit for 84 if absolutely necessary. Try to avoid too nested code by cleverly placing break, continue and return statements.
- Continuing long statements: To continue a statement you can use backslashes in which case you should align the next line with the last dot or equal sign, or indent four spaces.

Docstrings

All docstrings shall be formatted with reStructuredText as understood by Sphinx. Depending on the number of lines in the docstring, they are laid out differently. If it's just one line, the closing triple quote is on the same line as the opening, otherwise the text is on the same line as the opening quote and the triple quote that closes the string on its own line:

Comments

Rules for comments are similar to docstrings. Both shall be formatted with reStructuredText. If a comment is used to document an attribute, put a colon after the opening pound sign (#).

6.5.2 Design Standards

The system shall follow the restful-architectural style. The API backend shall be built with Flask, while the frontend (running in a different port) shall be built with AngularJS. The following table describes the API model of the project:

6.5.3 API Documentation

HTTP Status Codes

Success Codes

- 200 OK - Request succeeded. Response included
- 201 Created - Resource created. URL to new resource in Location header
- 204 No Content - Request succeeded, but no response body

Error Codes

- 400 Bad Request - Could not parse request
- 401 Unauthorized - No authentication credentials provided or authentication failed
- 403 Forbidden - Authenticated user does not have access
- 404 Not Found - Resource not found
- 415 Unsupported Media Type - POST/PUT/PATCH request occurred without a application/json content type
- 422 Unprocessable Entry - A request to modify or create a resource failed due to a validation error
- 429 Too Many Requests - Request rejected due to rate limiting
- 500, 501, 502, 503, etc - An internal server error occured

All 400 series errors (400, 401, 403, etc) will be returned with a JSON object in the body and a application/json content type.
{
“status”: “error”,
"message": "Not Found"
}

6.5.4 Coding Standards

For the Python Coding standard the SQA team follows the PEP 8 – Style Guide for Python Code. For the Javascript Coding Standard the SQA teams follow the Opinionated Angular Style Guide for Teams by `https://twitter.com/john_papa`.
The PEP 8 – Style GUide for Python Code is what most python programmers use. You can find its documentation at `https://www.python.org/dev/peps/pep-0008/`. Some examples for The Angular Style Guide:

1. Single Responsibility
    - Rule of 1 - Define 1 component per file, recommended to be less than 400 lines of code.
    - Small Functions - Define small functions, no more than 75 LOC (less is better).
2. IIFE
    - JavaScript Scopes - Wrap Angular components in an Immediately Invoked Function Expression (IIFE).
3. Modules
    - Avoid Naming Collisions - Use unique naming conventions with separators for sub-modules.
    - Definitions (aka Setters) - Declare modules without a variable using the setter syntax.
    - Getters - When using a module, avoid using a variable and instead use chaining with the getter syntax.

More can be found at `https://github.com/johnpapa/angular-styleguide/tree/master/a1`

6.5.5 Testing Standards

The SQA team will follow the lettuce BDD and Unit testing standard using the Gherkin syntax. Like Python, Gherkin is a line-oriented to define structure. Line endings terminate statements (eg, steps). Either spaces or tabs may be used for indentation (but spaces are more portable). Most lines start with a keyword(eg, Given, When, Then).
For the UI User tests the SQA team follow the System Usability Scale(SUS) where there are 10 questions and the users write a check in the checkbox on the scale, after which the SQA team will get the mean from the 20 testers and calculate for the Usability Score (More on this at Chapter 8).

6.5.6 Metrics

The SQA team will measure the quality of the delivered software, during the monthly checkups, by means of metrics.
The metrics the SQA team will follow are as follows:

- The amount of useful commentary must exceed 1/5 of the total size of the code.
- Maximum Depth of nested if-statements should be less than 4.
-

If there are any violations on any of these metrics are found then they must be resolved, unless the PM and QAM grants a permit

6.5.7 Compliance Monitoring

The SQA team will randomly check during which the references to other documents are checked. The referenced documents must include the author and a url link if possible. Any Updates of the software will also be tracked by the way of commits in a git repository.

6.5.8 Pentesting Plan

6.5.9 Event Response Chart

Form Control	Event	Response
Website	Page Loads	
Sign Up Button	Click	Display Sign Up Form
User Name	Field Receives Focus	Place Cursor in User Name Field
Email	Field Receives Focus	Place Cursor in Email Field.
	Field Loses Focus	Verify if Email is Valid
Password	Field Receives Focus	Place Cursor in Password Field
Confirm Password	Field Receives Focus	Place Cursor in Confirm Password Field.
	Field Loses Focus	Verify if input in Confirm Password Field is the same as Password Field
Create button	Click	Verify Validity of information inside the fields. Post to the Resource '/api/v1/users/'. Display Sign In Page
Already Have an Account Link	Click	Display Sign Up Form
Forgot Your Password Link	Click	Display Forgot Password Form
Enter Email	Field Receives Focus	Place Cursor in Enter Email Field
Next Button	Click	Verify Email. Display Confirm page
No Button	Click	Display Forget Password Form
Yes Button	Click	Send Password to Given Email. Diplay Password Sent Page

Back to Login Button	Click	Display Sign In Form
Sign In Button		
Login Button	Click	Verify Fields. Log User in. Display Ladning Page
Shopping Cart Icon	Click	Verify if logged in. Display Shopping Cart Page
Continue Shopping Button	Click	Display Landing Page
Product Image	Click	Display Product Page
Browse Button	Click	Display Landing Page
Add to Cart Button	Click	POST to the Resource '/api/v1/carts/'
Catalogue Button	Click	Display Catalogue Page
5 Stars Icon	Click	Record Number of Stars Clicked
Category Buttons	Click	Display Target Category
Search Box	Field Receives Focus	Place Cursor in Search Box Field
Search Button	Click	Find All Products with the same Characters on Field then Show Products
Dashboard Button	Click	Verify if User is Admin. Load Dashboard Page
Suppliers Button	Click	Display Supplier Page. GET Resource '/api/v1/suppliers/'
Customers Button	Click	Display Customer Page. GET Resource '/api/v1/customers/'

Orders Button	Click	Display Orders Page
Add New Suppliers Button	Click	Display Supplier Form.
Name	Field Receives Focus	Place Cursor in Name Field
Address	Field Receives Focus	Place Cursor in Address Field
Phone Number	Field Receives Focus	Place Cursor in Phone Number Field
Fax	Field Receives Focus	Place Cursor in Fax Field
Is Active	Field Receives Focus	Place Cursor on Is ACtive Checkbox
Add Button	Click	Verify the Fields. POST to a Resource uri. Display previous Page
Cancel Button	Click	Display previous Page
E-store Logo	Click	Display Dashboard Page
Add New Order Button	Click	Display Order Form
Customer ID	Field Receives Focus	Place Cursor in Customer ID field
Payment ID	Field Receives Focus	Place Cursor in Payment ID field
Transaction Date	Field Receives Focus	Place Cursor in Transaction Date field
Shipping Date	Field Receives Focus	Place Cursor in Shipping Date field
Time Stamp	Option Receives Focus	Place Cursor in Time Stamp option
Transaction Status	Field Receives Focus	Place Cursor in Transaction Status Field
Total	Field Receives Focus	Place Cursor in Total Field

Unit Price	Field Receives Focus	Place Cursor in Unit Price field
Discount	Field Receives Focus	Place Cursor in Discount field
Quantity	Field Receives Focus	Place Cursor in Quantity field
Back to Store Button	Click	Display Landing Page
Account Button	Click	Verify if logged in. Display Account Page
Change Password Button	Click	Display Change Password form
Enter Password	Field Receives Focus	Place Cursor in Enter Password Field
Enter New Password	Field Receives Focus	Place Cursor in Enter New Password Field
Verify New Password	Field Receives Focus	Place Cursor in Verify New Password Field
	Field Loses Focus	Verfiy that Enter New Password Field and Verify New Password Field are the same
Confirm Button	Click	Verify Fields. PUT to Resource '/api/v1/users/'. Display Success Page
Click Here to go Back to Account Button	Click	Display Account Page
Wishlist Button	Click	Display Wishlist Page
Heart Icon	Click	PUT to Resource '/api/v1/wishlist/'

Social Icons	Click	Display Target Social Website
Checkout Button	Click	Display Checkout Form
Remove Button	Click	Product is Removed from Shopping Bag
First Name	Field Receives Focus	Place Cursor in First Name Field
Last Name	Field Receives Focus	Place Cursor in Last Name Field
City	Field Receives Focus	Place Cursor in City Field
State	Field Receives Focus	Place Cursor in State Field
Postal Code	Field Receives Focus	Place Cursor in Postal Code Field
Country	Field Receives Focus	Place Cursor in Country Field
Billing Address	Field Receives Focus	Place Cursor in Billing Address Field
Shipping Address	Field Receives Focus	Place Cursor in Shipping Address Field
Date Created	Field Receives Focus	Place Cursor in Date Created Field
Next Button	Click	Display Paypal Page
Continue Button	Click	Display Order Information

6.6 Review

The SQA team checks all the project and product documents on a random date once a month to make sure that all the documents adhere to the documentation standards. The SQA team also checks if the Code adheres to the coding standards. If there are problems discovered, the problem must be solved within the day after which the document will be checked again.

6.7 Test

For the tests, the SQA team will test the API's, Javascript, and the User Interface. For the API and Javascript test refer to the Test Cases in Chapter 4 and for the User Interface Test Refer to Chapter 8.

6.8 Problem reporting and corrective actions

If problems arise in either the Documentation or the System they need to be resolved.
Examples of such Problems are:

Document Problems

- What is written and what is implemented in the system are different
- Errors

- Incompleteness
- Non compliance with the Documenting Standards

Code Problems

- Lack of functionality
- Wrong functionality
- Useless or Excess Code
- Non compliance with the coding and commentary standards

Problem Reporting Procedure

When a problem is detected, the person who discovered the error is responsible for reporting it to the PM and QAM. When a problem is discovered during a review, the member of the SQA team present is responsible.

Problem Solving Procedure

- The SQA team appoints the member who is responsible for that task. He/She is responsible for solving the problem
- When the problem is resovled the SQA team will then be notified and they will check whether the changes solved the problem
- If the problem cannot be solved or cannot be solved within a given amount of time, the appointed member will consult with the other team members. In the consultation they will decide on what to do with the problem

6.8.1 Changes in requirements of the customer

It is also possible that the requirements of the customer change. In this case, the requested change is matched to the Charter. If the change conforms to the Charter it is accepted. If it does not conform to the Charter, the team decides wether it will discard the changes or not.

6.9 Tools, techniques and methods

The SQA team has to make sure that appropriate tools, techniques and methods are used. The tools that the team uses are:

1. PyCharm
2. PgAdmin 3
3. Webstorm

These tools are readily available at `https://www.jetbrains.com/`

6.10 Code Control

It is the SQA teams responsibility to assure the correct handling of the code and the documents.
The following has to be valid:

- People are free to distribute and use the documents.
- All versions of the Documentation and the early versions of the Source Code is freely available at the project repository at `https://github.com/catherinemaglasang/e-store-by-scorpio`
- The Latest Source Code is only available to those authorized by the SQA team.

6.11 Media Control

The SQA team will check whether all the procedures and the techniques used are handled properly.

6.12 Supplier Control

All external software components in the program code, that have an unreliable source, will be tested according to the [ESA] standards. Software components that have reliable sources will undergo some quick tests. These tests will be focused on the parts of this software that are of importance to the project.

6.13 Training

The project requires sufficient skill in Python, Flask, and front end technologies like Angular, jQuery and Ajax. The learning curve throughout the project development has been steep and required training from the Advisor and co-team members.

6.14 Risk management

6.14.1 Categories of risks

The following are categories of risks that are relevant to the project:

Risks with respect to the work to be done

1. Miscommunication
Probability: High
Prevention: Daily stand ups or quick huddle should be done by the team on a regular basis. Major weekly meetings are done to keep address pressing issues in development. Team members should not hesitate to ask and re ask questions if things are unclear in order to avoid bottlenecks in the progress of the system. With regards to the customer, bi-monthly face-to-face meetups should be done to update and keep track of progress. If any confusions arise, the team may opt to use other communication mediums like phone calls, or emails to clear up problems.
Correction: When it becomes clear that miscommunication is causing problems, the team members involved and the customer are gathered in a meeting to clear things up.
Impact: High

Resource	URI	HTTP Method	Description
Inventory Module			
Item	/api/v1/items/	GET	Retrieve all items
	/api/v1/items/:id	GET	Retrieve single item
	/api/v1/items/:id	PUT	Update item
	/api/v1/items/	POST	Create new item
Type	/api/v1/types/	GET	Retrieve all item types
	/api/v1/types/:id	GET	Retrieve all types
	/api/v1/types/:id	PUT	Update type
	/api/v1/types/	POST	Create new type
Attribute	/api/v1/types/:id/attributes/	GET	Retrieve all attributes under the type id
	/api/v1/types/:id/attributes/:id/	GET	Retrieve single attribute under the type id
	/api/v1/types/:id/attributes/	POST	Create new attribute under the type id
	/api/v1/types/:id/attributes/:id/	PUT	Update attribute details under the type id
AttributeValue	/api/v1/items/:id/attributes/	GET	Retrieve all type-attribute pair values for each item assigned to a particular type
	/api/v1/items/:id/attributes/:id/	GET	Retrieve a single type-attribute pair value for an item
	/api/v1/items/:id/attributes/:id/	PUT	Update the type-attribute pair value
	/api/v1/items/:id/attributes/	POST	Create new attribute value based on the type assigned to an item
Image	/api/v1/items/:id/images/	GET	Retrieve all images for a single item
	/api/v1/items/:id/images/:id/	GET	Retrieve single image for an item
	/api/v1/items/:id/images/:id/	PUT	Update image for an item
	/api/v1/items/:id/images/	POST	Create new image for an item
Supplier	/api/v1/suppliers/	GET	Retrieve all suppliers
	/api/v1/suppliers/:id/	GET	Retrieve single supplier
	/api/v1/suppliers/:id/	PUT	Update a supplier
	/api/v1/suppliers/	POST	Create new supplier
Cart & POS Module			
Cart	/api/v1/carts/	POST	Create new cart instance
	/api/v1/carts/:id/	PUT	Update Cart
CartItem	/api/v1/carts/:id/items/	GET	Retrieve all cart items
	/api/v1/carts/:id/items/:id/	GET	Retrieve single cart item
	/api/v1/carts/:id/items/:id/	PUT	Update single cart item
	/api/v1/carts/:id/items/	POST	Add an item to cart
Order	/api/v1/orders/	GET	Retrieve all orders
	/api/v1/orders/:id/	GET	Retrieve single order
	/api/v1/orders/:id/	PUT	Update single order
	/api/v1/orders/	POST	Create new order
OrderItem	/api/v1/orders/:id/items/	GET	Retrieve all order items
	/api/v1/orders/:id/items/:id/	GET	Retrieve single order item
	/api/v1/orders/:id/items/:id/	PUT	Update single order item
	/api/v1/orders/:id/items/	POST	Add new item in order
Wishlist	/api/v1/wishlists/	GET	Retrieve all created wishlists
	/api/v1/wishlists/:id/	GET	Retrieve single wishlist
	/api/v1/wishlists/:id/	PUT	Update wishlist
	/api/v1/wishlists/	POST	Create new wishlist
WishlistItem	/api/v1/wishlists/:id/items/	GET	Retrieve all items under a single wishlist
	/api/v1/wishlists/:id/items/:id/	GET	Retrieve single wishlist item
	/api/v1/wishlists/:id/items/:id/	PUT	Update wishlist item
	/api/v1/wishlists/:id/items/	POST	Add new item in wishlist
User	/api/v1/users/	GET	Retrieve all users
	/api/v1/users/:id/	GET	Retrieve single user
	/api/v1/users/:id/	PUT	Update user
	/api/v1/users/	POST	Create new user
Group	/api/v1/groups/	GET	Retrieve all groups
	/api/v1/groups/:id/	GET	Retrieve single group
	/api/v1/groups/:id/	PUT	Update group
	/api/v1/groups/	POST	Create new group
	/api/v1/groups/:id/users/	GET	Retrieve all users under a group
	/api/v1/groups/:id/users/:id/	GET	Retrieve a user under a group
	/api/v1/groups/:id/users/:id/	PUT	Update a user in a group (e.g. Permission)
	/api/v1/groups/:id/users/	POST	Add a user to a group
Customer	/api/v1/site/:id/customers/	GET	Retrieve all customers in a site
	/api/v1/site/:id/customers/:id/	GET	Retrieve single customer in a site
	/api/v1/site/:id/customers/:id/	PUT	Update customer in a site
	/api/v1/site/:id/customers/	POST	Register new customer in site or business

Table 6.2: REST API Model

Chapter 7

Diagrams

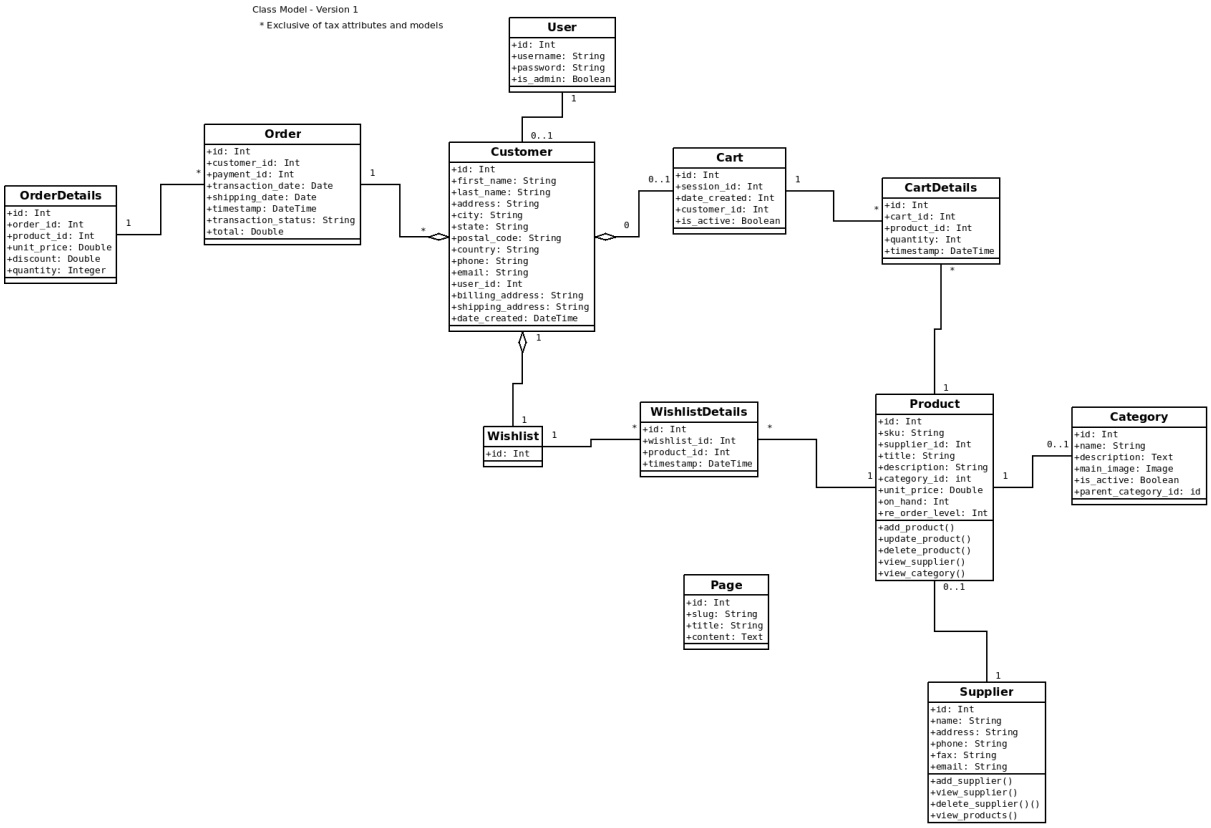

Figure 7.1: Class Diagram

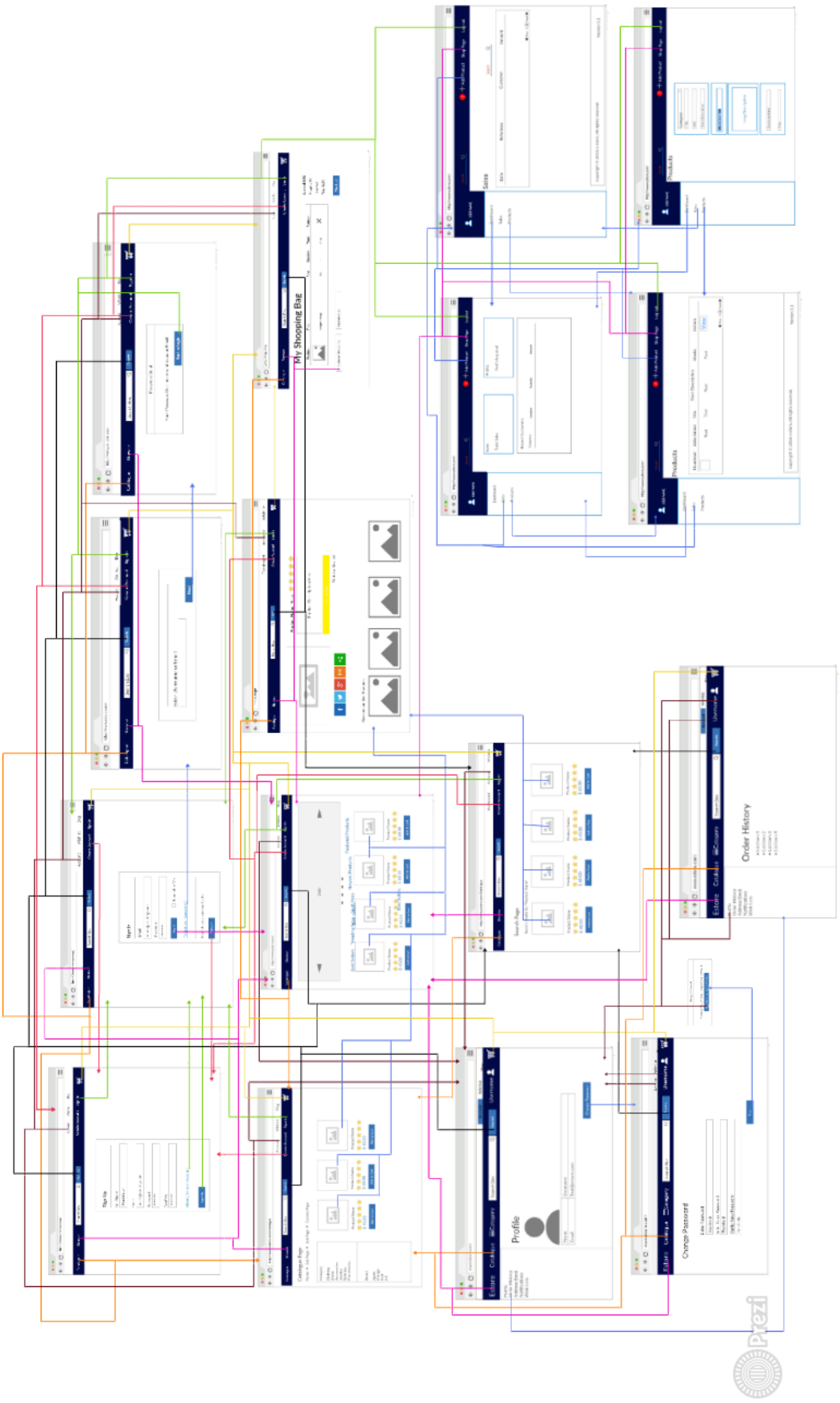

Figure 7.2: Storyboard (Clearer version found at `https://prezi.com/gzvwpix9o2mr/e-store-storyboard/`)

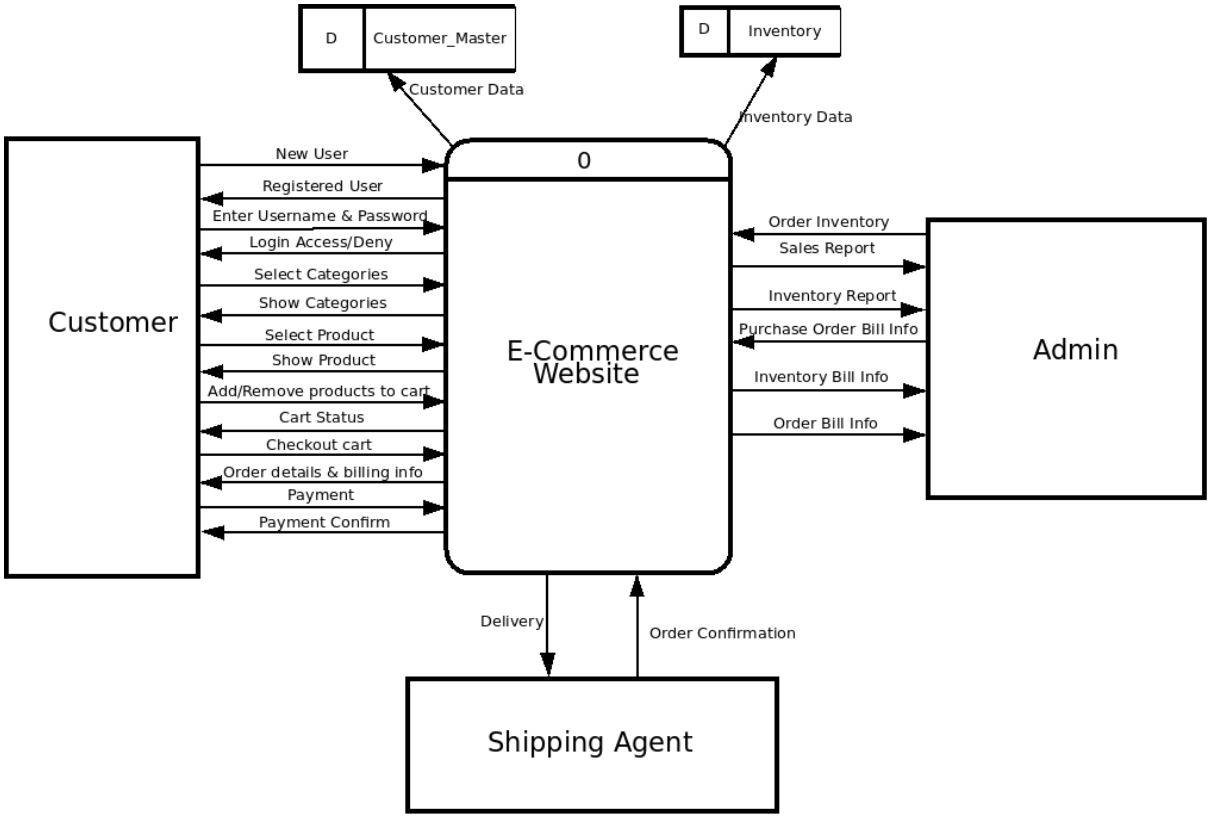

Figure 7.3: Data Flow Diagram

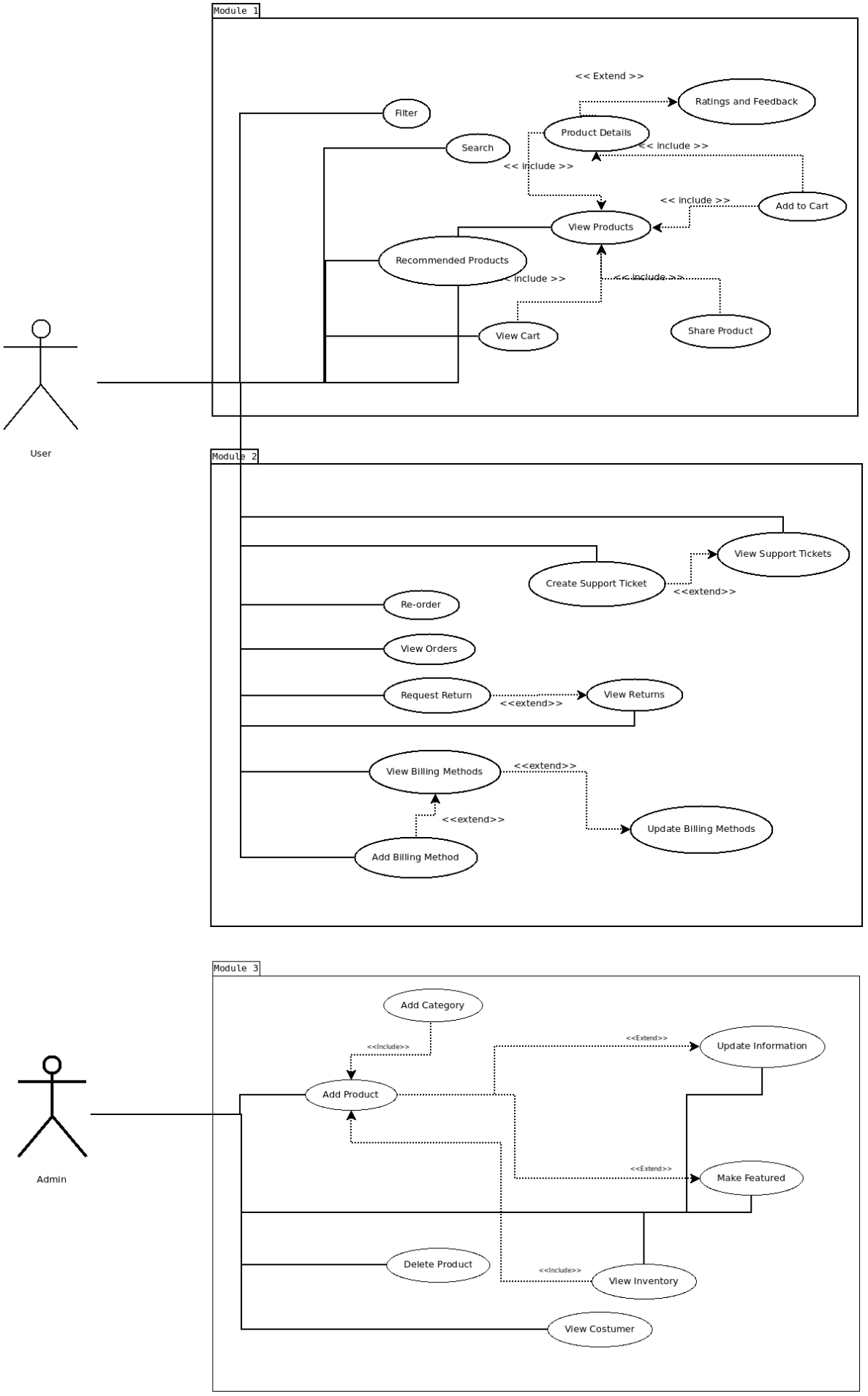

Figure 7.4: Use Case Diagram

	Activity	Planned Duration (D)	Actual Duration
A	Planning and Design	14	46
B	Authentication & Access	14	
C	Inventory & Catalogue Feature	18	
D	Orders & Sales	14	
E	Cart	7	
F	Wishlist	7	
G	Payment Processor/Handling	7	
H	CMS	7	
I	Reports & Analytics	7	
J	Promotions & Offers	7	

Workflow:
1. BDD and Unit Tests
2. API Development
3. Front End Development
4. Integration for api and front end
5. Deployment (Iteration Release)

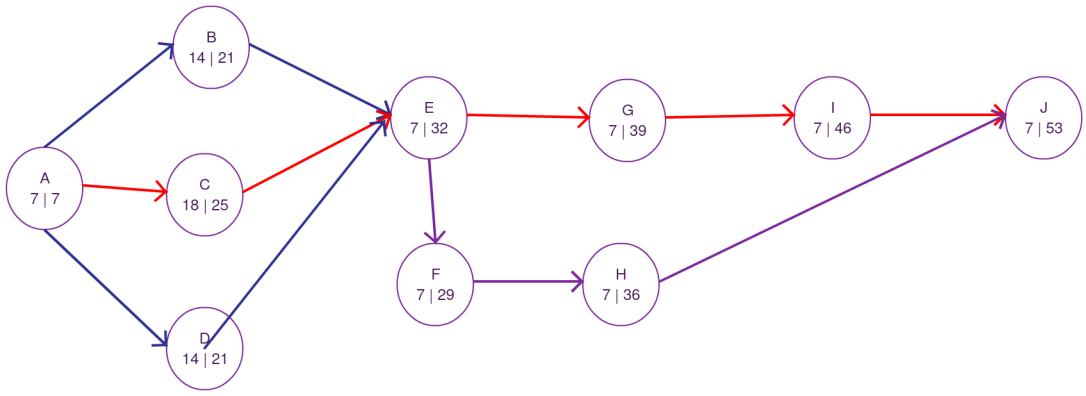

Figure 7.5: Pert Chart

Chapter 8

User Testing Results

8.1 System Usability Scale

	Strongly Disagree				Strongly Agree	Scale Position	Calculation	Score Contribution
1. I think that I would like to use this system frequently.	⊡	⊡	⊡	✓	⊡	4	$4-1$	3
2. I found the system unnecessarily complex.	⊡	⊡	✓	⊡	⊡	3	$5-3$	2
3. I thought the system was easy to use.	⊡	⊡	⊡	✓	⊡	4	$4-1$	3
4. I think that I would need the support of a technical person to be able to use this system.	⊡	✓	⊡	⊡	⊡	2	$5-2$	3
5. I found the various functions in this system were well integrated.	⊡	⊡	⊡	✓	⊡	4	$4-1$	3
6. I thought there was too much inconsistency in this system.	⊡	⊡	✓	⊡	⊡	3	$5-3$	2
7. I would imagine that most people would learn to use this system very quickly.	⊡	⊡	⊡	✓	⊡	4	$4-1$	3
8. I found the system very cumbersome to use.	⊡	⊡	✓	⊡	⊡	3	$5-3$	2
9. I felt very confident using the system.	⊡	⊡	⊡	✓	⊡	4	$4-1$	3
10. I needed to learn a lot of things before I could get going with this system.	⊡	⊡	✓	⊡	⊡	3	$5-3$	2

Total = 26, $Score = Total * 2.5 = 26 * 2.5 = 65$

8.2 Discussion

We have interviewed 20 users and gave them the above 10 SUS Questions and rate them from 1(strongly disgree) to 5(strongly agree) to test the usability of our system, and few scored the same. For each question, we got the scales by getting the mean of the 20 users and recorded them on the scoring form.

In question 1, we got a scale of 4 which means that most of them likes to use the system while in question 2 we got 3 which means some of them finds the system unnecessarily complex and some finds it necessarily complex . When asked if the system is easy to use most of them answered yes and we got a high score of 4 which then implies the answer to question 4 giving us a score of 2 that they don't need a technical person to use the system. According to them the functions in the system are well integrated giving us the score 4. In question 6, we got a scale of 3 which means that the system is not too consistent and not too inconsistent, it has inconsistency in terms of the design (e.g. form). As what they've experienced in using the system they believe that most people would learn to use the system quickly like they used it. In question 8, if the system is very cumbersome to use, we got a scale of 3 which means that it is not too slow nor too fast. In question 9, we got a scale of 4, which means they are confident to use the system, that they are not afraid clicking any buttons in the system but they still need to learn some things in the system.

After analyzing their feedbacks and answers, we then do the way SUS scored it and we got the Score of 65, converting it into a letter grade which gives us a C-. Having a Score of 65 means our Usability is below average since the average SUS score from all 500 studies is 68.

The way SUS scores is:

- For odd items: subtract one from the user response.
- For even-numbered items: subtract the user responses from 5
- This scales all values from 0 to 4 (with four being the most positive response).
- Add up the converted responses for each user and multiply that total by 2.5. This converts the range of possible values from 0 to 100 instead of from 0 to 40.

The table below shows the percentile ranks for a range of scores and how to associate a letter grade to the SUS score.

Grade	SUS	Percentile Rank
A^+	90	99%
A	82	93%
A^-	80	88%
B^+	78	83%
B	75	73%
B^-	73	67%
C^+	72	63%
C	68	50%
C^-	63	36%
D	55	20%
F	50	13%

8.3 Conclusion

After we get the results on how usable our system is through the use of System Usability Scale method we have discovered some weaknesses like inconsistency and inefficiency and strengths like manageability and easy to use that our system possessed. Having a score of 65 implies that our system really needs improvement. Given those weaknesses, our team knows now where to focus and what features do the users give their attention most. And since our system is an e-commerce platform for online merchants one of the most important factor that we have to consider based on the feedbacks of our users is first the interface design, whether it is eye-catching to the users or it's simple and easy to use and second is the functionality and efficiency of the system. With the help of the SUS method our team knows now how to improve our system and get a higher grade than 65.

Chapter 9

Questionnaire

9.1 Client Questionnaire

1. What kind of business is your company in?
2. What specific industries do you cater to?
3. On average, how many employees and departments do you have in your company/business?
4. How many people at your company/business will be involved with this project?
5. How did you manage your products or inventory, manually or you're using some applications(e.g. spreadsheets)?
6. What benefits do you expect from this project?
7. What keywords do you want people to use in search engines to find your site?
8. What are your design requirements?
9. What kinds of products will you be selling?
10. Roughly how many products will be listed on the site?
11. How would you like to organize your product?
12. Do you have product descriptions available?
13. Do you have high quality photos available for each product?
14. Do you want to allow product reviews or ratings?
15. Do you want to add social sharing icons to product pages?
16. Do you want the web site to track inventory?
17. Do you want to build an email list of customers for promotional purposes?
18. Do you offer quantity discounts?
19. Do you want to offer coupons?
20. Do you want to offer wish lists?
21. Do you have terms of service and refund policies in place for the site?

22. Would you want to integrate online payment methods in the system so that your customers will be able to pay online? (i.e. Credit Cards, Paypal)
23. Do you have any suggested features you'd like to be implemented in the system?
24. Does this project have a deadline?
25. What is your budget for this project? Are you willing to pay?
26. Do you have anything to add?

Chapter 10

References

https://bocoup.com/weblog/documenting-your-api
https://gist.github.com/iros/3426278
https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html
https://sourcemaking.com/design_patterns/flyweight
http://blog.rackspace.com/4-reference-architectures-to-optimize-your-ecommerce/
https://github.com/johnpapa/angular-styleguide/tree/master/a1
https://github.com/cucumber/cucumber/wiki/Gherkin
http://www.measuringu.com/sus.php
http://www.usabilitybok.org/sus