Team Amazonian Prime

Test Plan

April. 4, 2023
Version 2.0

Document Information

Revision History

Date	Version	Status	Prepared by	Comments
March 8, 2023	1.0		Amazonian Prime	
April 3, 2023	2.0	Complete	Amazonian Prime	Revisions for final submission: ● Updated our final test statuses ● Added (DESCOPED) tag to components that were descoped ○ Delete order as admin

Document Control

Role	Name	E-mail	Telephone
Professor	Jerry Jim		
TA	Marie Salomon		
TA	Shijun Shen		
Project Sponsor	Peter Smith		
	Mahmoud Al Khatib	mahmoudalkhatib.ubc@gmail.com	
	Michael He	michaelhe17@gmail.com	
	Joshua Luong	joshualuong@hotmail.com	
	Tristan Martinuson	tmartinuson@gmail.com	
	Elaine Shi	elaineshi328@gmail.com	
	William Suryawidjaja	suryawidjajaw@gmail.com	

Approval

Role	Name	Signature	Sign-off Date
Amazonian Prime Member	Mahmoud Al Khatib	Mahmoud Al Khatib	April 4, 2023
Amazonian Prime Member	Michael He	Michael He	April 4, 2023
Amazonian Prime Member	Joshua Luong	Joshua Luong	April 4, 2023
Amazonian Prime Member	Tristan Martinuson	Tristan Martinuson	April 4, 2023
Amazonian Prime Member	Elaine Shi	Elaine Shi	April 4, 2023
Amazonian Prime Member	William Suryawidjaja	William Suryawidjaja	April 4, 2023

Table of Contents

Overall Testing Approach

Overview
The following document details a proposed testing plan to be conducted in association with Amazonian Prime's E-Commerce application. The test plan will be conducted to ensure that the application performs to an acceptable standard, mostly free of bugs and can be deployed to the production environment.

The test plan focuses on the most important components, and are split into functional and non-functional categories. Most of the tests will be conducted as a black box test (without access to the code) and manually done by the testing team. We have specified test scripts, which include the input and expected output, that can be used as reference by the testing team.

Automation Tools
- Bug tracking tool:- We're going to make issue for bugs w bug tag
- Code Coverage:- Jest v.29.5 for Frontend, Istanbul v.1 + Nyc v.15.1.0 for Backend
- System Concurrency:- Python v.3.9 with native Thread and Requests v.2.28.2
- UI Traffic Automation:- Test Cafe v.2.4.0
- Endpoint Coverage:- Postman v.8

Risk Based Approach Testing

Since testing is one of the last things done in the development cycle, it is possible that it will be one aspect that will be cut short. Therefore we will need a prioritisation system to determine which components will need to be tested first and what sorts of bugs are acceptable and which need to be fixed right away.

For components, we will rank them by exposure and impact. Components that have high exposure and impact will have the highest priority for testing, while those with low impact or low exposure will have lower priority.

For bugs, we will have a similar system, ranking them based on exposure and impact as well. This way we will have a system for devising which bugs need to be fixed right away, and which ones are lower priority and can be acceptable in the final product if resources are cut short.

Test Cycles

Stages of Testing

- Unit Testing: We will be developing unit tests for our functions as well. Developers will be writing their own test. These tests are for discrete units such as testing a lambda function, or small functions written, as opposed to testing components that are combined together. We will be using Istanbul and Nyc to assess code coverages.

- System Acceptance Testing: involves testing the complete and integrated system as a whole. It is performed to ensure that the system meets its functional and non-functional requirements. This component will be done manually. Alternative automation options we can look into is Selenium, given the time and resources. Our SAT will be testing our functional and non-functional components. We will be writing our tests using Black box testing techniques. SAT will be handed to our Quality Assurance members.

- User Acceptance Testing: This testing will be to assess the product from the point of view of our stakeholders, teaching staff and the sponsor.

We have a happy path and negative path for each component that will be executed multiple times, per component. Please refer to the appendix to see the exact number of test cycles. The test execution will be assessed based on the passing the acceptance criteria outlined below.

Passing Criteria

For the testing stages to be considered a success, the following goals should be achieved:

- 95% of unit tests are passing with a code line coverage of at least 70% for the front end (determined using Jest) and 80% for the back end (determined using Istanbul test coverage).
- The Regression Test Pack must pass at least 3 times before deployment.
- A full system test must have been performed once using the specified test scripts, with a test passing rate of 95%.
- Any identified defects must have a percentage occurrence of less than 1 in 10000 cases.
- Documentation of code is reviewed by a senior project manager
- All non-functional tests are passing.

Suspension Criteria: If more than 5% of the tests are failing, the testing process is halted and the results of the tests are forwarded to the developers. Once the tests resume, the entire testing suite will be restarted (No assumptions that passing tests maintain their status).

Resumption Criteria: At least 80% of the failing tests must pass before restarting the testing suite.

Functional Test Plan

Components List

Risks mentioned in the Regression Testing section

Comp No.	Component	Component Description	Testing Technique	Differences in Approach	Test Scripts *Refer to appendix	Exposure and Impact Low/Medium/High	Test Status
1	Register as a user	Login/Sign up with Google Auth	Black box/Manual testing	SAT	**Happy Path:** **1.1** - A signed out user with a gmail registers and logs into our website using Google OAuth. (3 Test Cycles) **Negative Path:** **1.2** - The user is already signed in (2 Test Cycles) **1.3** - The user registers with an email already associated with an account (2 Test Cycles)	**Exposure:** High **Impact:** High	Complete
2	Complete buyer profile	Complete remaining fields for a buyer (e.g.. department, etc.)	Black box/Manual testing	SAT	**Happy Path:** **2.1** - The user adds their department, payment, and shipping info to their account. (3 Test Cycles) **Negative Path:** **2.2** - The user enters the incorrect format for one of the fields. (2 Test Cycles)	**Exposure:** High **Impact:** High	Complete
3	Complete seller profile	Seller Registration Modal	Black box/Manual testing	SAT	**Happy Path:** **3.1** - The user adds their banking info to their account. (3 Test Cycles) **Negative Path:** **3.2** - The user enters the incorrect format for one of the fields. (2 Test Cycles)	**Exposure:** Medium **Impact:** High	Complete
4	Add items to shopping cart	Add items to the user's own shopping cart	Black Box + Manual Testing	SAT	**Happy Path:** **4.1** - The user clicks on the add to shopping cart button in the product details page, and the item is now in their shopping cart. (3 Test Cycles)	**Exposure:** High **Impact:** High	Complete

					Negative Path: **4.2** - The user is not logged in with a valid buyer's account and fails to add the item to shopping cart (2 Test Cycles)		
5	Remove items from shopping cart	Remove items to the user's own shopping cart	Black Box + Manual Testing	SAT	**Happy Path:** **5.1** - The user clicks on the remove item button beside an item, and that item is removed from the shopping cart (3 Test Cycles) **Negative Path:** **5.2** - The user clicks on the remove item button but fails due to the item not being present or already been removed (2 Test Cycles)	**Exposure:** High **Impact:** High	Complete
6	Checkout Items	Ability to purchase a desired item from the user's own shopping cart.	Black Box + Manual	SAT	**Happy Path:** **6.1** - Users with valid buyer details are able to complete a purchase of a valid item (5 Test Cycles) **Negative Path:** **6.2** - Users with invalid buyer details are unable to complete a purchase (3 Test Cycles) **6.3** - Users with valid buyer detail are unable to complete an already purchased item within the timespan of adding it to cart. (3 Test Cycles)	**Exposure:** High **Impact:** High	Complete
7	View Order History	Ability to view the order history and shipping status of a previously made purchase.	Black Box + Manual	SAT	**Happy Path:** **7.1** - Users with a previously made purchase are able to view the status and history of their order. (2 Test Cycles) **Negative Path:** **7.2** - User with previous purchases is unable to view their relevant order status (2 Test Cycles)	**Exposure:** Medium **Impact:** Low	Complete
8	Post Product To Sell	Posting as a seller	Black box/Manual testing	SAT	**Happy Path:** **8.1** - User is able to select 'Sell Now', add	**Exposure:** Medium	Complete

					their credit card information (if they haven't already) fill their listing details and post it. (4 Test Cycles) **Negative Path:** **8.2** - User tries to sell information without having privileges to sell/ they don't have their credit card information (3 Test Cycles) **8.3** - User tries to add the listing with one or more field missing (3 Test Cycles)	**Impact:** High	
9	Delete User as Admin	Ability to remove a non admin user using an admin account	Black box/Manual testing	SAT	**Happy Path:** **9.1** - Admin is able to delete an existing user from the application (3 Test Cycles) **Negative Path:** **9.2** - The user the is being deleted does not exist (2 Test Cycles)	**Exposure:** Low **Impact:** Low	Complete
10	Promote User as Admin	Ability to promote a normal user to an admin using an admin account	Black box/Manual testing	SAT	**Happy Path:** **10.1** - Admin is able to promote a regular user to one with admin privileges (3 Test Cycles) **Negative Path:** **10.2** - The user was promoted by another admin, but the current admin has not refreshed the page (2 Test Cycles)	**Exposure:** Low **Impact:** Medium	Complete
11	Demote User as Admin	Ability to demote a normal user to an admin using an admin account	Black box/Manual testing	SAT	**Happy Path:** **11.1** - Admin is able to demote an admin user to one with regular privileges (3 Test Cycles) **Negative Path:** **11.2** - The user was demoted by another admin, but the current admin has not refreshed the page (2 Test Cycles)	**Exposure:** Low **Impact:** Medium	Complete

12	Delete Listing as Admin	Ability to remove a listing as an admin. Posting needs to be an active one (hasn't been deleted previously)	Black Box + Manual Testing	SAT	**Happy Path:** **12.1** - The admin selects a listing to delete and successfully deletes it (3 Test Cycles) **Negative Path:** **12.2** - The admin is trying to delete a listing that is no longer active (has been deleted before) (2 Test Cycles)	**Exposure:** Low **Impact:** Low	Complete
13	Delete Order as Admin (DESCOPED)	Ability to remove an order as an admin. The order must still be in progress to be deleted/ cancelled.	Black Box + Manual Testing	SAT	**Happy Path**: **13.1** - The admin selects an order to delete and successfully deletes it (3 Test Cycles) **Negative Path:** **13.2** - The admin is trying to delete a completed order. (2 Test Cycles)	**Exposure:** Low **Impact:** Low	N/A

Non-Functional Test Plan

Components List

Comp No.	Component	Component Description	Testing Technique	Differences in Approach	Test Scripts	Exposure and Impact Low/Medium/High	Test Status
14	API Security	The user needs to log in using Google OAuth before accessing endpoints	Black box + Manual Testing	SAT	**Happy Path:** **14.1** - The user is able to log in to their account through Google OAuth, where they can see the correct personal information such as address, banking info, etc. and be able to modify them. (3 Test Cycles) **Negative Path:** **14.2** - The user does not log in through Google OAuth and do not have access to the application beyond the login page (2 Test Cycles) **14.3** - A logged in user would fail to see other users' personal information that is not displayed publicly such as banking info, and they would not be able to modify another user's information (3 Test Cycles)	**Exposure:** High **Impact:** High	Complete
15	Concurrent Accesses	Web application can handle 10 concurrent transactions without issue	Black box + Automated Test Script	SAT	**Happy Path**: **15.1** - A script automates 10 concurrent accesses and transactions with various endpoint tests, all should pass and return a 2xx response with the expected results. (3 Test Cycles) **Negative Path**: **15.2** - A script automates more than 10 concurrent accesses and transactions with various endpoint tests with some threads failing to receive a response upon either timeout or error codes such as 5xx. (3 Test Cycles)	**Exposure:** Medium **Impact:** High	Complete
16	Performance Evaluation	Response time for searches/filters should be under 4 seconds	Black box + Manual Testing	SAT	**Happy Path**: **16.1, 16.3, 16.4** - A user searches/filters for an item or an admin searches for an order/user and receives a result within 4 seconds. (3 Test Cycles) **Negative Path**:	**Exposure:** High **Impact:** Low	Complete

					16.2 - A user searches for a non-existing item and receives “No results found” within 4 seconds (2 Test Cycles) **16.5, 16.6** - An admin searches/filters for a non-existing user or order and receives “No results found” within 4 seconds (2 Test Cycles)		

Automated Testing

Python Scripts

Our test suite for testing concurrent user accesses on our web application will be written in Python and use the Thread and Requests libraries for simulating user traffic on Amazonian Prime.

We decided to choose this approach given Python's simple yet powerful capabilities to automate test scripts in an asynchronous environment, its libraries which can simulate the user's engagement with endpoints as well as the accessibility to integrate this test into our pipeline and larger test suite.

The test plan is to build a simple function in Python that will test several API calls to both our backend and frontend services to simulate network traffic. This function will contain several different calls to multiple endpoints to ensure liveliness and correctness in expected responses.

For the purpose of the libraries we have decided to use, the Thread library will provide the capability to run this function over several threads quite easily by assigning the function to a series of 10 or more threads to simulate 10 concurrent users at any given time. The Requests library will be used to test various endpoints within the system to additionally simulate user traffic by engaging with the web application.

Security Testing

Security Testing

Our application employs Google OAuth to connect users to their account, so their password and part of their account security are handled by the Google API. However, to ensure the safety of users' personal data, including their banking information, we must conduct tests on our end. These tests include confirming that users are logged into the correct account associated with their Google account and that they only have access to their own personal information. We will verify that users can modify their personal information, but not that of other users. To prevent the possibility of leaking sensitive user data during transit, we will use encryption on appropriate data. Additionally, we will perform status checks on the user's account at key components to ensure they have the necessary privileges to carry out the action.

Test Data Approach

As part of our test data approach, we will utilise SQL queries to populate the database with representative data for our tests. Given the limited scope of most of our tests, we anticipate that our test database will be relatively small compared to the actual user base. To ensure comprehensive coverage, we plan to create at least one account for each of the following roles: buyer, seller, and admin. Furthermore, we will test scenarios involving accounts with multiple roles, such as a user who is both a buyer and a seller, to verify that the system handles the associated permissions appropriately. This approach enables us to conduct data-driven testing and simulate a variety of scenarios. All the Google accounts used for testing will be new accounts created specifically for this project.

Regression Testing

Explanation

A collection of test cases performed as each new release/software update is completed:

During each deployment, we need to ensure the core functionality of the Amazonian Prime application with a collection of key test cases that should run post deployment. As such, this will ensure the continuous integration and functionality by catching any regressions/ bugs that impede core functionality. Such key functionalities will include: component 8- Post Product To Sell; component 4- Add items to shopping cart; component 6- Checkout items.

Testing component 8 is part of the core functionality since it makes up the key feature of the seller module, and is the only entry point in adding listings to the application for other buyers to view/ purchase. The risk of a failed test would mean that the application would not be able to show any new listings. Similarly, component 4 and component 6 is essential for the final purchase process, imperative to the functionality of the application. As such, the risk of a failing test would mean that orders will not occur, effectively hindering the buyer module. Such components may not have both high impact or high exposure, but are deemed important and valuable to include in the regression pack as these components are more likely to iterate and change. Other components (e.g. component 14- Delete order as admin) are not as imperative to the entirety of the application's heath, or they are not likely to change as frequently. Therefore, the risk is deemed lower for these components.

Furthermore, the **risk of a failure** in the regression test will effectively prove detrimental to the user workflow, as these components have been deemed as the major functionality that must not have any regression during deployments.

Appendix

Glossary of Terms / Abbreviations

Terms / Abbreviations	Description
Jest	Frontend testing framework for writing unit tests as well as measuring the code coverage of those tests.
Istanbul with NYC	Backend testing framework for writing unit tests as well as measuring the code coverage of those tests.
Test Cafe	End-to-end testing software that automates testing through the user interface
Postman	API development tool that also provides automated testing of APIs
Google OAuth	Provides user authentication with a service without the need of maintaining user private information such as usernames, passwords and other sensitive details.

Test Script Definitions

Script Number	Description	Assumptions	Preconditions	Postconditions	Description
Functional Component Test Scripts					
1.1	**Register While Signed Out**	1) The user has a gmail account	1) The user is signed in to Amazonian Prime	1) The user is logged in and redirected to the landing page	1) Go to Amazonian Prime while logged out 2) Check that you are redirected to the sign in page 3) Check that the “Continue with Google” button is rendered underneath the “Sign in or Register” prompt on the right 4) Click on the “Continue with Google” button 5) Check that you are redirected to Google OAuth flows. 6) Click on the email you wish to register an account with 7) Check that you are redirected to the Complete Buyer Profile page. 8) Click on the “Skip this step” button 9) Check that you are redirected to the landing page 10) Check that your name is displayed on the banner in the top right corner
1.2	**Register While Signed In**	1) The user has a gmail account	1) The user is signed in to Amazonian Prime	1) The user remains on the landing page (not log-in page)	1) Go to Amazonian Prime while logged in 2) Check that you are not redirected to the sign in page
1.3	**Register With An Email That Is Already Associated With An Account**	1) The user has a gmail account	1) There is a registered user with a gmail that is identical to the current user’s	1) The user logs into the system	1) Go to Amazonian Prime while logged out 2) Check that you are redirected to the sign in page 3) Check that the “Continue with Google” button is rendered underneath the “Sign in or Register” prompt on the right

					4) Click on the “Continue with Google” button 5) Check that you are redirected to Google OAuth flows 6) Click on an email that you have already registered an account with before 7) If you have completed the buyer profile with this account before, check that you are redirected to the landing page 8) Check that your name is displayed on the banner in the top right corner. 9) If you have not completed the buyer profile with this account before, check that you are redirected to the Complete Buyer Profile page.
2.1	**Complete Buyer Profile With A Valid Information**	1) The user can fill out all required information 2) The user has a gmail account to log in with 3) The user has not yet completed the buyer profile	1) The user has navigated to the buyer’s module	1) The user is registered as a buyer	1) Register or log into an account that does not have a completed buyer profile as documented above 2) Check that you are redirected to the Complete Buyer Profile page 3) Check that the input fields for first name, last name, card number, CVC, and MM/YY are rendered underneath “Payment Details” 4) Check that the input fields for street address, city, province, postal code, and country are rendered under “Billing Address” 5) Uncheck the “Same as billing address” checkbox and check that the input fields for street address, city, province, postal code, and country are rendered under “Shipping Address” 6) Input a string for first name, last name, street address, city, province, postal code, and country 7) Input 3 digit number for CVC, a 4 digit number for MM/YY, and a 16 digit number for card number, and click on the “Start Shopping” button 8) Check that you are redirected to the landing page. 9) Go to checkout an item as documented below 10) Check that the payment details, billing address, and shipping address are auto populated
2.2	**Complete Buyer Profile With Invalid Information**	1) The user can fill out all required information	1) The user has a gmail account to log in with 2) The user has not yet	1) The user is not able to register as a buyer	1) Complete steps 1-5 in the happy path above 2) Input an invalid input for any of the fields 3) Click on the “Start Shopping” button

			completed the buyer profile		4) Check that the page blocks you from moving to the landing page and highlights the invalid fields 5) Leave any of the fields empty. Click on the “Start Shopping” button 6) Check that the page blocks you from moving to the landing page and highlights the empty fields
3.1	**Complete Seller Profile With Valid Information**	1) The user can fill out all required information	1) The user has a gmail account to log in with 2) The user has not yet completed the seller profile	1) The user is registered as a seller	1) Navigate to the landing page with an account that does not have a completed seller profile as documented above 2) Check that the “Sell” button is rendered in the navigation bar 3) Check that the “Sell now” button is rendered in the banner 4) Click on either the “Sell” or “Sell now” button 5) Check that the Seller Registration modal pops up. 6) Check that the input fields for institution, account, and transit number are rendered within the modal. 7) Input a 3 digit number for institution number, 5 digit number for transit number, and 7-12 digit number for account number 8) Click on the “Save” button 9) Check that the Seller Registration modal closes. 10) Navigate to the user profile page by clicking on the profile icon next to the shopping cart in the top right corner 11) Check that the banking information is displayed correctly.
3.2	**Complete Seller Profile With Invalid Information**	1) The user can fill out all required information	1) The user has a gmail account to log in with 2) The user has not yet completed the seller profile	1) The user is unable to register as a seller	1) Complete steps 1-6 in the happy path above. 2) Input an invalid input for any of the fields 3) Click on the “Save” button 4) Check that the Seller Registration modal stays open and highlights the invalid fields. 5) Leave any of the fields empty 6) Click on the “Save” button 7) Check that the Seller Registration modal stays open and highlights the empty fields
4.1	**Add Items to Shopping Cart**	1) The user has a gmail account	2) The user has a valid buyer account 3) The user starts with 0 or more items in their shopping cart	1) The user has the correct quantity of the new item added to their shopping cart, and the items	1) Test will be conducted while logged in as a buyer 2) Click on a listing to go to the product details page and check that the “Add to Cart” button is properly rendered with the correct placement and style

				already in the shopping cart in precondition remain unchanged	3) Select the desired quantity and click on the “Add to Cart” button 4) Click on the cart icon on the top right of the page to go to the shopping cart page 5) Check that the correct item and quantity is in the shopping cart
4.2	**Add Items to Shopping Cart Invalid Account Type**	1) The user has a gmail account	1) The user has a valid buyer account 2) The user starts with 0 or more items in their shopping cart	1) The user has the same items in the shopping cart as in precondition and no change has been made	1) Test will be conducted while not logged in as a valid buyer (seller, admin, or invalid buyer) 2) Complete step 2-3 from the happy path above 3) The user fails to add an item to the shopping cart, and they receive an error message telling them they are not logged in as a valid buyer.
5.1	**Remove an Item from Shopping Cart**	1) The user has a gmail account	1) The user has a valid buyer account 2) The user starts with 1 or more items in their shopping cart	1) The user has the correct item removed from their shopping cart, and the items already in the shopping cart in precondition remain unchanged	1) Test will be conducted while logged in as a valid buyer already in the shopping cart page 2) Check that the “Remove Item” button is properly rendered with the correct placement and style 3) Click on the “Remove Item” button 4) Check that the correct item is removed from the list of items in the shopping cart
5.2	**Remove an Item from Shopping Cart Invalid Item**	1) The user has a gmail account	1) The user has a valid buyer account 2) The user starts with 1 or more items in their shopping cart	1) The user fails to remove the selected item from their shopping cart, and the items already in the shopping cart in precondition remain unchanged	1) Complete steps 1-3 in the happy path above 2) The user fails to remove an item that is no longer present in the system 3) The user gets an error message stating the item is not present
6.1	**Checkout Items with Valid Payment Details**	1) The shopping cart component is functional 2) The order history component is functional	1) The user is logged in as valid buyer 2) The user has valid payment details attached to their account. 3) A test listing is available to be purchased. 4) The user’s email is already registered to receive notification	1) The user receives an email notification and can view the newly purchased order on the Order History page.	1) Test will be conducted while logged in as a buyer with valid payment details 2) The user selects a product listing from the landing page by clicking on the item image 3) The user selects “Buy now” from the item view page and is redirected to the Checkout Page 4) Verify that the desired item is viewable in the checkout page 5) Click on the “Checkout” button to make the desired purchase 6) Verify that the order has been placed via web popup and email notification 7) Verify that the order is visible from both the internal database as well as the Order History Page

6.2	**Checkout Items with Invalid Payment Details**	1) The shopping cart component is functional 2) The order history component is functional	1) The user is a logged in as a buyer 2) The user has valid payment details	1) The order is not placed in the user database, or in the order history	1) Test will be conducted while logged in as a buyer with invalid payment details 2) The user selects a product listing from the landing page by clicking on the item image 3) The user selects “Buy now” from the item view page and is redirected to the Checkout Page 4) Verify that the desired item is viewable in the checkout page 5) Click on the “Checkout” button to make the desired purchase 6) Verify that the order has been rejected with an error message stating the user has invalid payment details and no further progress has been made visible from the page 7) Verify that the order is not visible from both the internal database as well as the Order History Page
6.3	**Checkout Items with Valid Payment Details but Insufficient Funds**	1) The shopping cart component is functional 2) The order history component is functional	1) The user is a logged in as a buyer 2) The user has valid payment details 3) The user does not have enough funds in the card	1) The order does not complete	1) Test will be conducted while logged in as a buyer with valid payment details but containing insufficient funds 2) Verify that the desired item is viewable in the checkout page 3) Click on the button to make the desired purchase 4) Verify that the order has been rejected with an error message stating the user has invalid payment details and no progress has been made visible from the page 5) Verify that the order is not visible from both the internal database as well as the Order History Page
7.1	**Order History with Previously Made Purchases**	1) The database containing user linked details to orders is properly mapped.	1) The user is logged in as a buyer with a valid account constructed. 2) The SQL script for constructing an order to a user’s order history is created	1) The order is viewable from the Order History page and contains the correct details in regards to order status	1) Tests will be conducted while logged in as a buyer. 2) The user runs the SQL script to add an item to the user’s order history 3) The user navigates to the Order tab by clicking “Orders” 4) Verify that the injected order is viewable on the Order History page by scrolling through the previously made orders 5) Verify that the status of each order matches the expected status set by the SQL script.
7.2	**Order History with Previously Made Purchases but are Not Viewable**	1) The database containing user linked details to orders is	1) The user is logged in as a buyer with a valid account constructed. 2) The SQL script for constructing	1) The order is not viewable from the Order History page	1) Tests will be conducted while logged in as a buyer. 2) The user runs the SQL script to add an item to the user’s order history 3) The user navigates to the Order tab by clicking “Orders”

		properly mapped.	an order to a user's order history is created		4) Verify that there are no orders viewable on the Order History page. 5) Verify through manipulation of the web page with techniques such as querying and sorting reveal no orders.
8.1	**Post Product To Sell As A Valid Seller**	1) The bank information is fake but can be used in this system for authenticatio n purposes.	1) The user is logged in as a seller and has a valid account created for this purpose	1) The listing is posted and is viewable by the user under their listings with the correct test specifications	1) Test will be conducted while logged in as a valid seller with valid bank information 2) Click on "Sell Now" to go to the "Add Listing" page, make sure that all the fields are rendered properly. 3) Upload an image to the listing successfully in JPEG format 4) Check that the listing is posted by visiting the Listings page
8.2	**Post Product To Sell As An Invalid Seller**	1) The user has a valid account	1) The user is logged in as a seller	1) The listing is posted and is viewable by the user under their listings with the correct test specifications 2) The seller has updated banking information added to their account and can be verified in the database.	1) Test will be conducted while logged in as a invalid seller with no banking information 2) Click on **"Sell Now"** to go to the **"Add Payment Information"** page, make sure that all the fields are rendered properly. 3) Verify that the user cannot continue until valid payment information is added to the seller's account. 4) The user continues to the "Add Listing" page and verifies that all the fields are rendered properly. 5) Upload an image to the listing successfully in JPEG format 6) Check that the listing is posted
9.1	**Delete User As An Admin That Has Not Been Deleted Yet**	1) The current user (admin) has admin privileges 2) The user being deleted exists in the application	1) The user logs in as an admin	1) The user gets deleted	1) As an admin, log into an account that has admin privileges 2) Verify that you get redirected to the landing page 3) From the landing page, check that the user icon on the tool bar is rendered 4) Click the user icon 5) Upon clicking the user icon, check that the user specific actions are rendered as a menu, and includes an option to "Switch To Admin Role" 6) Click the button to switch to admin role 7) Verify that you are redirected to the admin landing page 8) Verify that the page has a "Users" button in the toolbar 9) Upon click the "Users" button, verify that you get redirected to the page with users 10) Verify that there is a "Deactivate User" button next to users

					11) Find the user you wish to deactivate 12) Click the “Deactivate User” button for that user 13) Verify that there is a success toast in the application mentioning “The user has been deleted”
9.2	**Delete User As An Admin That Does Not Exist**	1) The current user (admin) has admin privileges 2) The user being deleted does not exist in the application	1) The user logs in as an admin	1) No user gets deleted	1) Follow steps 1- 12 in happy path above 2) Verify that there is a warning toast conveying the message: “Failed to delete a user: this user does not exist in the system”
10.1	**Promote User as Admin When The User Has Not Yet Been Promoted By Another Admin**	1) The user has admin privileges 2) The user has not yet been promoted	1) The user logs in as an admin	1) The user gets promoted by the admin	1) As an admin, log into an account that has admin privileges 2) Verify that you get redirected to the landing page 3) From the landing page, check that the user icon on the tool bar is rendered 4) Click the user icon 5) Upon clicking the user icon, check that the user specific actions are rendered as a menu, and includes an option to “Switch To Admin Role” 6) Click the button to switch to admin role 7) Verify that you are redirected to the admin landing page 8) Verify that the page has a “Users” button in the toolbar 9) Upon click the “Users” button, verify that you get redirected to the page with users 10) Verify that you see a “Promote” button next to the users 11) Find the user you wish to promote, and click “Promote” 12) Verify that there is a success toast that says that the user has been promoted
10.2	**Promote User As Admin When the User Has Already Been Promoted by Another Admin**	1) The user has admin privileges 2) The user has already been promoted	1) The user logs in as an admin	1) The user privilege does not change	1) Follow steps 1-11 in happy path above 2) Verify that there is a notification toast that informs you that the user has *already* been promoted, and nothing has changed
11.1	**Demote User As Admin When the User Has**	1) The user has admin privileges	1) The user logs in as an admin	1) The user gets demoted by the admin	1) As an admin, log into an account that has admin privileges 2) Verify that you get redirected to the landing page

	Not Yet Been Demoted by Another Admin	2) The user has not been demoted			3) From the landing page, check that the user icon on the tool bar is rendered 4) Click the user icon 5) Upon clicking the user icon, check that the user specific actions are rendered as a menu, and includes an option to “Switch To Admin Role” 6) Click the button to switch to admin role 7) Verify that you are redirected to the admin landing page 8) Verify that the page has a “Users” button in the toolbar 9) Upon click the “Users” button, verify that you get redirected to the page with users 10) Verify that you see a “Demote” button next to the users 11) Find the user you wish to promote, and click “Demote” 12) Verify that there is a success toast that says that the user has been demoted
11.2	**Demote User As Admin When the User Has Already Been Demoted by Another Admin**	1) The user has admin privileges 2) The user has already been demoted	1) The user logs in as an admin	1) The user privilege does not change	1) Follow steps 1-11 in happy path above 2) Verify that there is a notification toast that informs you that the user has *already* been demoted, and nothing has changed
12.1	**Delete Listing as Admin when the listing is still active**	1) The current user (admin) has admin privileges 2) The listing being deleted exists in the application	1) The user logs in as an admin	3) The listing has been deleted from the database	1) Log into an account that has admin privileges 2) Access the admin module by clicking on the user icon, and selecting the “Switch to Admin Role”. 3) Inside the admin module, select the option to view all listings 4) Select one of the listings from the list and click on it to view the details 5) Once redirected to the Product details page, select the option to remove the listing 6) Confirm that you would like to delete the listing 7) Verify that the deletion is a success by getting a prompt that the listing has been deleted
12.2	**Delete Listing as Admin when the listing is no**	1) The current user (admin) has admin privileges 2) The listing being deleted	1) The user logs in as an admin	1) No listing gets deleted	1) Log into an account that has admin privileges 2) Access the admin module by clicking on the user icon, and selecting the “Switch to Admin Role”.

	longer active	exists in the application			3) Inside the admin module, select the option to view all listings 4) Select one of the listings from the list and click on it to view the details 5) Once redirected to the Product details page, select the option to remove the listing. 6) Verify that there is a notification that informs the order is no longer active and cannot be deleted.
13.1	**Delete Order as Admin when the order hasn't been completed yet**	1) The order being deleted is not complete 2) The order being deleted exists in the database 3) The current user (admin) has admin privileges	1) The user logs in as an admin	1) The order gets deleted from the database	1) Log into an account that has admin privileges 2) Access the admin module by clicking on the user icon, and selecting the "Switch to Admin Role". 3) Inside the admin module, select the option to view all orders 4) Select one of the orders from the list and click on the option to remove the order 5) Confirm that you would like to delete the order 6) Verify that the deletion is a success by getting a prompt that the order has been deleted
13.2	**Delete Order as Admin when the order has been completed**	1) The order being deleted is not complete 2) The current user (admin) has admin privileges 3) The order being deleted exists in the database	1) The user logs in as an admin	1) No order gets deleted	1) Log into an account that has admin privileges 2) Access the admin module by clicking on the user icon, and selecting the "Switch to Admin Role". 3) Inside the admin module, select the option to view all orders 4) Select one of the orders that has been completed from the list and click on the option to remove the order 5) Verify that there is a notification that informs that the order has been completed and cannot be deleted.
Non Functional Test Scripts					
14.1	**API Security**	1) Google OAuth is implemented and connections can be established	1) The user must be logged out from Amazonian Prime 2) The user has a registered account using Google	1) The user can see the correct personal information in their profile 2) The user has the ability to modify the information in their profile	1) The user visits Amazonian Prime while logged out 2) The user clicks on "Continue with Google" to log in via Google OAuth 3) Once they login, he user should see the landing page with their profile icon on the top right 4) Click on the profile icon to go to account details

					5) Check that the information there is correct for this Google account 6) Check that the user can modify their personal information
14.2	**API Security Block Access If Not Logged In**	1) Google OAuth is implemented and connections can be established	1) The user must be logged out from Amazonian Prime	1) The user does not have access to anything beyond the login page	1) The user visits the website Amazonian Prime 2) The user cannot move onto the main website without going through Google OAuth 3) If the user clicks on “Continue with Google” but attempts to input an invalid login, they will be blocked from proceeding by Google.
14.3	**API Security Block Access to Other Users**	1) Google OAuth is implemented and connections can be established	1) The user must be logged out from Amazonian Prime	1) The user does not have access to other users’ private information 2) The user cannot modify other users’ information in any way	1) Complete steps 1-3 on the happy path above. 2) Check that a non-admin user would not be able to see private information of other users 3) Check that a non-admin user have no means of modifying any information of other users
15.1	**Handling Multiple Concurrent Users**	1) Internet connectivity for the tester is not being throttled or have excessive traffic on at the time of testing.	1) The test script is written and is either running on a tester’s personal workstation with Python installed and the corresponding libraries or the pipeline has created a valid Docker container to run this test.	1) The test passes with each of the threads exiting with valid return codes from the server.	1) Test script will be conducted using user credentials of various accounts for authentication. 2) The test script will iterate up to 10 times to construct 10 threads for running the test function. 3) Each thread will then start on the test function through another loop. 4) A thread will call several GET requests on a user’s order history and current listings view on the main page to both the frontend and backend. 5) A thread will simulate following the buyer workflow of placing a listing in their cart and removing it via POST and PUT requests to the backend. 6) A thread will simulate following the sellers workflow of submitting a new product for listing and removing that product shortly after 7) Each thread shall receive a 2xx response with the expected response message and header, while within a reasonable amount of time.
15.2	**Handling More than 10 Concurrent Users**	1) Internet connectivity for the tester is not being throttled or have excessive	1) The test script is written and is either running on a tester’s personal workstation with Python	1) The test script returns with one or multiple failed responses and logs the errors to console for debugging.	1) Test script will be conducted using user credentials of various accounts for authentication. 2) The test script will iterate beyond a limit of 10 times to

		traffic on at the time of testing.	installed and the corresponding libraries or the pipeline has created a valid Docker container to run this test.		construct many threads for running the test function. 3) Each thread will then start on the test function through another loop. 4) A thread will call several GET requests on a user's order history and current listings view on the main page to both the frontend and backend. 5) A thread will simulate following the buyer workflow of placing a listing in their cart and removing it via POST and PUT requests to the backend. 6) A thread will simulate following the sellers workflow of submitting a new product for listing and removing that product shortly after 7) Each thread shall check if a response returns with a 5xx or other invalid response, or simply timeouts before the user receives a valid response.
16.1	**Search for an existing item**	1) The item being searched for exists in the database	1) The account used for this test is registered and is able to view listings on the landing page.	1) The results load within 4 seconds and are accurate	1) Navigate to the landing page on a registered account as documented above. 2) Search for an existing item by name in the search bar in the navigation bar. 3) Check that the search results page loads with accurate results within 4 seconds
16.2	**Search for a non-existing item**	1) The item being searched for does not exist in the database	1) The account used for this test is registered and is able to view listings on the landing page.	1) The results load within 4 seconds and are accurate	1) Navigate to the landing page on a registered account as documented above. 2) Search for a non-existing item by name in the search bar in the navigation bar. 3) Check that the search results page loads with "No results found" within 4 seconds.
16.3	**Search for an existing user as an admin**	1) The user being searched for exists in the database	1) The user has been granted admin access	1) The results load within 4 seconds and are accurate	1) Navigate to the admin module on an admin account as documented above. 2) Navigate to the users section by clicking on the "Users" button in the navigation bar 3) Search for an existing user by name in the search bar in the navigation bar. 4) Check that the search results page loads with accurate results within 4 seconds.
16.4	**Search for an existing order as an admin**	1) The order being searched for exists in the database	1) The user has been granted admin access	1) The results load within 4 seconds and are accurate	1) Navigate to the admin module on an admin account as documented above. 2) Navigate to the orders section by clicking on the "Orders" button in the navigation bar

					3) Search for an existing order by name/orderID in the search bar in the navigation bar. 4) Check that the search results page loads with accurate results within 4 seconds.
16.5	**Search for a non-existing user as an admin**	1) The user being searched for does not exist in the database	1) The user has been granted admin access	1) The results load within 4 seconds and are accurate	1) Navigate to the admin module on an admin account as documented above. 2) Navigate to the users section by clicking on the “Users” button in the navigation bar 3) Search for a non-existing user by name in the search bar in the navigation bar. 4) Check that the search results page loads with “No results found” within 4 seconds.
16.6	**Search for a non-existing order as an admin**	1) The order being searched for does not exist in the database	1) The user has been granted admin access	1) The results load within 4 seconds and are accurate	1) Navigate to the admin module on an admin account as documented above. 2) Navigate to the orders section by clicking on the “Orders” button in the navigation bar 3) Search for a non-existing order by name/orderID in the search bar in the navigation bar. 4) Check that the search results page loads with “No results found” within 4 seconds.