BoilerX

Code Inspection, Design Inspection, Unit Testing Log

Team Members: Sean Chew, Zuyuan Fan, Qi Meng, Ling Zhang

Design Inspection

Product	BoilerX		
Date	Feb. 10, 2018		
Author	Development team		
Moderator	Sean Chew		
Inspector	Zuyuan Fan, Ling Zhang		
Recorder	Qi Meng		
Defect #	Description	Severity	How corrected
1	Sign up window does not contain field for sign up confirmation code	3	Implement "confirmation code" filed and create a send confirmation code button on the "purdue email address" field
2	Some of the submission field boxes is designed using some fixed-size elements constraints. This means that some fields in the UI have a maximum allowable size.	3	Warn the user if the context within the submission field exceeds the element constraint and limit user's maximum input on required fields.
3	The system doesn't allow users to submit images larger than a fixed limit in size of 3MB. This might pose an issue for users submitting high-definition pictures either for their profile pictures or item images, as these files then to be larger in size.	2	As our system isn't currently capable of supporting a large database, an error message will pop up when the user tries to upload an image that is too large in size.
4	There is no protection mechanism set in place for buyers to confirm a purchase.	3	Implement a 2-step verification process that will prompt the buyer to confirm the purchase of the

	This might cause problems such as an accidental purchase of a good due to a misclick.		good.
5	After a buyer buys an item, the item is not immediately taken down from the site. This might cause problems such as multiple buyers purchasing the same item. However, this isn't a definite drawback as it gives the seller secondary options incase the primary deal falls through.	3	Implement a queue data structure in the database, recording the order of users who purchased the good. This will allow the seller to go through the queue and sell the good to the first available customer who meets the seller's requirements.

Code Inspection

Code inspection revealed the following defects:

Product	BoilerX		
Date	Feb. 10, 2018		
Author	Development team		
Defect #	Description	Severity	How corrected
1	Methods returned literal true and false values instead of the correct syntax for each associated with them.	2	Edit boolean return syntax statements to respond accordingly
2	Code indentation not done properly. Syntax error caught by IDE.	3	All indentation errors were corrected.
3	Api method /user/create will save user password to database, security issue may arise.	3	Remove "pass: data.password" from Item attribute of dynamoDB put request.
4	Dynamically constructed payload is not initialized.	1	Change "let attr_name;" to "let attr_name = {};" Change "let attr_value;" to "let attr_value = {};"

Unit Testing Report

The system relies on the following modules:

- Query Module
- Item Display Interface
- User Profile Interface
- User Profile Manager
- Interface Components
- Item Management Module
- Item Management Interface
- Authentication Module

Query Module

Query module is the backend module responsible for getting items according to search keywords and filters from the database. The module consists of API methods to handle requests from client choices and input. Users will input search string and select conditions on the results including category, subject, price range, etc. The module will look up the database and provide client with list of desired items. Database operations are tested using "serverless invoke local --function f --path p" to perform mock events pre-written at path p.

Product	BoilerX		
Module	Query Module		
Author	Development team		
Defect #	Description	Severity	How corrected
1	Get_items* methods get_items_popularity and get_items_price will throw exception if body is empty: no filter/search condition specified	1	Create another method that get all items according to popularity for home page use, with no filter conditions.
2	get_items* methods will return error if search keyword is an empty string.	2	Verify the string input in search bar: the get method will not be triggered when length of input string is less than 1.
3	get_items* return nothing when specifying price range, even if there are valid items in the table.	1	The upper and lower bound of price were parsed as string. Use Number(data.price[0]) to set lower bound of price to number,

			same for upper bound.
4	get_items* return error when no search keyword specified.	1	When concatenating filter condition string, an " AND " is always at the beginning, except for condition for keyword. Eg. " AND #cat = :cat". So there will be a redundant "AND" if keyword condition does not exist in request body. Add a boolean variable to add the " AND " only when the next condition string to be added is not the first one.

Interface Components

Interface components include buttons, input manager, and other components in web interface. These interface components are tested using Mocha, Chai, and jsdom React testing framework with automated test runs.

Product	BoilerX		
Module	Button UI Widget		
Author	Development team		
Defect #	Description	Severity	How corrected
1	Test_select_callback showed that proper callback function was not called when button is selected	3	Now added proper push function in onSubmit attribute of <form> that direct user to correct path or respond with correct alert.

Product	BoilerX		
Module	Input UI Widget		
Author	Development team		
Defect #	Description	Severity	How corrected
1	Test_input_length showed that no constraints on the length of	2	Added proper constraints on length that stop user from

	the input.		entering more words
2	Test_input showed that texts are stopped from entering.	3	Added proper form reducer that takes in user input and update its value.

Item Display Interface

The item interface is the frontend user interface that handles displaying all fetched items on the search result page as well as handles displaying detailed item page. User interact with the interface by input in search bar and click select item filters. It is tested manually.

Product	BoilerX		
Module	Item Display Interface		
Author	Development team		
Defect #	Description	Severity	How corrected
1	Item tags could not be displayed on the items on the search result page.	3	Not yet corrected. Correction will need to connect and bind the results from reducers and actions.
2	Item description for each individual item on the search result page could not display "Read more" when the description exceed 17 words.	2	In style.css, include "text-overflow: ellipsis;" for the text description class.
3	The photo gallery for the item on the detailed item page can only display one image.	2	Not yet corrected. Correction will need to use the react-redux framework to pass the image urls to the container.

Authentication Module

The authentication manager is the integrated module that handles user signup, login, and logout authentication as well as a two-step verification process verifying that the user has a valid purdue email account. User will provide valid purdue email and a password for login and signup. Both the frontend and backend need to verify user input. Upon success, AWS Cognito will issue credential to use the webapp. Logging out will simply clear the credential. This module is tested mostly by hand with various inputs. Database operations are tested using "serverless invoke local --function f --path p" to perform mock events pre-written at path p.

Product	BoilerX		
Module	Authentication Module		
Author	Development team		
Defect #	Description	Severity	How corrected
1	Login method boolean constantly returns “true” even when the username/password does not match.	1	Fixed the defect due to a logical error in the frontend code.
2	Passwords require at least 6 characters with one uppercase letter, one lowercase and one special character. Currently, system only checks to see if there are at least 6 characters.	3	A boolean condition at front end checks dynamically if the input password satisfies the requirement. We also configured AWS Cognito user pool for the app to accept desired password.
3	Cognito user id is not saved in the database. Only email is saved as primary key of user table, which might be visible to other hostile users.	3	Although we ensure uniqueness of email through Cognito federated identity, it is more desirable to hide their information. We extract user id from event.requestContext.identity.cognitoIdentityId And save it to user table. We might remove the email attribute later.