BoilerX

Incremental Testing and Regression Testing Log

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

Classification of Components

Module Description

Query Module

Query module is the backend module responsible for getting items according to search keywords and filters from the database.

Input: A string representing the searched item that the user wants to retrieve.

Output: The array of item objects that matches the string input.

Interface Components

Interface components include buttons, input manager and other components in the web interface.

Input: Frontend interface labels and callback function.

Output: Callback function is called when button is pressed.

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.

Input: A string inputted into the search bar representing the desired item.

Output: The item objects that matches the string input.

Authentication Module

The authentication manager is the integrated module that handles user signup, login, and authentication as well as a two-step verification process verifying that the user has a valid purdue email account.

Input: User signup/login credentials such as Purdue email address, account password, and confirmation code.

Output: A confirmation of whether or not the signup/login was successful.

User Profile Interface

The user profile interface is the frontend user interface that handles displaying the user's profile information. Users interact with the interface by clicking on their profile picture located at the top right of the page.

Input: A mouse click on the user's profile picture located at the top right of the page.

Output: The user's profile information, including username, email, password, etc.

Item Management Interface

The item management interface is the frontend user interface that handles displaying all fetched items the user bought and items the users sold. Here, sellers will be able to edit the status of items that they are currently selling as well. Users interact with the item management interface through the user profile interface.

Input: A mouse click on the "items bought" and "items sold" UI button accessed through the user profile interface.

Output: List of items that the user bought and sold in the past.

Item Management Module

The item management module is the backend module that handles requests to update details of an item. Web application client will request to change fields of a certain item. The module will look up the database and save the update details to the the desired item.

Input: ItemID, new values of item fields to be updated.

Output: full item detail after updating.

User Profile Module

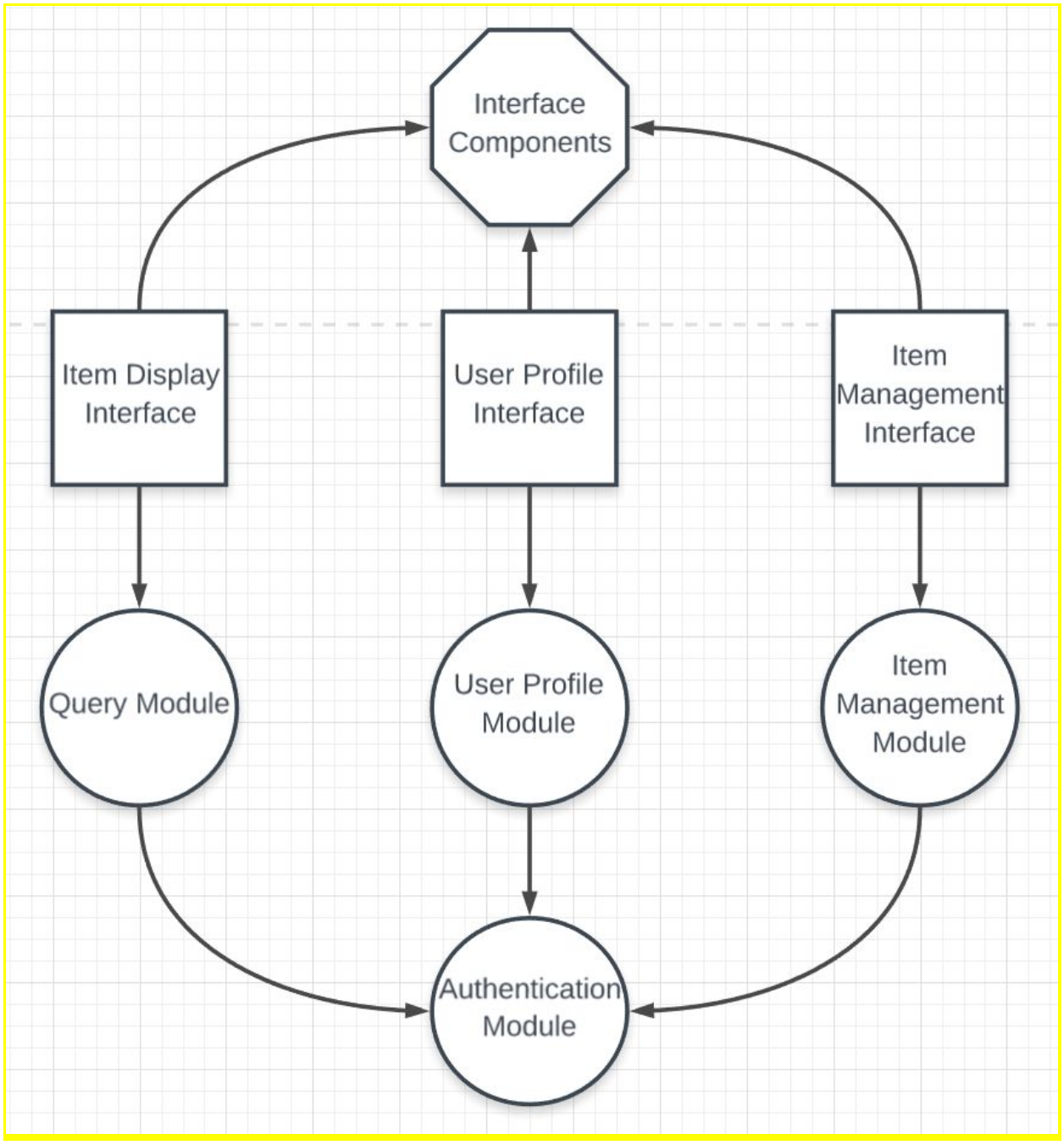
The user profile management module is the backend module that handles requests to get and update detailed user profile. Web application client will request to change fields of a certain user profile. The module will look up the database and save the update details to the the desired user.

Input: userId, new values of user fields to be updated.

Output: full user profile after updating.

Incremental Testing Technique

We incrementally tested our application using a bottom-up approach. The modules at the bottom were implemented and tested first and we would fix those bugs. Next, we would integrate the modules together with the interfaces at the top. Then, testing would have to be done again until the application fixes all the bugs.



From a bottom-up incremental testing approach, we first used drivers to test the backend modules at the bottom to ensure that we got rid of the majority of the bugs located within the AWS modules. Afterwards, we would integrate the lower level modules with the upper level interfaces one step at a time and tested progressively. At each stage, the integrated modules and interfaces would have to pass a series of unit test cases before we would move on with more integration which is tested.

Incremental and Regression Testing

Automation

Query Module

Module testing changed from manual command line testing with valid or invalid mock events to automated Linux shell script tests. This component is also tested with sets of regression tests. Each time a bug is fixed, this component is tested on inputs that previously caused the bug, the inputs that may relate to this bug, and random inputs. Thus, we make sure that the bug is correctly fixed and no new bugs are introduced.

Interface Components

Interface components are tested using the Jest and Enzyme React testing framework with automated test runs. Tests are done to examine all boundary cases like the visibility of interface component, which includes a variety of inputs and different combination of components.

Item Display Interface

Item display interface testing uses WebdriverIO automation testing framework. Tests are designed to cover edge cases and a variety of inputs under different user status. Item display interface acts differently when users are logged in and logged out.

Authentication Module

Module testing changed from manual command line testing with valid or invalid mock events to automated Linux shell script tests. This component is also tested with sets of regression tests. Each time a bug is fixed, this component is tested on inputs that previously caused the bug, the inputs that may relate to this bug, and random inputs. Thus, we make sure that the bug is correctly fixed and no new bugs are introduced.

User Profile Interface

User profile interface testing uses WebdriverIO automation testing framework. Test covers multiple cases where user navigate through the user profile interface, and change username and profile picture. User profile interface acts differently when users are logged in and logged out.

Item Management Interface

Item management interface testing uses WebdriverIO automation testing framework. Test covers multiple cases where user input item name, item price, item tag, and short

description and upload item picture. Item management interface would only be accessed when the user is logged in.

Item Management Module

Database operations of item management module are tested using serverless to invoke local API methods with mock requests and automated using Linux shell scripts. Formatted mock requests are randomly generated with Node.js scripts. Generation of these random mock requests are automated in the test scripts. This component is also tested with sets of regression tests. Each time a bug is fixed, this component is tested on inputs that previously caused the bug, the inputs that may relate to this bug, and random inputs. Thus, we make sure that the bug is correctly fixed and no new bugs are introduced.

User Profile Module

Database operations of user profile module are tested using serverless to invoke local API methods with mock requests and automated using Linux shell scripts. Formatted mock requests are randomly generated with Node.js scripts. Generation of these random mock requests are automated in the test scripts. This component is also tested with sets of regression tests. Each time a bug is fixed, this component is tested on inputs that previously caused the bug, the inputs that may relate to this bug, and random inputs. Thus, we make sure that the bug is correctly fixed and no new bugs are introduced.

Defect Log

Module	Query Module

Incremental test

Defect No.	Description	Severity	Solution
1	When the frontend was using the API GET request to fetch the items by keywords using "get-item" Lambda function, the backend failed to execute GET request.	2	Fetch all item information to the client, managing a local filter and search.

Regression test

Defect No.	Description	Severity	Solution
1	When the buyer presses buy, the frontend isn't able to retrieve the seller's email through the backend API.	3	By passing seller's information when getting items from backend API, we could easily retrieve seller's email & send email by another API call.

Module	Item Display Interface

Incremental test

Defect No.	Description	Severity	Solution
1	Views of each item could not be increased each time a user click on the item	2	Implement backend API call, and call the API call when a user clicked on the item.
2	Name and description for the item display interface have unlimited text length.	2	By setting maxLength for FormControl input field on edit and sell item interfaces.

Module	User Profile Interface

Incremental test

Defect No.	Description	Severity	Solution
1	User profile picture can not be uploaded to database with access denied error	2	Grant access to bucket in config.js
2	User profile pictures can not be saved into a specific folder in the bucket	1	Grant access to specific folders in IAM and append specific folder name when creating AWS.S3 object in frontend.

Regression test

Defect No.	Description	Severity	Solution

1	User profile information are not automatically updated after the user press save button	1	Refresh the page by adding window.location.reload() upon clicking the saving button.

Module	Item Management Interface

Incremental test

Defect No.	Description	Severity	Solution
1	Failed to upload item picture to database with access denied error	2	Grant access to bucket in config.js
2	Failed to save item pictures into a item_img folder in the bucket	1	Grant access to item_img folders in IAM and append specific folder name when creating AWS.S3 object in frontend.

Regression test

Defect No.	Description	Severity	Solution
1	Item information is not automatically updated after the user presses the save button	1	Refresh the page by adding window.location.reload() upon clicking the saving button.

Module	Item Management Module

Incremental test

Defect No.	Description	Severity	Solution
1	Item popularity can be updated by user. However, it should be automatically incremented by the application when another user views the item.	1	Popularity should not be updated by the update item function. Create a separate function to increase the item popularity every time a user view the item
2	Server fail to create new item in	1	Some optional fields are required

	the database.		to have a successful create item api call. In the backend api call, check if those optional fields exist in the request; if not, put a placeholder value for that field.
3	There are some attributes that are never used, such as item creation time.	3	Remove those items in post item and update item functions, to achieve higher efficiency.

Regression test

Defect No.	Description	Severity	Solution
1	Server fail to create new item in the database.	1	The newly added checking of required fields contains a typo. Should check if 'event.body.sellerName' exist, instead of 'sellerName' which will always return false.
2	Server fail to update the popularity of the new item.	1	Update popularity function tried to find the field named popularity in request body and update the popularity accordingly. However, we do not use parsed in values, the function will increment the popularity attribute directly. The problem was resolved by removing the checking of popularity attribute in request body.
3	After removing redundant attributes of add/update item, the item cannot be updated. After the last attribute's removal, the UpdateExpression has an extra comma at the end of the string, causing the AWS document client to throw an EOF error.	1	Remove the last extra comma of UpdateExpression of update_item function.

Updated Product Backlog

Requirements

- Functional

Backlog ID	Functional Requirements	Hours	Status
1	As a user, I would like to be able to sign up using my Purdue email	2	Completed in Sprint 1
2	As a user, I would like to be able to verify my email address	4	Completed in Sprint 1
3	As a user, I would like to be able to set a password and change password any time I log in.	3	Removed from Sprint 2
4	As a user, I would like to be able to change username	2	Completed in Sprint 2
5	As a user, I would like to be able to set a profile picture	2	Completed in Sprint 2
6	As a user, I would like to be able to login to my account with my Purdue email and password	5	Completed in Sprint 1
7	As a user, I would like to be able to logout of my account	3	Completed in Sprint 1
8	As a user, I would like to be able to search for items by keywords	7	Completed in Sprint 2

9	As a user, I would like to be able to filter items by category	3	Removed from Sprint 2
10	As a user, I would like to be able to sort items by price or popularity	4	Removed from Sprint 2
11	As a user, I would like to be able to see the most popular items on the homepage	5	Completed in Sprint 1
12	As a user, I would like to be able to view my account information, which includes my personal profile and item listings	4	Completed in Sprint 2
13	As a user (seller), I would like to be able to post an item for sale	3	Completed in Sprint 2
14	As a user (seller), I would like to be able to add title and/or description for the item that I want to sell	2	Completed in Sprint 2
15	As a user (seller), I would like to be able to upload pictures of the item that I want to sell	3	Completed in Sprint 2
16	As a user (seller), I would like to be able to create tags for the item that I want to sell	2	Removed from Sprint 2
17	As a user (seller), I would like to be able to set a price for the item that I want to sell	2	Completed in Sprint 2
18	As a user (seller), I would like to be able to edit my item information after I post them for sale	2	Completed in Sprint 2
19	As a user (buyer), I would like to be able to buy an item by clicking the “buy button” which will send an system email notification to the seller	5	Completed in Sprint 2

20	As a user (seller), I would like to be able to receive a system email stating the buyer's email and the item information once a sale is made	3	Completed in Sprint 2
21	As a user (seller), I would like to be able to take down the listed item after it is sold	5	Completed in Sprint 2
22	As a developer, I would like to inspect my code, unit inspect my code, and inspect the design of my code	100	Completed in Sprint 2
	Total Hours:	*171*	

- Non-functional

Backlog ID	Non-functional Requirements	Hours	Status
23	As a developer, I would like to learn how to use ReactJS	10	Completed in Sprint 1
24	As a developer, I would like to learn how to use serverless framework to configure AWS services	10	Completed in Sprint 1
25	As a developer, I would like to learn to build a custom API hosted on AWS API Gateway	20	Completed in Sprint 1
26	As a developer, I would like to learn how to configure DynamoDB and S3 cloud storage for our web app	10	Completed in Sprint 1
	Total Hours:	*50*	