Circuit Collective Phase 4 Report

Circuit Collective Final Project Submission

1. Final Codebase

Our final codebase is available publicly in our **Github repository**.

---

2. Final Test Report

Unit test

Unit test is test by (condition == condition), result will or be true or false

Test ID	Method/Class	Input(s)	Expected Output(s)	Testing Approach (Manual/Automated)	Assigned Team Member	Test Result

UT-01-CB	insert()	new Game() {{
		name ="Game1";
		desc = "Desc1";
		stock = 1;
		Tags = Set.of("TagA", "TagB");
		revenue = 160.0;
		price = 80.0;
		}},
			Game	Automated (JUnit)	Jerry Nathan	true

UT-02-TB	insert()	null	null	Automated (JUnit)	Jerry Nathan	true
UT-03-TB	list()	null	Game[0..*]	Automated (JUnit)	Jerry Nathan	true
UT-04-CB	search()	“Mario”	Game[0..*]	Automated (JUnit)	Jerry Nathan	true

UT-05-CB	search()	null	Game[0..*]	Automated (JUnit)	Jerry Nathan	true
UT-06-CB	search()	“”	Game[0..*]	Automated (JUnit)	Jerry Nathan	true

UT-07-OB	fetch(`${api}/game/search?l=25&q=${query}`)	“”	JSON
			List of gameData
			{
			Id: number,
			Desc: string,
			Stock: number,
			Revenue: number,
			Price: number,
			Tags: String[0..*]
			}
				Automated (JUnit)	Jerry
					Tobenna
						true

UT-08-OB	fetch(`${api}/game/search?l=25&q=${query}`)	“Mario”	JSON
			List of gameData
			{
			Id: number,
			Desc: string,
			Stock: number,
			Revenue: number,
			Price: number,
			Tags: String[0..*]
			}
				Automated (JUnit)	Jerry
					Tobenna
						true

UT-09-OB	fetch(`${api}/game`)	null	JSON
			List of gameData
			{
			Id: number,
			Desc: string,
			Stock: number,
			Revenue: number,
			Price: number,
			Tags: String[0..*]
			}
				Automated (JUnit)	Jerry
					Tobenna
						true

UT-10-OB	fetch(`${api}/admin/game/${id}`, {method: "DELETE"})	null	null	Automated (JUnit)	Jerry Tobenna	true
UT-11-OB	fetch(`${api}/admi	“Mario”	null	Automated	Jerry	true

	n/game/${id}`, { method: "DELETE"})			(JUnit)	Tobenna	

UT-12-OB	fetch(`${api}/admin/game/${id}`, { method: "PUT"})	“Mario”	JSON of gameData
			{
			Id: number,
			Desc: string,
			Stock: number,
			Revenue: number,
			Price: number,
			Tags: String[0..*]
			}
				Automated (JUnit)	Jerry Tobenna	true

UT-13-OB	fetch(`${api}/admin/game/${id}`, { method: "PUT"})	null	null	Automated (JUnit)	Jerry Tobenna	true
UT-14-OB	fetch(`${api}/admin/game/${id}`, { method: "POST"})	null	null	Automated (JUnit)	Jerry Tobenna	true

UT-15-OB	fetch(`${api}/admin/game/${id}`, { method: "POST"})	“Mario”	JSON of gameData
			{
			Id: number,
			Desc: string,
			Stock: number,
			Revenue: number,
			Price: number,
			Tags: String[0..*]
			}
				Automated (JUnit)	Jerry Tobenna	true

Integration test

Integration test involved with multiple component, result is described since it cannot have actual program output

Test ID	Components Involved	Preconditions	Steps	Expected Result	Assigned Team Member	Test Result
IT-01-OB	Login Interface input Backend login validation	Backend valid user logged in	Enter valid username and password, submit	Backend valid as login successful	Jerry Nathan William Bryce	Valid password: Successful login, accessed to category page. Backend components register as valid users. Invalid password: Login failed and stood in the login page.
IT-02-CB	Web API on localhost Web API connect on javascript	Can be accessed by javascript on frontend	Request data in javascript, using fetch()	JSON responses depend on method called by fetch()	Nathan Tobenna	Frontend: All responses by javascript fetch() worked from the Web API component. Backend: Successful got input from frontend and response with backend.
IT-03-OB	Apache Lucene Search Search engine wrapper	Algorithm run over list, give index based on string being searched	On backend, call search() of algorithm imported	Successful return list of index	Nathan	Giving input name with exist item: Successful return result of item. Giving input name with non-exist item: Successful return with empty list of items.
IT-04-TB	Inserting item interface input Database writing	Database proceed input from frontend, return a JSON based on given parameter	On html category web, type in name, desc, stock, revenue, price, stock, tags, click add	Database return JSON contain game information base on given information	Jerry Nathan Tobenna	Giving valid input of item attribute: Successful inserted new item into item list. Giving invalid input of item attribute: Nothing happened, the item failed to insert into the item list.
IT-05-TB	Deleting item interface input	Database remove item, update	On html category web, click delete	Database successfully	Jerry Nathan	Deleting exist item: Successful removed

	Database removing	frontend item list	button right next to item display	delete item	Tobenna	from item list Deleting non-exist item: Nothing happened, since the item does not exist.
IT-06-TB	Search bar interface input Database search engine calculation	Database proceed input from frontend, using search algorithm and return result	On html category web, type in text in search bar, hit enter	Database successfully run over algorithm and return JSON contain list of result	Jerry Nathan Tobenna William Bryce	Input string into search bar: Successful return reasonable result of item from item list.
IT-07-TB	Item new edit interface input Database property change	Database change item's attribute	On html category web. click edit and change attribute of item hit enter	Database successfully changed item's attribute	Jerry Nathan Tobenna	Editing attribute and with saving change: Successful change, and a new attribute is displayed. Editing attribute and without saving change: Nothing changed, since it did not save the change.

System tests

System test involved with whole program, result is described since it cannot have actual program output

Test ID	Scenario	Preconditions	Steps	Expected Outcome	Assigned Team Member	Test Result
ST-01-CB	Logging	User logged in	Enter username and password in html web	Logged in and direct toward category page	Jerry Nathan Tobenna William Bryce	Successful login and direct to category page, and is accessible to backend databases.
ST-02-TB	Inserting	Item is inserted in category	On html category web, type in name, desc, stock, revenue, price, stock, tags, click add	Displays new item in list that is requesting to add	Jerry Nathan Tobenna William Bryce	Giving valid input: Successful added item info to backend, and display new item in list in frontend. Giving invalid input: Nothing added to the backend, and nothing displayed in the frontend.
ST-03-OB	Searching	Filter list based on what is being searched	On html category web, type in text in search bar, hit enter	List has been filtered based on text inside search bar	Jerry Nathan Tobenna William Bryce	Search with random input: Based on input, all results of the item are close to input, which is reasonable.
ST-04-TB	Deleting	Item is deleted and gone in category	On html category web, click delete button right next to item display	Item disappear from category list	Jerry Nathan Tobenna William Bryce	Item is removed from the backend and no longer displayed in the item list in the frontend.
ST-05-TB	Editing	Item attribute changed	On html category web. click edit and change attribute of item hit enter	Item attribute changed and shows new attribute inside category page	Jerry Nathan Tobenna William Bryce	Changing attribute with save: Attribute change in database and displayed in item list in frontend. Changing attribute without save: Nothing has been changed in the database and the frontend stands the same.

Bug discovered:

- Text Field's Text Not Resetting & Clearing:
    - Documented:
    - Occur after inserting an item.
    - After adding an item, the text within the text field remains instead of clearing its contents.
    - Solved by setting the text field to an empty string.
- Login Page Redirection:
    - Documented:
    - Occur after logging in.
    - After attempting to log in as an administrator, the login page does not direct to the game category page.
    - Solved by directing in frontend.
- Inserting Incorrect Data Types via JSON Parsing:
    - Documented:
    - Occur when attempting input invalid data when inserting and editing
    - Inserting the wrong data type causes the javascript file to break and failing to fetch the data from the database.
    - Fix by parsing data differently.
- Invalid Index ID:
    - Documented:
    - Occur when inserting a new item.
    - Each initialized item has a random and/or incorrect ID instead of an organized index ID.
    - Fixed by index increment instead of random.

Test coverage metrics:

Unit test	**Integration test**	**System test**
Inserting a game into the database	The Login System	Logging In
Listing the games in the database	Web API on localhost	Inserting Items into the Catalog
Searching for a game in the database	Apache Lucene Search	Searching for Items in the Catalog
Fetching a game from the database	Inserting Item	Deleting Items from the Catalog
	Delete Item	Editing Items in the Catalog
	Search Item	
	Edit Item	

---

3. Deployment or Packaged Executable

A release can be found in the project repository. Instructions are in the README file and are linked in the release notes. The JAR is quite large due to the numerous dependencies. It could likely be shrunk with proguard rules however binary size was not a priority for this project.

---

4. Contributions Report

William Wedemire *(Team Lead)* - As the Team Lead for the project, William had a highly documentation and leadership based role on the project. This included development of slideshows used during customer meetings, along with creation of documentation related to the phase reports. Additionally, William assisted in mitigating problems between team members, and ensuring that teamwork ran smoothly. Finally, William fulfilled his responsibilities as team lead with planning scrum meetings, and being the main point of contact for the customer if any important issues or changes came up.

Bryce Gill *(Technical Manager)* - As the Technical Manager, Bryce had a highly documentation and management based role on the project. This included management of pull requests for the github, testing of pull requests, and identifying issues within the project. Additionally, Bryce assisted with leading the project in the form of assisting in working with team members for bettering team dynamics, and conflict resolution. Finally, Bryce fulfilled his responsibilities as Technical Manager with overseeing version control, repo management, and the creation of many of the diagrams including many of the UML diagrams and burndown charts.

Tobenna Nnaobi *(Front-End Lead)* - As Front-End Lead, Tobenna had a highly development based role on the project. This can be seen through their many contributions to the github repo in the form of changes and additions to the creation of the front end and GUI, as well as the search bar recommendation and stock by location features. Finally, Tobenna fulfilled their responsibilities as Front-End Lead with being responsible for the large majority of the development and management of the front-end and GUI.

Nathan Aguiar *(Back-End Lead)* - As Back-End Lead, Nathan had a highly development based role on the project. This can be seen through their many contributions to the github repo in the form of changes and additions to the creation of the back end and database. Nathan took care of the backend work, writing the entirety of the REST API, implementing the authentication system, setting up database connectivity as well as writing the initial backend tests. The goal with Nathan's work was to have a clean and elegant backend implementation with minimal business logic, opting to leave a lot of the lower level behaviors to Spring and Hibernate Search. Nathan also set up a Swagger API viewer to make the development of the frontend easier as well as to provide the ability to test the API prior to implementing the frontend for new changes. Finally, Nathan fulfilled their responsibilities as Back-End Lead with being responsible for the large majority of the development and management of the back-end and database.

Jerry Yang *(Software Quality Assurance Lead)* - As the Software Quality Assurance (SQA) Lead, Jerry played a role in ensuring the stability, performance, and reliability of the project. His responsibilities were heavily focused on rigorous testing and thorough documentation throughout each iteration. Jerry was in charge of creating comprehensive test plans, coordinating and executing test cases, and performing detailed regression testing at the end of each iteration. Jerry also identified, tracked, and verified the numerous bugs while collaborating closely with another team member to ensure code efficiency and adherence to best practices.

5. Demonstration of Functionality

Demonstration Video (Duration 13:07 minutes)

This demonstration video showcases the features implemented in our web application. The functionality of each feature is demonstrated with examples for each feature.

---

6. Bonus Modifier Statement

In a full live deployment of our program, the web-ui would be accessible by all store employees without needing them to run their own instance of the project. In the case of a retail chain, it would be possible to run a single instance at a headquarters and access it remotely via a mesh VPN such as Tailscale or by hosting the page on a publicly reachable port. The web frontend is a lot more intuitive to the average person than a native ui would likely be. A web frontend is also obviously infinitely more portable as it has 0 dependencies on the client and doesn't require installing any software which can't be said of a native ui or even a TUI.