Circuit Collective Iteration 3 Testing Plan

1. Introduction

1.1 Summary of Testing

The project's testing is led by QA lead Jerry and assisted by the rest of the Circuit Collective team. Unit, Integration, and System tests were performed using an automated testing framework to test functions individually and as part of the whole system. During this process, all use cases were tested in System Testing, covering all the tasks designed for the system.

1.2 Tested System Components

The core methods and functions of the project will be tested in the unit test, including:

- Inserting a game into the database
- Listing the games in the database
- Searching for a game in the database
- Fetching a game from the database

The following functions that were integrated into the catalog system will be tested in the Integration Tests:

- The Login System
- Web API on localhost
- Apache Lucene Search
- Inserting Item
- Delete Item
- Search Item
- Edit Item

The System Test will then test the following in context of the entire system:

- Logging In
- Inserting Items into the Catalog
- Searching for Items in the Catalog
- Deleting Items from the Catalog
- Editing Items in the Catalog

1.3 Testing Framework

The JUnit automated testing framework will be implemented to automate unit tests.

2. Unit tests

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

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

UT-02-TB	insert()	null	null	Automated (JUnit)	Jerry Nathan
UT-03-TB	list()	null	Game[0..*]	Automated (JUnit)	Jerry Nathan
UT-04-CB	search()	“Mario”	Game[0..*]	Automated (JUnit)	Jerry Nathan
UT-05-CB	search()	null	Game[0..*]	Automated (JUnit)	Jerry Nathan
UT-06-CB	search()	“”	Game[0..*]	Automated (JUnit)	Jerry Nathan

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

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

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

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

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

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

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

3. Integration Tests

Test ID	Components Involved	Preconditions	Steps	Expected Result	Assigned Team Member
IT-01-OB	Login System	Backend valid user logged in	Enter valid username and password, submit	Backend valid as login successful	Jerry Nathan William Bryce
IT-02-CB	Web API on localhost	Can be accessed by javascript on frontend	Request data in javascript, using fetch()	JSON responses depend on method called by fetch()	Nathan Tobenna
IT-03-OB	Apache Lucene Search	Algorithm run over list, give index based on string being searched	On backend, call search() of algorithm imported	Successful return list of index	Nathan
IT-04-TB	Inserting item	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
IT-05-TB	Delete item	Database remove item, update frontend item list	On html category web, click delete button right next to item display	Database successfully delete item	Jerry Nathan Tobenna

IT-06-TB	Search item	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
IT-07-TB	Edit item	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

4. System Tests

Test ID	Scenario	Preconditions	Steps	Expected Outcome	Assigned Team Member
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
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
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
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
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

5. Demonstration of Coverage

5.1 Covering User Stories

Listed below are the user stories the client created for this project, along with the tests that apply to that user story:

User Story	**Applicable System Tests**
As an inventory controller, I want a search bar, so software allows me to search games directly.	UT-04-CB search() UT-05-CB search() UT-06-CB search() IT-02-CB Web API on localhost IT-03-OB Apache Lucene Search ST-03-OB Searching
As a manager, I want an up to date stock catalogue, so software can keep track of which videogames are currently available.	IT-02-CB Web API on localhost IT-04-TB Insert Item IT-05-TB Delete Item IT-07-TB Edit Item ST-02-TB Inserting ST-04-TB Deleting ST-05-TB Editing
As a manager, I want to be able to keep track of shipments.	Low Priority, not yet implemented.
As a user, I want the search bar to automatically give a recommendation list based on what is typed in the search bar, so that makes game searching faster.	Low Priority, not yet implemented.
As a manager, I want a customer purchase log, so that I can manage them easier.	Low Priority, not yet implemented.
As an inventory controller, I want to give each game a tag, so organization can be easier.	UT-01-CB insert() IT-02-CB Web API on localhost IT-04-TB Insert Item IT-07-TB Edit Item ST-02-TB Inserting ST-04-TB Editing
As a user, I want to log in securely using the universal password	IT-01-OB Login System IT-02-CB Web API on localhost ST-01-CB Logging
As a manager, I want to track sales and revenue analytics, so that I can monitor a game's performance and relay that to developers.	UT-07-OB fetch UT-08-OB fetch UT-09-OB fetch UT-12-OB fetch UT-15-OB fetch

	IT-03-OB Apache Lucene Search IT-06-TB Search Item ST-03-OB Searching

5.2 High-Risk Components

We've identified the following high risk components, and discuss how they will be covered through testing:

- **Apache Lucene Search Engine**
  The Apache Lucene Search Engine is used to search for game objects. Because this open-sourced searching algorithm is not tailored to our project, it can cause errors if the user inserts invalid inputs. We will specifically test the algorithm in Iteration Test IT-03-OB, as a part of System Test ST-03-OB, and through unit tests UT-04-CB, UT-05-CB, and UT-06-CB to analyze and correct behaviour.

5.3 Test Case Summary

- *In summary, unit testing has a major quantity of test cases at fifteen and implemented our opaque box view tests. This is because we were testing various inputs across the insert() and fetch() functions, including fetch requests. The opaque box tests encapsulate testing of elements users do not see but still affect the system.*

- *Integration testing has fewer test cases at seven due to a relatively small number of integrations needed for our catalogue system and implemented translucent box tests. Translucent box tests encapsulate testing of elements users partly see.*

- *System Testing has the fewest case tests at five, and the use cases for our system are straightforward and streamlined. This result is because the use cases revolve around accessing and displaying items in the database. System Testing implemented our open box tests, which encapsulate testing of elements users fully see and interact with.*

5.4 Coverage Gaps

The gaps in our testing coverage revolve around the yet to be implemented user stories. This will be remedied with testing the features as their being built and implemented in the following iteration, ensuring they're tested for the delivered product.