SE 3XA3: Test Plan
Synergy Inventory Management System (SIMS)

Team #33, 'Sick Ideas'
Nathan Coit - 400022342
Lucas Shanks - 400029943
Cameron Van Ravens - 400020215

December 5, 2017

Contents

List of Tables

List of Figures

Table 1: **Revision History**

Date	**Version**	**Notes**
Oct. 27th, 2017	0.0	Initial Revision
Dec. 4th, 2017	1.0	Revision 1

1 General Information

1.1 Purpose

The purpose of this document is to provide a detailed outline in regards to the testing of the *Synergy Inventory Management System* (SIMS). However, due to the early derivation of the test cases, the proposed test plans are subject to change with future developments to the application at hand. These major changes will be outlined in the Revision History (see Table 1).

1.2 Scope

This project is a web-based inventory management system, intended to be accessed and used from a web browser. The scope of testing for such project will cover:

- The RESTful API
- The frontend functionality
- The backend functionality and responsiveness
- Any algorithms used
- The overall usability of the system on a variety of web browsers

1.3 Acronyms, Abbreviations, and Symbols

Please refer to Table 2 for abbreviations and acronyms, and refer to Table 3 for definitions.

Table 2: **Table of Abbreviations**

Abbreviation	**Definition**
SIMS	(Synergy Inventory Management System) The name of this application
RESTful API	RESTful (Representational State Transfer) Application Program Interface
CI/CD	Continuous Integration/Continuous Deployment
CSV	Comma Separated Values

1.4 Overview of Document

The following is a brief outline of what is covered in this document:

- The plan for software tests in section 2
- A description of system tests in section 3
- Tests for the Proof of Concept in section 4
- Comparison to the existing implementation section 5
- The unit testing plan in section 6

2 Plan

2.1 Software Description

Description of system found in Figure 1

2.2 Test Team

The test team consists of all three group members for writing and manually running test cases:

- Nathan Coit
- Lucas Shanks
- Cameron Van Ravens

Table 3: **Table of Definitions**

Term	**Definition**
URL	A string used to refer to a web-based address or request
Local Storage	Browser storage that is written to a cache for consistent data access
Database	A PostgreSQL relational database, running on a remote server
Angular App	A web application using the Angular 4 framework
Token	A string encrypted with user data which is used to verify user access permissions
Valid Token	A token that has been checked to be valid for this web application
HTTP	The standard web request protocol
HTTPS	The standard secure web request protocol
CSV	Comma Seperated Variable File
Page	A web page displayed to the user
JSON	(JavaScript Object Notation) A representation of objects in JavaScript
Frontend	This refers to development done on the visible HTML web pages
Backend	This refers to development done on the functional side of the site, including business logic, RESTful API, and data storage
Most Popular Browsers	This refers to up-to-date editions of Mozilla Firefox, Chrome, Opera, Internet Explorer, Microsoft Edge and Safari browsers

2.3 Automated Testing Approach

~~Automated testing will be done by setting up Continuous Integration on GitLab to run all the automatic tests whenever a change in detected in the **Staging** branch of the project. This will run all the tests setup with *Karma* and *Protractor*, reporting to the testing team if the tests were successful before merging the Staging codebase into the **Master** branch.~~ Automated testing was decided to be unfeasible and out of scope for our website design. Some automated tested shall be used to test the backend endpoints of the

Function	**Inputs**	**Outputs**	**Summary**
1. Login	Login Data	N/A	The login function of the system
2. Register	User Data	N/A	The registration function of the system for creating an account
3. View inventory	UserID	Inventory Page	The function to view a specific inventory page
4. Edit Inventory	ID and info	N/A	The function for editing, adding, and deleting items in an inventory
5. View User Data	UserID	User Details	The function to view a user's own details
6. Edit User Data	UserID and Data	New User Data	The function to edit a user's data in tbe system
7. Logout	UserID	N/A	The function for logging out of the system

Figure 1: Program Description

website.

2.4 Testing Tools

The tools used for testing are:

- ~~**Karma**: JavaScript and Typescript unit tests~~
- ~~**Protractor**: End-to-End frontend tests (Integration testing)~~
- ~~**Istanbul**: Code test coverage~~
- ~~**GitLab CI/CD**: Automated testing and deployment to server~~
- **LoadImpact**: To test the number of concurrent users the site can handle
- **Postman**: To test REST api endpoints

2.5 Testing Schedule

For the testing schedule, please refer to this ganttproject file.

3 System Test Description

3.1 Tests for Functional Requirements

3.1.1 REQ1 Testing

URL accessible from most browsers

1. test-00

   Type: Functional(dynamic, manual)

   Initial State: Database and system running.

   Input: HTTPS request from an up to date Chrome browser.

   Output: Angular app.

   How test will be performed: Enter the synergy inventory management system URL in an up to date Chrome browser and check that the angular app is returned.

2. test-01

   Type: Functional(dynamic, manual)

   Initial State: Database and system running.

   Input: HTTPS request from an up to date Firefox browser

   Output: Angular app

   How test will be performed: Enter the synergy inventory management system URL in an up to date Firefox browser and check that the angular app is returned.

3. test-02

   Type: Functional(dynamic, manual)

Initial State: Database and system running

Input: HTTPS request from an up to date Safari browser

Output: Angular app

How test will be performed: Enter the synergy inventory management system URL in an up to date Safari browser and check that the angular app is returned.

4. test-03

   Type: Functional(dynamic, manual)

   Initial State: Database and system running.

   Input: HTTPS request from an up to date opera browser

   Output: Angular app

   How test will be performed: Enter the synergy inventory management system URL in an up to date Opera browser and check that the angular app is returned.

5. test-04

   Type: Functional(dynamic, manual)

   Initial State: Database and system running

   Input: HTTPS request from an up to date internet explorer browser

   Output: Angular app

   How test will be performed: Enter the synergy inventory management system URL in an up to date Internet Explorer browser and check that the angular app is returned.

6. test-05

   Type: Functional(dynamic, manual)

   Initial State: Database and system running

   Input: HTTPS request to a missing URL

   Output: Angular app

How test will be performed: The test will be considered a pass if after entering a synergy inventory management system URL that is not specified in the system, either a 404 page is displayed or the user is redirected to the login page.

3.1.2 REQ2 Testing

Logged in user can view their data

1. test-10

   Type: Functional(dynamic, manual)

   Initial State: User has a valid login token in local storage and is on the user data page.

   How test will be performed: The test will be considered a pass if the page shows all the user data and the user data matches the data in the database.

2. test-11

   Type: Functional(dynamic, manual)

   Initial State: Use has a valid login token in local storage

   How test will be performed: The test will be considered a pass if a user with a valid token is not redirected upon navigating to the user data page.

3.1.3 REQ3 Testing

Login brings user to main overview or last page

1. test-20

   Type: Functional(dynamic, manual)

   Initial State: User not logged in and navigated to the login page

   How test will be performed: Test will be considered a pass if the user is navigated to the home page after successfully logging in.

2. test-21

   Type: Functional(dynamic, ~~automatic~~ manual)

   Initial State: User not logged in and redirected from the products page to the login page

   How test will be performed: Test will be considered a pass if the user is redirected to the products page after successfully logging in.

3. test-22

   Type: Functional(dynamic, manual)

   Initial State: User not logged in and redirected from the home page

   How test will be performed: Test will be considered a pass if the user is redirected to the home page after successfully logging in.

3.1.4 REQ4 Testing

Add items to database

1. test-30

   Type: Functional(dynamic, manual)

   Initial State: No items in table

   Input: A product name and a product description

   Output: A confirmation message

   How test will be performed: The test will be considered a pass if an item can be added to an empty table through the products page and the user is alerted that the item has successfully been added

2. test-31

   Type: Functional(dynamic, manual)

   Initial State: Items in table

   Input: A product name and product description

Output: A confirmation message

How test will be performed: The test will be considered a pass if the item was successfully added to the table through the product page and the user is alerted with a success message.

3.1.5 REQ5 Testing

Import items from CSV document

1. test-40

   Type: Functional(dynamic, ~~automatic~~ manual)

   Initial State: No items in database

   Input: Properly formatted CSV file from import page

   Output: confirmation message

   How test will be performed: The test will be considered a pass if all the items in the csv file match the items in the table.

2. test-41

   Type: Functional(dynamic, ~~automatic~~ manual)

   Initial State: Items in the user table

   Input: Properly formatted CSV file through import page

   Output: Confirmation message

   How test will be performed: The test will be considered a pass if all the items in the CSV file are present in the table as well as all the items that were in the table before the import.

3.1.6 REQ6 Testing

User shall be able to edit their data

1. test-50

   Type: Functional(dynamic, manual)

Initial State: User logged in and on the user details page

How test will be performed: Test will be considered a pass if the user details are editable through the user details page and these changes are saved into the database upon navigating away from the user details and then back.

3.1.7 REQ7 Testing

Application shall be mobile responsive

1. test-60

   Type: Functional(Dynamic, Manual)

   Initial State: Database and server running

   Input: Navigate to URL on an iPhone mobile device.

   Output: Angular app

   How test will be performed: Upon navigating to the SIMS URL on an IPhone mobile device, the angular app is displayed.

2. test-61

   Type: Functional(Dynamic, Manual)

   Initial State: Database and server running

   Input: Navigate to URL on a Samsung mobile device

   Output: Angular app

   How test will be performed: Upon navigating to the SIMS URL on a Samsung mobile device, the angular app is displayed.

3. test-62

   Type: Functional(Dynamic, Manual)

   Initial State: Database and server running

   Input: Navigate to URL on a BlackBerry mobile device.

   Output: Angular app

How test will be performed: Upon navigating to the SIMS URL on a Blackberry mobile device, the angular app is displayed.

3.1.8 REQ8 Testing

Multiple users viewing a single inventory page

1. test-70

   Type: System(dynamic, manual)

   Initial State: Two users of the same company logged in, one user currently has a specific inventory page open

   How test will be performed: The test will be considered a pass if the second user is able to view the same specific products page at the same time.

2. test-71

   Type: System(dynamic, manual)

   Initial State: A user logged in and on a specific products page, and multiple other users of the same company logged in.

   How test will be performed: The test will be considered a pass if up to 10 users from the same company are also able to view the specific inventory page at the same time

3.1.9 REQ9 Testing

Database shall have a recent backup

1. test-80

   Type: Functional(dynamic, manual)

   How test will be performed: Test will be considered a pass if a new database can be set up with a back up of the current database that matches the contents of the current database.

2. test-81

   Type: Functional(static, automatic)

   How test will be performed: Test will be considered a pass if the creation date of the newest backup is less than three days old.

3.2 Tests for Nonfunctional Requirements

3.2.1 Look and feel Requirements

User Feedback

1. test-90

   Type: Functional (manual)

   Initial State: N/A

   Input: A user at the landing page, a questionnaire, and a Test Team member to interview the user.

   Output: Detailed feedback from the user.

   How test will be performed: A user is tasked with setting up a basic account and navigating around the website in whatever manner they please. This test will be considered a pass if the majority of feedback returned is positive, while all feedback will still be reviewed.

3.2.2 Usability and Humanity Requirements

Questionnaires and Interviews

1. test-100

   Type: Functional (dynamic, manual)

   Initial State: N/A

   Input: A user at the website landing page and a questionnaire.

   Output: Detailed feedback from the user.

   How test will be performed: A user is tasked with setting up a basic account and exploring the primary features of the application.

Feedback will be obtained through the use of a questionnaire and later reviewed.

2. test-101

   Type: Functional (dynamic, manual)

   Initial State: N/A

   Input: A user at the website landing page.

   Output: Detailed feedback from the user.

   How test will be performed: A user is tasked with setting up a basic account and exploring the primary features of the application. During this time, the user will be interviewed and discuss website usability with one or more members of the Test Team.

3.2.3 Performance Requirements

Web Performance

1. test-110

   Type: Functional (manual)

   Initial State: N/A

   Input: A tester at the website landing page.

   Output: Performance information on the application

   How test will be performed: The application will be manually tested by the Test Team. All features of the website must be explored and shall respond to a user request in less than 5 seconds.

2. test-111

   Type: Functional (manual)

   Initial State: N/A

   Input: The angular application and LoadImpact

   Output: Performance data on the website

How test will be performed: The application will be automatically tested by LoadImpact. This will help to benchmark performance and run capacity tests to determine the maximum number of concurrent users that can be accommodated by the application with little change in expected performance.

3.2.4 Operational and Environmental Requirements

The Product shall be Available through most Popular Browsers

1. test-120

   Type: System (dynamic, manual)

   How test will be performed: The user will try to access the web application through the most popular browsers. If the web application displays and works similarly and as intended across all the tested browsers, then that is considered a pass.

3.2.5 Maintainability and Support Requirements

Application Maintainability

1. test-130

   Type: System (manual)

   Initial State: N/A

   Input: Angular app

   Output: Angular app

   How test will be performed: Through white-box testing, the entire application will be reviewed by the Test Team. Changes may be made so the software is more easily modifiable to suit the client's future needs or any future functionality that may be implemented. If the Test Team finds no updates that need to be made, then that is considered a pass.

3.2.6 Security Requirements

Users Must be Logged in to Access the Application

1. test-140

   Type: System (dynamic, manual)

   Initial State: User is logged out of the system, or does not have a valid token

   Input: User navigates to Inventory page

   Output: User is redirected

   How test will be performed: The user will attempt to access the any page (other than login and register) while they are logged out. The user being redirected to the login page will be considered a pass.

2. test-141

   Type: System (dynamic, manual)

   Initial State: User is logged out of the system, or does not have a valid token

   Input: User submits valid login information at the login page form

   Output: User is redirected

   How test will be performed: The user will submit valid login information while they are logged out. After successfully logging in, the user being redirected to the application home page will be considered a pass.

Users should only be able to View their own Inventory and Data

1. test-142

   Type: System (dynamic, manual)

   Initial State: User is logged in, or has a valid token

   Input: User visits the inventory page

Output: Items are displayed in items table

How test will be performed: The user will access their inventory page while they are logged in. If the items displayed in the items table are items with a company_id field matching the user's company_id field, then this is considered a pass.

3.3 Traceability Between Test Cases and Requirements

Req't	**Test Case ID**
REQ1	00 - 05
REQ2	10 - 11
REQ3	20 - 22
REQ4	30 - 31
REQ5	40 - 41
REQ6	50
REQ7	60 - 62
REQ8	70 - 71
REQ9	80 - 84

Figure 2: Requirement Traceability Matrix

4 Tests for Proof of Concept

4.1 POC Inventory Page Tests

4.1.1 Retrieval of Products

1. test-poc00

   Type: Integration(dynamic, manual)

   Initial State: No products are displayed, function is called

   Input: N/A

   Output: The list of products is displayed

   How test will be performed: The test will be considered a pass if the user visits the inventory page and is presented with all of the products in the database in a visible table.

2. test-poc01

   Type: Unit(dynamic, automatic)

Initial State: No products are returned, GET request made to endpoint

Input: GET /api/products

Output: Status 200 response, JSON formatted list of products.

How test will be performed: The unit test will make a GET request to /api/products, and will be considered a pass if the returned response status is 200 and all of the products currently in the database are returned in the response.

3. test-poc02

   Type: Unit(dynamic, automatic)

   Initial State: No products are returned, GET request made to endpoint

   Input: GET /api/products/:id

   Output: Status 200 response, JSON formatted product entry

   How test will be performed: The unit test will make a GET request to /api/products/:id, where id relates to an existing entry in the database. The test will be considered a pass if the returned response status is 200 and the product matching this id is returned in the response.

4.1.2 Creation of Products

1. test-poc10

   Type: Integration(dynamic, manual)

   Initial State: No products are in database, user enters data for a new product, the function is called

   Input: Enters data in appropriate fields on web page

   Output: The newly created product is stored in database and appears on screen

   How test will be performed: The test will be considered a pass if the user creates a new product in the Inventory page and after submitting the new product it appears in the visible products table.

2. test-poc11

   Type: Unit(dynamic, automatic)

   Initial State: No products are in database, POST request made to endpoint

   Input: POST /api/products with appropriate data in request body

   Output: Status 201 response, new entry in database

   How test will be performed: The unit test will make a POST request to /api/products with the appropriate data in the request body (formatted as JSON), and will be considered a pass if the response returns status 200 and the new product is stored in the database.

4.1.3 Updating Product Information

1. test-poc20

   Type: Integration(dynamic, manual)

   Initial State: At least one product in the database, user enters new data for a specific product, the function is called

   Input: Enters data into appropriate fields on web page

   Output: The new product information is stored in database and appears on screen

   How test will be performed: The test will be considered a pass if the user enters the modified product information in the correct form, and after submitting the new product information the changes are reflected in the visible products table.

2. test-poc21

   Type: Unit(dynamic, automatic)

   Initial State: At least one product in the database, PATCH request made to endpoint

   Input: PATCH /api/products/:id with appropriate data in request body

Output: Status 200 response, changes are stored in database

How test will be performed: The unit test will make a PATCH request to /api/products/:id, where id relates to an existing entry in the database, with the appropriate data in the request body (formatted as JSON), and will be considered a pass if the response returns status 200 and the changes to the product is stored in the database.

4.1.4 Deleting a Product

1. test-poc30

   Type: Integration(dynamic, manual)

   Initial State: At least one product in the database, user clicks on the garbage bin icon beside a product, the function is called

   Input: User clicks on garbage bin icon beside a product

   Output: The product is removed from the database, the change is reflected on the screen

   How test will be performed: The test will be considered a pass if the user clicks on the garbage bin icon beside a product in the visible products table, and the product is subsequently removed from the table.

2. test-poc31

   Type: Unit(dynamic, automatic)

   Initial State: At least one product in the database, DELETE request made to endpoint

   Input: DELETE /api/products/:id

   Output: Status 200 response, entry is removed from the database

   How test will be performed: The unit test will make a DELETE request to /api/products/:id, where id relates to an existing entry in the database, and will be considered a pass if the response returns status 200 and the product is removed from the database.

5 Comparison to Existing Implementation

In the existing implementation of this project there are no test cases or unit tests provided as it was not setup to include any automated testing, and therefore any testing was done manually. In this implementation, there will be test cases provided as well as support for automated unit and integration testing.

6 Unit Testing Plan

6.1 Unit testing of internal functions

Functions that do not change state variables shall be tested using a black box test method, where the desired output shall be checked against the actual output for a test set of inputs.
For functions that require or change state variables, white box testing will be used to check that the state variables have been changed to desired values after function calls.
For front end functions, stubs may be set up if no backend function has been set up to recieve the information sent. Also for backend functions, stubs and drivers may be set up to test functionality if no frontend functionality has been set up at the time.

7 Appendix

This is where you can place additional information.

7.1 Symbolic Parameters

No symbolic constants are used.

7.2 Usability Survey Questions

Usability Questionnare

What do you think the purpose of this website is?

How did you find the layout of the website?

On a scale of 1 to 10, with 10 being Very Appealing and 1 being Very Unappealing, how appealing did you find the website?

1 2 3 4 5 6 7 8 9 10

On a scale of 1 to 10, with 10 being Very Easy and 1 being Very Hard, how easy to use did you find the website?

1 2 3 4 5 6 7 8 9 10

On a scale of 1 to 10, with 10 being Very Fast and 1 being Very Slow, how responsive did you find the website?

1 2 3 4 5 6 7 8 9 10

If you could change one thing on the website, what would you change first?

Figure 3: Usability Questionnaire