Team PorkBelly

Personalised CRM Testing Plan

Document Version	Composed on	Author	Notes
1.0	28th September, 2021	Walter Van	Laying out initial plans.

The goal of this testing plan is to ensure that the product is functionally sufficient once deployed and sent to the user. From a software components perspective, the application consists of a human facing front-end, and server back-end.

Test Item	Version
UI prototype	Latest
Server Back-end	Latest
Back-end + Front-end	Latest

As the product is developed in an Agile fashion, the methodology of testing would follow in similar fashion. Tests would be done alongside the development of new features and completion of user stories. Hence, testing may be carried out as soon as those features are completed.

The Agile methodology provides a list of user stories that the application aims to accommodate. These have been extracted and compiled in the Requirement Elicitation document found here: https://github.com/chomosuke/IT-PROJECT-PorkBellyPro/blob/docs/docs/week6_artefacts/RequirementElicitation/RequirementElicitation.pdf.

From these user stories the following features to be tested have been identified:

Feature	Relevant User Story(s)
User registration feature	1
User login feature	2
Card detail display feature	6
Card detail edit feature	3, 4, 8, 9
Card deletion feature	5
Card keyword search feature	7
Tag attachment feature	10
Tag editing feature	12, 14
Tag deletion feature	13
Card tag filter feature	11
User data fetch feature	2

The follow features have been excluded from this testing plan:

Feature	Relevant User Story(s)	Reason
Card field-wise search	15	User story was determined to be optional.
Card Favourite preference feature	16	User story was determined to be optional.

Overall approach:

Each feature listed involves a back-end endpoint and front-end interface. The back-end would be tested automatically using the JEST test suite, however, the front-end may be tested manually due to the changes in layout and appearances during sprints. In addition, time constraints may require some automated testing to be carried out manually.

Testing Level	Testing Type	Target Item	Testing Method
N/A	Usability Testing	UI Prototype	Remote Interview, Manual
Acceptance tests	Functional, Usability	Back-end + Front-end	Manual
Integration tests	Functional, Security	Back-end	Automated, Manual
Unit tests	Functional	Back-end	Automated

Usability Testing:

Usability testing would be done using remote interviews with testers external to the team. This was done to receive feedback on the current (or upcoming) interfaces for the application. Testers would attempt to compete some tasks using the interface and their feedback would be gathered.

Task Number	Task Description	Features Tested*
1	Log in through the login page	User login feature
2	Look for the details of Walter Mart, state their contact email.	Card detail display feature
3	Edit some fields for the card in task 2	Card detail edit feature
4	Delete the card in task 2	Card deletion feature
5	Try to create and save a new card	Card detail edit feature
6	State where you may wish the search for cards	Card keyword search feature
7	Logout	General home page

Acceptance Tests:

The table below shows some acceptance tests to be carried out on the assembled application (back-end + front-end). While the table lists expected results, testers should record any unreported side effects / bugs.

Test Number	Test Name	Feature(s) Tested
AC1	Successful registration	User registration feature
AC2	Failed registration using non-unique username	User registration feature
AC3	Successful login	User login feature, User data fetch feature
AC4	Failed login with incorrect credentials	User login feature
AC5	View card detail	Card detail display feature
AC6	Successful creation of card	Card detail edit feature
AC7	Successful editing of card details	Card detail edit feature
AC8	Cancel new card details	Card detail edit feature
AC9	Successful deletion of card	Card deletion feature
AC10	Keyword search with matches	Card keyword search feature
AC11	Keyword search no matches	Card keyword search feature
AC12	Successful attachment of tag(s) to card	Tag attachment feature
AC13	Successful detachment of tag(s) from card	Tag attachment feature
AC14	Successful tag creation	Tag editing feature
AC15	Successful tag editing with standalone tag	Tag editing feature
AC16	Successful tag deletion with standalone tag	Tag deletion feature
AC17	Successful tag deletion with attached tag	Tag deletion feature
AC18	Successful tag editing with attached tag	Tag editing feature
AC19	Tag filter with matches	Card tag filter feature
AC20	Tag filter with no matches	Card tag filter feature
AC21	Tag filter removal	Card tag filter feature

Test Number	AC1
Test Name	Successful registration
Feature(s) Tested	User registration feature
Preconditions	User is on the registration page
Steps	• Enter a unique username into the username field • Enter any password into the password field • Click register button
Pass criteria	User is redirected to the login page.

Test Number	AC2
Test Name	Failed registration using non-unique username
Feature(s) Tested	User registration feature
Preconditions	User is on the registration page
Steps	• Enter a username that is already present in the database into the username field • Enter any password into the password field • Click register button
Pass criteria	User is shown an error dialog stating the failed registration.

Test Number	AC3
Test Name	Successful login
Feature(s) Tested	User login feature, User data fetch feature
Preconditions	User is on the login page
Steps	• Enter a username that is already present in the database into the username field • Enter the password associated with that username • Click login button
Pass criteria	User is directed to the home page displaying the list of cards they own.

Test Number	AC4
Test Name	Failed login with incorrect credentials
Feature(s) Tested	User login feature
Preconditions	User is on the login page
Steps	• Enter a username that is already present in the database • Enter a password different from that associated with the user • Click the login button • Enter a username that is not been registered • Enter any password • Click the login button
Pass criteria	Both sequences should result in user being displayed a error dialog stating a failed login.

Test Number	AC5
Test Name	View card detail
Feature(s) Tested	Card detail display feature
Preconditions	User is logged in and views their card list, AC3 is successful
Steps	• Select any card in the card list
Pass criteria	A panel is displayed showing the card's image (if any), the name, email, company, job title, phone number, attached tags, notes, and additional fields. Clicking any other card in the list replace the panel shown.

Test Number	AC6
Test Name	Successful creation of card
Feature(s) Tested	Card detail edit feature
Preconditions	User is on their home page, AC3 is successful
Steps	• User clicks the new card button • User enters the new details of the card • User clicks button to confirms the details
Pass criteria	The card panel is withdrawn, and the newly created card is permanently added to the database, as well as is appended to the card list on the home page.

Test Number	AC7
Test Name	Successful editing of card details
Feature(s) Tested	Card detail edit feature
Preconditions	User successfully achieves AC5
Steps	• User clicks edit card button • User changes some of the field value(s) of the card • User changes (or removes) the image of the card • User clicks the save button
Pass criteria	The card panel returns to its read-only state, displaying the newly updated card information. The database is permanently changed to reflect these changes and the changes persist on the website.

Test Number	AC8
Test Name	Cancel new card details
Feature(s) Tested	Card detail edit feature
Preconditions	User successfully achieves AC5
Steps	• User clicks edit card button • User changes some of the field value(s) of the card • User changes (or removes) the image of the card • User clicks the cancel button
Pass criteria	The card panel reverts to its previous state. Database is not called. Card detailed are left unchanged

Test Number	AC9
Test Name	Successful deletion of card
Feature(s) Tested	Card deletion feature
Preconditions	User successfully achieves AC5
Steps	• User clicks the edit card button • User clicks the card delete button • User confirms deletion on the confirmation dialog
Pass criteria	The panel is withdrawn. The card is permanently removed from the database. The card is removed from the card list on the home page even on refresh.

Test Number	AC10
Test Name	Keyword search with matches
Feature(s) Tested	Card keyword search feature
Preconditions	User is on the home page, AC3 is successful.
Steps	• User enters keywords (e.g name, email, phone number...) from a card into the search bar
Pass criteria	The card list shown is filtered to show only cards that contain those words entered. Card list returns to original state when the search bar is cleared

Test Number	AC11
Test Name	Keyword search with no matches
Feature(s) Tested	Card keyword search feature
Preconditions	User is on the home page: AC3 is successful
Steps	• User enters keywords not present on any card in the card list into the search bar
Pass criteria	The card list is filtered, no cards are shown. Cards return when keywords are removed

Test Number	AC12
Test Name	Successful attachment of tag(s) to card
Feature(s) Tested	Tag attachment feature
Preconditions	User is viewing a card's details: AC5 is successful
Steps	• User clicks on edit card button ○ (Optional) make some other changes to the card • User clicks on attach tags button • (Optional) user searches using tag labels • User clicks on tag(s) to attach to card • User clicks on save button
Pass criteria	Clicking on tags display them tags on the details panel. Cards are permanently updated upon clicking the save button. Changes persist when the card is revisited or page is refreshed.

Test Number	AC13
Test Name	Successful detachment of tag(s) from card
Feature(s) Tested	Tag attachment feature
Preconditions	User is viewing a card with tags on: AC5 is successful.
Steps	• User clicks on edit card button ○ (Optional) make addition changes to card • User clicks on the removal button on tag(s) • User clicks on save button
Pass criteria	Tags are removed from display on clicking removal button. Cards are permanently updated on saving. Changes persist even if the card is revisited or the page is refreshed.

Test Number	AC14
Test Name	Successful tag creation
Feature(s) Tested	Tag editing feature
Preconditions	User is viewing a card's details: AC5 is successful.
Steps	• User clicks on edit card button ○ (Optional) make changes to the card • User clicks on attach tags button • (Optional) User enters a new label name in search bar • User clicks on new tag button
Pass criteria	The tag list is appended with a new tag of optional label. This new tag is present in the database and persists on component refresh. The card's changes should remain as they are not saved nor cancelled.

Test Number	AC15
Test Name	Successful tag editing with standalone tag
Feature(s) Tested	Tag editing feature
Preconditions	User is viewing a card's details: AC5 is successful.
Steps	• User clicks on edit card button ○ (Optional) make changes to the card • User clicks on attach tags button • User clicks on edit tag for a tag of their liking • User changes the tag's label and / or colour • User closes the editing window.
Pass criteria	The select tag is updated in the database to reflect these new changes. The tag is updated with these new changes throughout the rest of the application (Home page). The card's changes should remain as they are not saved nor cancelled.

Test Number	AC16
Test Name	Successful tag deletion with standalone tag
Feature(s) Tested	Tag deletion feature

Preconditions	User is viewing a card's details: AC5 is successful
Steps	• User clicks on edit card button ○ (Optional) make changes to the card's details • User clicks on attach tags button • User clicks on edit tag button for a particular tag. • User clicks delete tag button
Pass criteria	Tag is deleted from the database permanently. Interface reflects to remove the deleted tag (Home page). The card's changes should remain as they are not saved nor cancelled.

Test Number	AC17
Test Name	Successful tag deletion with attached tag
Feature(s) Tested	Tag deletion feature
Preconditions	User is viewing a card's details with tags: AC5 is successful
Steps	• User clicks on edit card button ○ (Optional) make changes to card's details • User clicks on attach tags button • User clicks on edit tag button a tag attached to a card • User clicks on delete tag button
Pass criteria	Tag is deleted from the database permanently. Tag is removed from the card, as well as all the cards it had been attached too. Changes permeate to other parts of the application (e.g Home page). The optional changes made should remain as they are not saved nor cancelled.

Test Number	AC18
Test Name	Successful tag editing with attached tag
Feature(s) Tested	Tag editing feature
Preconditions	User is viewing a card's details with tags: AC5 is successful
Steps	• User clicks on edit card button ○ (Optional) make changes to card's details • User clicks on attach tags button • User clicks on edit tag button for a tag attached to a card • User changes tag label and / or colour • User closes the editing window
Pass criteria	Tag is updated in the database, and the tags on the user interface are updated accordingly. Changes persist even when page is refreshed. Optional changes to the cards, however, remain as they are not saved nor cancelled (although refreshing the page will automatically revert them).

Test Number	AC19
Test Name	Tag filter with matches
Feature(s) Tested	Card tag filter feature
Preconditions	User is on the home page: AC3 is successful
Steps	• User clicks on the tags belonging to some cards on the home page

Pass criteria	Selected tags appear in the search bar. Cards are filtered to display cards that contain the selected tags.

Test Number	AC20
Test Name	Tag filter with no matches
Feature(s) Tested	Card tag filter feature
Preconditions	User is on the home page: AC3 is successful
Steps	• User clicks on tags that belong to no card
Pass criteria	Selected tags appear in the search bar. Cards are filtered to display no cards.

Test Number	AC21
Test Name	Tag filter removal
Feature(s) Tested	Card tag filter feature
Preconditions	User is on the home page with tags filter active: AC3, AC20 or AC19 is successful
Steps	• User clicks on removal button on the tags in the search bar
Pass criteria	Removed tags disappear from the search bar.

Integration testing:
Integration testing could be either done 'manually' or through automated means. The integration strategy opted would be a 'Big bang' strategy where all units are assembled and tested in one go.

Integration Strategy	Big Bang
Test Procedure	Start up the database and back-end server. Send a series of requests to the server and monitor responses & database for effects.
Automated Test Suite (Local)	JEST with Supertest running locally on local database instances.
Automated Test Suite (CI)	JEST with Supertest running in workflow with a workflow instance of back-end server and database.
Manual Test Suite	POSTman to send API requests, MongoDB Compass to view changes.

Testing Requests:
Requests to the back-end should cover all RESTful api endpoints supported by the backend. Details on the api can be found at https://github.com/chomosuke/IT-PROJECT-PorkBellyPro/tree/docs/docs/api.

Test Number	Test Name	Feature(s) Tested
IN1	Successful registration	User registration feature
IN2	Registration with taken username	User registration feature
IN3	Successful user login	User login feature
IN4	User login with incorrect credentials	User login feature
IN5	Successful retrieval of user data	User data fetch feature
IN6	Unauthorised request to retrieve user data	User data fetch feature
IN7	Successful creation of card	Card detail edit feature
IN8	Creation of card with missing mandatory field data	Card detail edit feature
IN9	Successful updating of card of some fields	Card detail edit feature
IN10	Unauthorised updating of non-owned card.	Card detail edit feature
IN11	Successful updating of card tags	Tag attachment feature
IN12	Unauthorised attachment of non-owned tag to owned card	Tag attachment feature
IN13	Retrieval of card image	Card detail display feature
IN14	Unauthorised retrieval of card image	Card detail display feature
IN15	Successful deletion of card	Card deletion feature
IN16	Deletion of card not in database	Card deletion feature
IN17	Unauthorised deletion of non-owned card	Card deletion feature
IN18	Successful creation of tag	Tag editing feature

IN19	Creation of tag with missing field	Tag editing feature
IN20	Unauthorised creation of tag (missing user)	Tag editing feature
IN21	Successful updating of tag fields	Tag editing feature
IN22	Updating a tag that does not exist on the user	Tag editing feature
IN23	Successful deletion of tag	Tag deletion feature
IN24	Deleting a tag that does not exist on the user	Tag deletion feature
IN25	User logout	User login feature

Test Number	IN1	
Test Name	Successful registration	
Feature(s) Tested	User registration feature	
Preconditions	User with username "bob" is not present in system	
Example Request(s)	Resource	/api/register
	Method	POST
	Content-type	application/json
	Body	{ username: "bob", password: "hashedpassword" }
Pass criteria	Response returns status 201. Database contains a new user entry with username "bob" and password a hash of "hashedpassword".	

Test Number	IN2	
Test Name	Registration with taken username	
Feature(s) Tested	User registration feature	
Preconditions	User with username "marley" is present in system	
Example Request(s)	Resource	/api/register
	Method	POST
	Content-type	application/json
	Body	{ username: "marley", password: "anypassword"}
Pass criteria	Response with status 409.	

Test Number	IN3	
Test Name	Successful user login	
Feature(s) Tested	User login feature	
Preconditions	IN1 is successful: a user is saved on the database using username "bob" and password "hashedpassword".	
Example Request(s)	Resource	/api/login
	Method	POST
	Content-type	application/json

	Body	{ username: "bob", password: "hashedpassword" }
Pass criteria	Response with status 200. Cookie is set with name 'Token'. Token needs to have httponly field and secure set. (However, for manual, tests with this dependency these fields may be unset in the testing platform).	

Test Number	IN4	
Test Name	User login with incorrect credentials	
Feature(s) Tested	User login feature	
Preconditions	IN1 is successful: a user is saved on the database using username "jill" with password "jack"	
Example Request(s)	Resource	/api/login
	Method	POST
	Content-type	application/json
	Body	{ username: "jill", password: "jackie" }
	Resource	/api/login
	Method	POST
	Content-type	application/json
	Body	{ username: "bob", password: "jack" }
Pass criteria	Both response with status 401.	

Test Number	IN5	
Test Name	Successful retrieval of user data	
Feature(s) Tested	User data fetch feature	
Preconditions	IN3 is successful: browser holds 'Token' cookie from login as "Bill"	
Example Request(s)	Resource	/api/me
	Method	GET
	Content-type	
	Body	
Pass criteria	Response with 200 and returning details of user including the populated fields: username, settings, cards, and tags. (Refer to api documentation for shape)	

Test Number	IN6	
Test Name	Unauthorised request to retrieve user data	
Feature(s) Tested	User data fetch feature	
Preconditions	Browser does not hold 'Token' cookie, or cookie is expired	
Example Request(s)	Resource	/api/me
	Method	GET

	Content-type	
	Body	
Pass criteria	Response with status 401.	

Test Number	IN7	
Test Name	Successful creation of card	
Feature(s) Tested	Card detail edit feature	
Preconditions	IN3 is successful: 'Token' cookie is present from user 'Bill'	
Example Request(s)	Resource	/api/card
	Method	PUT
	Content-type	application/json

	Body	{
		"name": "Clear Image",
		"phone": "012",
		"email": "life@is.good",
		"jobTitle": "Handyman",
		"company": "wollies",
		"fields": [],
		"image" : <base64 encoding of JPEG image>
		"tags": []
		}
		(NB: image is an optional field)

Pass criteria	Response with status 201 with JSON body reflecting details provided. Given the example:
	{
	"id": <MongoDB ObjectId>,
	"favorite": false,
	"name": "Clear Image",
	"phone": "012",
	"email": "life@is.good",
	"jobTitle": "Handyman",
	"company": "wollies",
	"fields": [],
	"tags": []
	}
	
	New card document is created in the database associated with the id of 'Bill' in the User field.	

Test Number	IN8	
Test Name	Creation of card with missing mandatory field data	
Feature(s) Tested	Card detail edit feature	
Preconditions	IN3 is successful: 'Token' cookie is set to from login in as 'Bill'	
Example Request(s)	Resource	/api/card
	Method	PUT
	Content-type	application/json

	Body	{
		    "name": "Clear Image",
		    "phone": "012",
		    "email": "life@is.good",
		    "jobTitle": "Handyman",
		    "company": "wollies",
		    "fields": [],
		    "tags": []
		}
		(with any missing field(s) omitted: tags, fields, etc...)

Pass criteria	Response with status 400. New document is not created.	

Test Number	IN9	
Test Name	Successful updating of card of some fields	
Feature(s) Tested	Card detail edit feature	
Preconditions	IN3 is succesful: 'Token' cookie is present from logging in a 'Bill'. Database contains card of id with user field set Bill's id.	
Example Request(s)	Resource	/api/card
	Method	PATCH
	Content-type	application/json

	Body	{
		    "id": <Card id>,
		    "name": "Clear Image",
		    "phone": "012",
		    "email": "life@is.good",
		    "jobTitle": "Handyman",
		    "company": "wollies",
		    "fields": [{ "key" : "boss", "value": "billie" }],
		    "image": <base64 encoding of JPEG image>
		}
		(any field except id could be omitted)

Pass criteria	Response with status 200 with JSON body reflecting new values passed. Database document for the card is updated.	

Test Number	IN10	
Test Name	Unauthorised updating of non-owned card	
Feature(s) Tested	Card detail edit feature	
Preconditions	IN3 is successful: 'Token' cookie is set from logging in a 'Bob' Database contains card of id with associated user to 'Dylan'.	
Example Request(s)	Resource	/api/card
	Method	PATCH
	Content-type	application/json

	Body	{
		    "id": <Card id belonging to Dylan>,
		    "name": "Clear Image",
		    "phone": "012",
		    "email": "life@is.good",
		    "jobTitle": "Handyman",

		"company": "wollies", "fields": [{ "key" : "boss", "value": "billie" }], "image": <base64 encoding of JPEG image> } (any field except id could be omitted)
Pass criteria	Response with status 401.	

Test Number	IN11	
Test Name	Successful updating of card tags	
Feature(s) Tested	Tag attachment feature	
Preconditions	IN3 is successful: 'Token' cookie is set from logging in as 'Bill'. Card of id exists in database associated with Bill. Tag of id2 exists in database associated with Bill.	
Example Request(s)	Resource	/api/card
	Method	PATCH
	Content-type	application/json

	Body	{
		"id": <Card id belonging to Bill>
		"tags": [<Existing Tag ids> ,<New Tag id2>]
		}

Pass criteria	Response with status 200 with JSON body returning card details reflecting updated values. Database document is updated to reflect these.	

Test Number	IN12	
Test Name	Unauthorised attached to non-owned tag to owned card	
Feature(s) Tested	Tag attachment feature	
Preconditions	IN3 is successful: 'Token' cookie is set of from logging in as 'Bill'. Card of id exists in database associated with Bill. Tag of id2 exists in database associated with user 'Bob.'	
Example Request(s)	Resource	/api/card
	Method	PATCH
	Content-type	application/json

	Body	{
		"id": <Card id belonging to Bill>
		"tags": [<Existing Tag ids>, <Tag id2 belonging to Bob>]
		}

Pass criteria	Response with status 401.	

Test Number	IN13
Test Name	Retrieval of card image
Feature(s) Tested	Card detail display feature
Preconditions	IN5 is successful: Card data is collected containing image hash data. { id: <Card Object id>,

	... imageHash: <Hash String>, }	
Example Request(s)	Resource	/api/image/<hash string>
	Method	GET
	Content-type	
	Body	
Pass criteria	Response of status 200. Response sends an JPEG image in base64 encoding.	

Test Number	IN14	
Test Name	Unauthorised retrieval of card image	
Feature(s) Tested	Card detail display feature	
Preconditions	IN5 is successful: Card data belong to user X is collected. Image Hash Y is not part of it.	
Example Request(s)	Resource	/api/image/<Image hash Y>
	Method	GET
	Content-type	
	Body	
	(Alternatively, pass a hash that does not exist in database)	
Pass criteria	Response of status 404	

Test Number	IN15	
Test Name	Successful deletion of card	
Feature(s) Tested	Card deletion feature	
Preconditions	IN3 is successful: 'Token' cookie is set from logging in as "Bill". Card of id exists on database belonging to Bill	
Example Request(s)	Resource	/api/card
	Method	DELETE
	Content-type	application/json

	Body	{
		"id": <Card Id belonging to Bill>
		}

Pass criteria	Response of status 204. Card is removed from the database.	

Test Number	IN16
Test Name	Deletion of card not in database
Feature(s) Tested	Card deletion feature
Preconditions	IN3 is successful: 'Token' cookie is set from logging in as "Bill". Card of id does not exist in database.

Example Request(s)	Resource	/api/card
	Method	DELETE
	Content-type	application/json

	Body	{
		"id": <Card Id that doesn't exist>
		}

Pass criteria	Response of status 410.	

Test Number	IN17	
Test Name	Unauthorised Deletion of non-owned card	
Feature(s) Tested	Card deletion feature	
Preconditions	IN3 is successful: 'Token' cookie is set from logging in as "Bill". Card of id belongs to "Bob" exists in database.	
Example Request(s)	Resource	/api/card
	Method	DELETE
	Content-type	application/json

	Body	{
		"id": <Card Id belonging to Bob>
		}

Pass criteria	Response of status 410	

Test Number	IN18	
Test Name	Successful Creation of Tag	
Feature(s) Tested	Tag editing feature	
Preconditions	IN3 is successful: 'Token' cookie is set from logging in as "Bill"	
Example Request(s)	Resource	/api/tag
	Method	PUT
	Content-type	application/json

	Body	{
		"label": "Big Boss",
		"color": "red"
		}

	(Color field needs to be a valid CSS colour string)	
Pass criteria	Response with status 201 with JSON response body showing id, label and color. New Tag document is stored in the database with user field equal to "Bill" id.	

Test Number	IN19
Test Name	Successful Creation of Tag
Feature(s) Tested	Tag editing feature
Preconditions	IN3 is successful: 'Token' cookie is set from logging in as "Bill"

Example Request(s)	Resource	/api/tag
	Method	PUT
	Content-type	application/json

	Body	{
		"color": "red"
		}

	(Color field needs to be a valid CSS colour string)	
Pass criteria	Response with status 400.	

Test Number	IN20	
Test Name	Unauthorised creation of tag (missing user)	
Feature(s) Tested	Tag editing feature	
Preconditions		
Example Request(s)	Resource	/api/tag
	Method	PUT
	Content-type	application/json

	Body	{
		"label" : "Boss",
		"color": "red"
		}

	(Color field needs to be a valid CSS colour string)	
Pass criteria	Response with status 401, due to failed authentication.	

Test Number	IN21	
Test Name	Successful updating of tag fields	
Feature(s) Tested	Tag editing feature	
Preconditions	IN3 is successful: 'Token' cookie is set from logging in as "Bill". Tag of id exists on database belonging to Bill	
Example Request(s)	Resource	/api/tag
	Method	PATCH
	Content-type	application/json

	Body	{
		"id": <Tag id>,
		"label": "tooy",
		"color": "green"
		}

	(Fields label and color may be omitted)	
Pass criteria	Response with status 200 and JSON body showing updated details. Database document updated.	

Test Number	IN22
Test Name	Updating a tag that does not exist on the user
Feature(s) Tested	Tag editing feature

Preconditions	IN3 is successful: 'Token' cookie is set from logging in as "Bill". Tag of id does not exist.	
Example Request(s)	Resource	/api/tag
	Method	PATCH
	Content-type	application/json

	Body	{
		"id": <Tag id>,
		"label": "Fools",
		"name": "gold"
		}

	(Alternatively, id may exist but belonging to another user).	
Pass criteria	Response with status 410.	

Test Number	IN23	
Test Name	Successful deletion of tag	
Feature(s) Tested	Tag deletion feature	
Preconditions	IN3 is successful: "Token" cookie is set from logging in as "Bill". Tag of id exists on database belonging to Bill.	
Example Request(s)	Resource	/api/tag
	Method	DELETE
	Content-type	application/json

	Body	{
		"id": <Tag Id>
		}

Pass criteria	Response with status 200. Tag is removed from database and all previously associated cards in database.	

Test Number	IN24	
Test Name	Deleting a tag that does not exist on the user	
Feature(s) Tested	Tag deletion feature	
Preconditions	IN3 is successful: "Token" cookie is set from logging in as "Bill". Tag of id does not exist on database, or belongs to another user.	
Example Request(s)	Resource	/api/tag
	Method	DELETE
	Content-type	application/json

	Body	{
		"id": <Tag Id>
		}

Pass criteria	Response with status 410.	

Test Number	IN25
Test Name	User log out

Feature(s) Tested	User login feature	
Preconditions	IN3 is successful	
Example Request(s)	Resource	/api/logout
	Method	POST
	Content-type	
	Body	
Pass criteria	Response with status 200. "Token" cookie is cleared from environment.	

Unit tests:
Unit tests are applied to the back-end points for the routes defined in the repository. These Api routes can be found here:
https://github.com/chomosuke/IT-PROJECT-PorkBellyPro/tree/docs/docs/api.

These specific endpoints, are able to be tested at the unit level automatically using JEST. These tests are created in AGILE fashion, during their respective endpoint's implementation.