Test Plan and Document

COMP90082 2021 SM1 Software Project

For all teams in Project TO

Revision History

Date	Version	Description	Author
30/04/2021	1.2	Update format and 2.0-Covered_Requirement content	Yuwei Tsai
28/04/2021	1.1	Adjust format for document	Jin Kai Teh
24/04/2021	1.0	Initial draft	Andrea Law

Table of contents

Introduction

This document documents the test purposes, target audiences, test cases, and test data for Project TO of COMP90082 2021 SM1 Software Projects.

1.1 Proposal

The purpose of this document is to define and present the test cases for the Rocky Valley eCommerce Web Application project. These test cases cover frontend end-to-end testing, backend API testing, and acceptance tests implemented by all three teams(Rectify, Mobiusation, Pentagon).

Framework we are using is the V-model of the software development lifecycle.

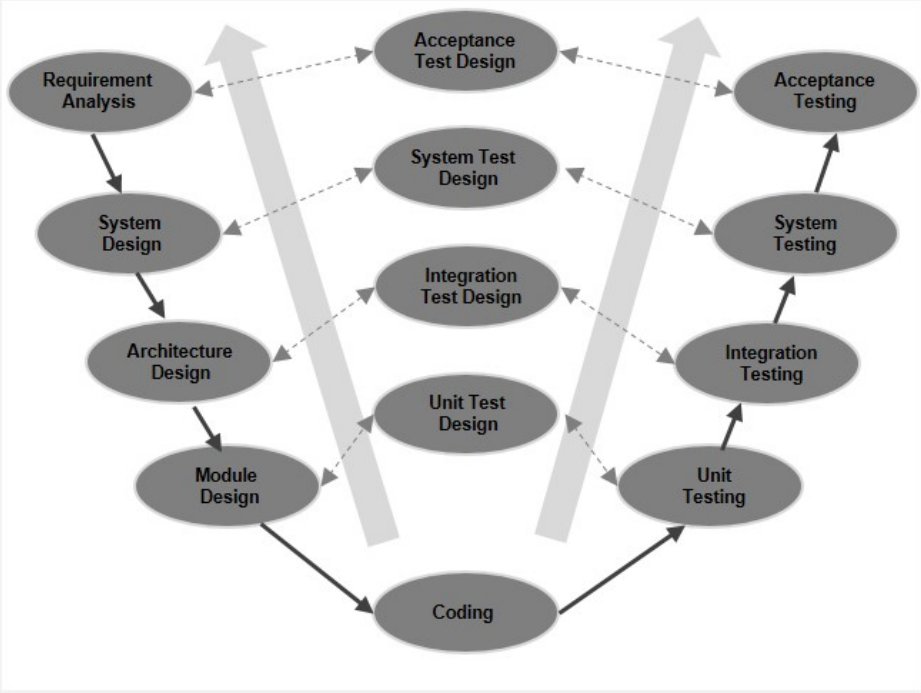

Image source: https://www.tutorialspoint.com/sdlc/sdlc_v_model.htm

1.2 Target audience

Intended audiences of this document include students working on Project TO and the teaching team from the University of Melbourne, clients (Squizz and Rocky Valley) and any other internal/external stakeholders that intend to review the final product of this project or to continue the development of the final product.

1.3 Conventions, terms and abbreviations

This section explains the concept of important terms used throughout the document. These terms are described in the following table, in alphabetical order.

Term	Description
Unit test	Test functions of the backend app, with a focus on API routers and handling
Integration test	Test integration ● between the frontend app and the backend app ● on the local environment ● on the cloud environment ● end-to-end testing
User acceptance test	Get acceptance from end-users on ● User interface design ● User experience design ● Workflow design
UC	Use case

1.4 Testing tools

- End-to-end integration test: Cypress, based on Javascript. Ref: Why Cypress
- Unit test: Pytest, based on Python. Ref: Pytest doc

Covered Requirements

This section lists the system requirements covered by the test cases.

2.1 Functional requirements summary

Maybe add a table here of all tests?

Tests

This section covers tests written for the project.

3.1 Frontend End-to-End Tests

Here is a summary of all the front-end end-to-end tests written with Cypress.

3.1.1 Scenario - User login

TEST CASE ID	TEST SCENARIO	TEST CASE	PRE-CONDITION	TEST STEPS	TEST DATA	EXPECTED RESULT	POST CONDITION	ACTUAL RESULT	STATUS (PASS/FAIL)
TC...	Verify login page	Login successfully as a customer	Need a valid user account to login	login page -> Enter customer		Able to login	passed	passed	passed

				name-> Enter password Click the login button					
	Verify login page	Login fail due to wrong password	Need an invalid user account to login	login page -> Enter customer name-> Enter password Click the login button Click the login button		Unable to login	failed	failed	passed
	Verify login page	Login successfully as a Staff	Need a valid staff account to login	login page -> Enter staff name-> Enter password Click the login button		Able to login	passed	passed	passed
	Verify login page	Login failed due to inexistent username	Need an invalid staff account to login	login page -> Enter staff name-> Enter password Click the login button		Unable to login	failed	failed	passed
	Verify login modal	Login successfully as a customer	Need a valid user account to login	Click login-> Enter customer name-> Enter password Click the login button		Able to login	passed	passed	passed
	Verify login modal	Login fail due to wrong password	Need an invalid user account to login	Click login -> Enter customer name-> Enter password Click the login button Click the login button		Unable to login	failed	failed	passed
	Verify login modal	Login successfully as a Staff	Need a valid staff account to login	Click login -> Enter staff name-> Enter password Click the login button		Able to login	passed	passed	passed
	Verify login modal	Login failed due to inexistent username	Need an invalid staff account to login	Click login -> Enter staff name-> Enter password Click the login button		Unable to login	failed	failed	passed

3.1.2 Scenario - User registration

TEST CASE ID	TEST SCENARIO	TEST CASE	PRE-CONDITION	TEST STEPS	TEST DATA	EXPECTED RESULT	POST CONDITION	ACTUAL RESULT	STATUS (PASS/FAIL)

TC...	Verify customer registration	Register as a customer by valid format successfully	Need a valid valid format of information	Register page -> Enter username-> Enter password-> Enter other informations according to fields-> Click submit		Able to register	passed	passed	passed
	Verify customer registration	Register as a customer using another valid format successfully	Need a valid valid format of information	Register page -> Enter username-> Enter password-> Enter other informations according to fields-> Click submit		Able to register	passed	passed	passed
	Verify customer registration	Register failed due to the wrong psw format	Need a invalid valid format of information	Register page -> Enter username-> Enter password-> Enter other informations according to fields-> Click submit		Unable to register	failed	failed	passed
	Verify customer registration	Register failed due to the wrong psw format again	Need a invalid valid format of information	Register page -> Enter username-> Enter password-> Enter other informations according to fields-> Click submit		Unable to register	failed	failed	passed
	Verify customer registration	Register failed due to the invalid email format	Need a invalid valid format of information	Register page -> Enter username-> Enter password-> Enter other informations according to fields-> Click submit		Unable to register	failed	failed	passed
	Verify customer registration	Register failed due to the existed email account	Need a invalid valid format of information	Register page -> Enter username-> Enter password-> Enter other informations according to fields-> Click submit		Unable to register	failed	failed	passed

	Verify customer registration	Register failed due to the existed username account	Need a invalid valid format of information	Register page -> Enter username-> Enter password-> Enter other informations according to fields-> Click submit		Unable to register	failed	failed	passed
	Verify staff registration	Register as a staff by valid format successfully	Need a valid format of information	Register page -> Enter username-> Enter password-> Enter other informations according to fields-> Click submit		Able to register	passed	passed	passed
	Verify staff registration	Register failed due to the wrong psw format	Need a invalid valid format of information	Register page -> Enter username-> Enter password-> Enter other informations according to fields-> Click submit		Unable to register	failed	failed	passed
	Verify staff registration	Register failed due to the wrong psw format again	Need a invalid valid format of information	Register page -> Enter username-> Enter password-> Enter other informations according to fields-> Click submit		Unable to register	failed	failed	passed
	Verify staff registration	Register failed due to the invalid email format	Need a invalid valid format of information	Register page -> Enter username-> Enter password-> Enter other informations according to fields-> Click submit		Unable to register	failed	failed	passed
	Verify staff registration	Register failed due to the existed email account	Need a invalid valid format of information	Register page -> Enter username-> Enter password-> Enter other informations according to fields-> Click submit		Unable to register	failed	failed	passed

	Verify staff registration	Register failed due to the existed staff username account	Need a invalid valid format of information	Register page -> Enter username-> Enter password-> Enter other informations according to fields-> Click submit		Unable to register	failed	failed	passed

3.1.3 Scenario - Customer identity pages browsing

TEST CASE ID	TEST SCENARIO	TEST CASE	PRE-CONDITION	TEST STEPS	TEST DATA	EXPECTED RESULT	POST CONDITION	ACTUAL RESULT	STATUS (PASS/FAIL)
TC...	Staff Home page	Enter into the staff home page	Need a valid staff account to login	login page->Enter user name->Enter password->Click login button->Check the navigation bar have the content that only belongs to the staff	{username: ruby, password:squizz}	The body of webpage should contain word “management”	passed	The body of webpage contains word “management”	passed
	Customer Home page	Enter into the customer home page	Need a valid customer account to login	login page->Enter user name->Enter password->Click login button->Check the body of webpage have the content that only belongs to the customer	{username:user1, password:squizz}	The body of webpage should contain word “package”	passed	The body of webpage contains word “package”	passed
	Profile page	Edit customer account profile successfully	Need a valid customer account to login	login page->Enter user name->Enter password->Click login button->Click name button->Click Account button->Click Profile->Click edit button->edit height to 175->change skill level to intermediate	{username:user1, password:squizz, height:170, skill level:Beginner}	The height value should be 175 and the skill level should be Intermediate.	passed	The value changs to 175 and Intermediate.	passed

	Profile page	The working condition of cancel button in customer profile page	Need a valid customer account to login	login page->Enter user name->Enter password->Click login->Click name button->Click Account->Click Profile->Click edit ->Click cancel	{username:user1, password:squizz}	The height value should be 170 and the skill level should be beginner.	passed	The value doesn't change.	passed
	Address page	Add address successfully	Need a valid customer account to login	Login->Click name button->Click Account->Click Address->Click add address->input 48 Bouverie St and 3053	{username:user1, password:squizz, state:VIC, city:melbourne, address:48 Bouverie St, postcode:3053}	The address web page should contain 48 Bouverie St.	passed	The address appears on the webpage.	passed
	Address page	Delete address successfully	Need a valid customer account to login	Login->Click name button->Click Account->Click Address->find 127 Swanson St and click delete address	{username:user1, password:squizz}	The address web page should not contain 127 Swanson St.	passed	The address doesn't appear on the webpage.	passed
	Address page	Edit address successfully	Need a valid customer account to login	Login->Click name button->Click Account->Click Address->find 125 Swanston St->Click edit->input 128 Swanston St	{username:user1, password:squizz, address:125 Swanston St,}	The address web page should contain 128 Swanston St.	passed	The address webpage contains 128 Swanston St.	passes
	Order History Page	Search the order by customer name	Need a valid customer account to login and have order	Login->Click Order History->input ruby->Click search	{username:user1, password:squizz}	The webpage should only contain ruby's order.	passed	The webpage just contains Ruby's order.	passed
	Order History Page	Cancel the order and change the status successfully	Need a valid customer account to login and have order	Login->Click Order History->input ruby->Click search->Click cancel of 1st line	{username:user1, password:squizz}	The webpage should not contain Done status.	passed	The webpage just contains 'New' status order record.	passed
	Add recipients	Add the details of the recipient.	Add a package into the shopping cart.	Login->Click Package->Click View Detail of 1st package->Click type of race track->Click add to cart->Click input recipient->Click add recipient->fill in the input boxes->Click add	{username:user1, password:squizz}	The webpage should not contain 'add recipient'.	passed	The webpage doesn't contain 'add recipient'.	passed

	Edit recipients	The details of the recipient have been changed.	Add a package into the shopping cart and details of the recipient.	Login->Click Package->Click View Detail of 1st package->Click type of race track->Click add to cart->Click input recipient->Click edit recipient->fill in the input boxes->Click edit	{username:user1, password:squizz}	Click edit recipient again, the content in the input box should contain the value that has been edited.	passed	The value changes.	passed

3.1.4 Scenario - Customer Booking Process

TEST CASE ID	TEST SCENARIO	TEST CASE	PRE-CONDITION	TEST STEPS	TEST DATA	EXPECTED RESULT	POST CONDITION	ACTUAL RESULT	STATUS (PASS/FAIL)
	Shopping process	Add a package into the shopping cart.	Need a valid customer account to login	Login->Click Package->Click View Detail of 1st package->Click type of race track->Click add to cart->Click continue ordering	{username:user1, password:squizz}	The switched webpage contains the 1st package name and the shopping cart web page should contain the 1st package name.	passed	Contain the right package name of both package detail webpage and shopping cart webpage.	passed
	Shopping process	The shopping cart could cache selected packages.	Add one package in the shopping cart..	Click Package->Click View Detail of 3rd package->Click type of race track->Click add to cart->Click continue ordering	{username:user1, password:squizz}	The switched webpage contains the 3rd package name and the shopping cart web page should contain above 2 package names.	passed	Contain the two right package names of the shopping cart webpage. And contain the right package name of the package detail webpage.	passed
	Shopping process	The shopping cart could cache selected packages and add the right packages.	Add two packages in the shopping cart..	Click Package->Click View Detail of 5th package->Click type of race track->Click add to cart->Click continue ordering	{username:user1, password:squizz}	The switched webpage contains the 5th package name and the shopping cart web page should contain above 3 package names.	passed	Contain the three right package names of the shopping cart webpage.And contain the right package name of the package detail webpage.	passed

	Shopping process	The shopping cart could cache selected packages and add the right packages.	Add three packages in the shopping cart..	Click Package->Click View Detail of 7th package->Click type of race track->Click add to cart->Click continue ordering	{username:user1, password:squizz}	The switched webpage contains the 7th package name and the shopping cart web page should contain above 4 package names.	passed	Contain the four right package names of the shopping cart webpage.And contain the right package name of the package detail webpage.	passed
	Shopping process	The shopping cart could cache selected packages and add the right packages.	Add four packages in the shopping cart..	Click Package->Click View Detail of 9th package->Click type of race track->Click add to cart->Click continue ordering	{username:user1, password:squizz}	The switched webpage contains the 9th package name and the shopping cart web page should contain above 5 package names.	passed	Contain the five right package names of the shopping cart webpage.And contain the right package name of the package detail webpage.	passed
	Shopping process	The shopping cart could cache selected packages and add the right packages.	Add five packages in the shopping cart..	Click Package->Click View Detail of 11th package->Click type of race track->Click add to cart->Click input recipient	{username:user1, password:squizz}	The switched webpage contains the 11th package name and the shopping cart web page should contain above 6 package names.	passed	Contain the six right package names of the shopping cart webpage.And contain the right package name of the package detail webpage.	passed
	Hiring process	Check that the packages cached in the shopping cart could be successfully booked.	Add 6 packages into shopping carts	Click Package->Click View Detail->Click type of race track->Click add to cart->Click input recipient	{username:user1, password:squizz}	The checkout web page should only contain packages that are chosen.	passed	The checkout web page just contains the 6 packages that are chosen.	passed
	Hiring process	The contract could automatically generate after booking packages in the checkout page successfully.	Add 1 package and recipient	Login->Click Package->Click View Detail of 1st package->Click type of race track->Click add to cart->Click input recipient->Click add recipient->fill in the input boxes->Click add->Click checkout->Click print	{username:user1, password:squizz}	The web page should contain ‘print contract’.	passed	The webpage contains ‘print contract’.	passed

3.1.5 Scenario - Staff identity pages browsing

TEST CASE ID	TEST SCENARIO	TEST CASE	PRE-CONDITION	TEST STEPS	TEST DATA	EXPECTED RESULT	POST CONDITION	ACTUAL RESULT	STATUS (PASS/FAIL)
TC...	Staff identity browsing	Login as a staff correctly and click into Calendar page, confirm the correct url and content.	Need a valid staff account to login	login page -> Enter staff name-> Enter password Click the login button -> click the Calendar tab and make assertions	{username:ruby, password:squizz}	Able to login Able to preview the Calendar page and relevant content are work.	passed	passed	passed
	Staff identity browsing	Login as a staff correctly and click into Contract page, confirm the correct url and content.	Need a valid staff account to login	login page -> Enter staff name-> Enter password Click the login button -> click the Calendar tab and make assertions	{username:ruby, password:squizz}	Able to login Able to preview the Contract page and relevant content are work.	passed	passed	passed
	Staff identity browsing	Login as a staff correctly and click into Management page, confirm the correct url and content.	Need a valid staff account to login	login page -> Enter staff name-> Enter password Click the login button -> click the Calendar tab and make assertions	{username:ruby, password:squizz}	Able to login Able to preview the Management page and relevant content are work.	passed	passed	passed
	Staff identity browsing	Login as a staff correctly and click into other page but Home, then click the Home tab, confirm the redirection and content.	Need a valid staff account to login	login page -> Enter staff name-> Enter password Click the login button -> click the Calendar tab and make assertions	{username:ruby, password:squizz}	Able to login Able to preview direct to other pages and redirect back to home page.	passed	passed	passed

3.1.1 User authentication functional test cases

3.1.1.1 UC01: User login

TC01: Customer is able to log in

TC02: Staff is able to log in

3.1.2 Staff and Customer functional test cases

3.1.2.1 UC02: Equipment Packages display(during the hiring process)

TC03: Customers are able to view and select the equipment package.(in stock available)

TC04: Customers are able to view the category of which the equipment package belongs to.

TC05: Customers are able to view and select the “extra” table within the same page that is able to hire.

TC06: Customers are able to view and select the “trail_type” field within the same page that is able to select.

TC07: Customers are unable to view and select the equipment package.(out of stock)

TC08: Staff is able to view the fully functional equipment package page.

TC09: Staff is able to add a new package class from UI.

TC010: Staff is able to delete a package class from UI.

TC11: Staff is able to update a package content from UI.

3..2 UC03: **Calendar display**

TC12: Customers are able to view the stock status of their selected-specific packages on the calendar .
TC13: Staff are able to view the stock status of all the equipment and packages availability on the calendar.

3.2 Backend API Tests

3.2.1 Endpoint - /addresses

TEST CASE ID	TEST SCENARIO	TEST CASE	PRE-CONDITION	TEST STEPS	TEST DATA	EXPECTED RESULT	POST CONDITION	ACTUAL RESULT	STATUS (PASS/FAIL)
TC...	Get addresses by user id	Enter a valid user id and get all the addresses created by the user	-	-Create a test client -Client calls the addresses endpoint accepting filter conditions with a GET request -Client passes user id in the GET request -Backend returns a JSON in the response -Confirm the returned JSON contains addresses matching the filter condition	User_id = 1	-Returns a response code of 200 -Returns a list of addresses available in the database as a JSON string with User_id = 1	-	-Returns a response code of 200 -Returns a list of addresses available in the database as a JSON string with User_id = 1	PASS
	Create an address	Enter valid address data and create a new address with current user	-	-Create a test client -Client calls the addresses endpoint accepting address data with a POST request -Backend returns a JSON in the response -Confirm the returned JSON contains addresses	"state": "NSW", "city": "Sydney", "postcode": "2000", "address_line": "Test address line 1", "user_id": 1, "order_id": 0	-Returns a response code of 201 -Returns created address available in the database as a JSON string	-	-Returns a response code of 201 -Returns created address available in the database as a JSON string	PASS

				matching the filter condition					
	Update an address	Enter valid address data and update a new address with current address id	The address data with current address id is created	-Create a test client -Client calls the addresses endpoint accepting address data with a PUT request -Backend returns a JSON in the response -Confirm the returned JSON contains addresses	"state": "NSW", "city": "Newcastle", "postcode": "2267", "address_line": "Test address line 2", "user_id": 1, "order_id": 0	-Returns a response code of 202 -Returns updated address available in the database as a JSON string - Return "Updated"	-	-Returns a response code of 202 -Returns updated address available in the database as a JSON string - Return "Updated"	PASS
	Delete an address	Delete an address with current address id	The address data with current address id is created	-Create a test client -Client calls the addresses endpoint with a DELETE request	-	Returns a response code of 204	-	Returns a response code of 204	PASS
	Delete an address that does not exist	Delete an address with invalid address id	-	-Create a test client -Client calls the addresses endpoint with a DELETE request	Address_id = 0	Returns a response code of 204	-	Returns a response code of 204	PASS
	Update an address that does not exist						-		PASS

3.2.2 Endpoint - /auth

TEST CASE ID	TEST SCENARIO	TEST CASE	PRE-CONDITION	TEST STEPS	TEST DATA	EXPECTED RESULT	POST CONDITION	ACTUAL RESULT	STATUS (PASS/FAIL)
	Login								PASS
	Login with wrong password								PASS

3.2.3 Endpoint - /categories

TEST CASE ID	TEST SCENARIO	TEST CASE	PRE-CONDITION	TEST STEPS	TEST DATA	EXPECTED RESULT	POST CONDITION	ACTUAL RESULT	STATUS (PASS/FAIL)

3.2.4 Endpoint - /packages

TEST CASE ID	TEST SCENARIO	TEST CASE	PRE-CONDITION	TEST STEPS	TEST DATA	EXPECTED RESULT	POST CONDITION	ACTUAL RESULT	STATUS (PASS/FAIL)
TC...	Get all packages	Send a GET request to the **packages** endpoint	The endpoint handles a GET request	-Create a test client -Test client calls the **packages** endpoint with a GET request -Backend returns a JSON in the response -Confirm the returned JSON contains a package matching the filter condition	Packages available in the test database	-Returns a response code of 200 -Returns a list of packages available in the database as a JSON string	-	-Returns a response code of 200 -Returns a list of packages available in the database as a JSON string	PASS
	Get packages with a single filter	Send a GET request to the **packages** endpoint with a **query** parameter in a filter	The endpoint handles a GET request and support a **query** parameter in the filter	-Create a test client -Client calls the **packages** endpoint accepting filter conditions with a GET request -Client passes values into the query parameter in the GET request -Backend returns a JSON in the response -Confirm the returned JSON	Packages available in the test database where the name of at least one package has this term “Beginner Package - Adult” in its name	-Returns a response code of 200 -Returns a list of packages available in the database as a JSON string, where the first package contains the term “Beginner Package - Adult” in its name	-	-Returns a response code of 200 -Returns a list of packages available in the database as a JSON string, where the first package contains the term “Beginner Package - Adult” in its name	PASS

				contains a package matching the filter condition					
	Get packages with a single filter	Send a GET request to the **packages** endpoint with a **category_id** parameter in the filter	The endpoint handles a GET request and support a **category_id** parameter in a filter	-Create a test client -Client calls the **packages** endpoint accepting filter conditions with a GET request -Client passes values into the **category_id** parameter in the GET request -Backend returns a JSON in the response -Confirm the returned JSON contains a package matching the filter condition	Packages available in the test database where at least one package has a **category_id** = 1	-Returns a response code of 200 -Returns a list of packages available in the database as a JSON string, where the first package has a **category_id** = 1	-	-Returns a response code of 200 -Returns a list of packages available in the database as a JSON string, where the first package has a **category_id** = 1	PASS
	Get packages with a single filter	Send a GET request to the **packages** endpoint with a **skill_level_id** parameter in the filter	The endpoint handles a GET request and support a **skill_level_id** parameter in a filter	-Create a test client -Client calls the **packages** endpoint accepting filter conditions with a GET request -Client passes values into the **skill_level_id** parameter in the GET request -Backend returns a JSON in the response -Confirm the returned JSON contains a package matching the filter condition	Packages available in the test database where at least one package has a **skill_level_id** = 1	-Returns a response code of 200 -Returns a list of packages available in the database as a JSON string, where the first package has a **skill_level_id** = 1	-	-Returns a response code of 200 -Returns a list of packages available in the database as a JSON string, where the first package has a **skill_level_id** = 1	PASS
	Get packages with a single filter	Send a GET request to the **packages** endpoint with a **age_group_id** parameter in the filter	The endpoint handles a GET request and support a **age_group_id** parameter in a filter	-Create a test client -Client calls the **packages** endpoint accepting filter conditions with a GET request -Client passes values into the query parameter in the GET request	Packages available in the test database where at least one package has a **age_group_id** = 1	-Returns a response code of 200 -Returns a list of packages available in the database as a JSON string, where the first package has a **age_group_id** = 1	-	-Returns a response code of 200 -Returns a list of packages available in the database as a JSON string, where the first package has a **age_group_id** = 1	PASS

				-Backend returns a JSON in the response -Confirm the returned JSON contains a package matching the filter condition					
	Get packages with a single filter	Send a GET request to the **packages** endpoint with at least one parameter in a filter	The endpoint handles a GET request when it has no result to return	-Create a test client -Client calls the **packages** endpoint accepting filter conditions with a GET request -Client passes an invalid category_id = 0 into the GET request's filter -Backend returns a JSON in the response -Confirm the returned JSON is empty	Ensure all packages have a valid **category_id** starting from 0	-Returns a response code of 200 -Returns an empty JSON string	-	-Returns a response code of 200 -Returns an empty JSON string	PASS
	Get packages with **two** filters	Send a GET request to the **packages** endpoint with two parameters (**category_id**, **skill_level_id**) in a filter	The endpoint handles a GET request and support two parameters in a filter	-Create a test client -Client calls the **packages** endpoint accepting filter conditions with a GET request -Client passes values of more than one parameter into the GET request -Backend returns a JSON in the response -Confirm the returned JSON contains a package matching the filter condition	Packages available in the test database where at least one package has a **category_id** = 1 AND **skill_level_id** = 1	-Returns a response code of 200 -Returns a list of packages available in the database as a JSON string, where the first package has a **category_id** = 1 AND **skill_level_id** = 1	-	-Returns a response code of 200 -Returns a list of packages available in the database as a JSON string, where the first package has a **category_id** = 1 AND **skill_level_id** = 1	PASS
	Get packages with **three** filters	Send a GET request to the **packages** endpoint with two parameters (**category_id**, **skill_level_id**, **age_group_id**) in a filter	The endpoint handles a GET request and support three parameters in a filter	-Create a test client -Client calls the **packages** endpoint accepting filter conditions with a GET request -Client passes values of more than one parameter into	Packages available in the test database where at least one package has a **category_id** = 1 AND **skill_level_id** = 1 AND **age_group_id** = 1	-Returns a response code of 200 -Returns a list of packages available in the database as a JSON string, where the first package has a **category_id** = 1	-	-Returns a response code of 200 -Returns a list of packages available in the database as a JSON string, where the first package has a **category_id** = 1	PASS

				the GET request -Backend returns a JSON in the response -Confirm the returned JSON contains a package matching the filter condition		AND **skill_level_id** = 1 AND **age_group_id** = 1		AND **skill_level_id** = 1 AND **age_group_id** = 1	
	Create a package	Send a POST request to the packages endpoint to create a package with **valid** data	The endpoint handles a POST request	-Create a test client -Client calls the **packages** endpoint with a POST request and valid data for the new package -Backend returns a JSON in the response -Confirm the returned JSON contains a package matching the package data sent in the request	Valid package data to be sent in the request	-Returns a response code of 201 -Returns the package created in the database in a JSON string	-	-Returns a response code of 201 -Returns the package created in the database in a JSON string	PASS
	Create a package	Send a POST request to the packages endpoint to create a package with **invalid** data	The endpoint handles a POST request with invalid data	-Create a test client -Client calls the **packages** endpoint with a POST request and invalid data for the new package -Backend returns a JSON in the response -Confirm the returned JSON contains a package matching the package data sent in the request	Invalid package data to be sent in the request. Example of invalid data includes having NULL for mandatory fields such as category_id	-Returns a response code of 422	-	-Returns a response code of 422	PASS
	Delete a package	Send a DELETE request to the packages endpoint to delete an existing package	The endpoint handles a DELETE request	-A package exists in the database -Create a test client -Client calls the packages endpoint with a DELETE request and passes in the package id for the package to be deleted	A package in the database	-Returns a response code of 204	-	-Returns a response code of 204	PASS

				-Backend deletes the package in the database -Backend sends back a response of the result					
	Delete a package	Send a DELETE request to the packages endpoint to delete a package that does not exist	The endpoint handles a DELETE request with invalid data	-A package exists in the database -Create a test client -Client calls the packages endpoint with a DELETE request and passes in a package id = 0 (which is invalid and would never exist) for the package to be deleted -Backend deletes the package in the database -Backend sends back a response of the result	-	-Returns a response code of 204	-	-Returns a response code of 204	PASS
	Update a package								

3.2.5 Endpoint - /products

TEST CASE ID	TEST SCENARIO	TEST CASE	PRE-CONDITION	TEST STEPS	TEST DATA	EXPECTED RESULT	POST CONDITION	ACTUAL RESULT	STATUS (PASS/FAIL)

3.2.6 Endpoint - /users

TEST CASE ID	TEST SCENARIO	TEST CASE	PRE-CONDITION	TEST STEPS	TEST DATA	EXPECTED RESULT	POST CONDITION	ACTUAL RESULT	STATUS (PASS/FAIL)
	Get all users								PASS
	Create a new user								PASS
	Filter users								PASS
	Get user by id								PASS
	Update a user								PASS
	Create a user that username exists								PASS
	Create a user that email exists								PASS

3.3 User Acceptance Tests

3.3.1 Low fidelity prototype and workflow

TC14: Developers and potential end users are able to understand how the new system will work

Test type: Functional	**Execution Type:** Manual
Objective: To test the flow of the conversation, the layout of the elements, or the terminology and ensure the workflow is reasonable without critical mistakes.	
Setup: ● A developer draws the workflow on paper ● A facilitator presents it to other possible end users ● Observers see how the end users interpret the workflow and record the results for follow up discussions and improvements	
Expected Outcome: ● A list of feedback for improving the initial workflow and get it ready to draw the high fidelity version with more details	

Notes: ● Visit this page to understand how a low-fi prototype should be tested ● One student took the role as the facilitator and the facilitator ● Other students took the role as the observer ● Students within our team and from other teams who did not work on the prototype took the role as the end users
Test Result: ● A list of questions to the client for clarifying some of the details that were ambiguous ● A list of feedback to take care of in the high fidelity design

Screenshot:

3.3.2 High fidelity prototype and workflow

TC15: Developers and potential end users are able to visualize and experience how the new system would work

Test type: Functional	**Execution Type:** Manual
Objective: To examine usability questions in detail and make conclusions about the user behaviour.	
Setup: ● Developers clarify requirements with the client, including but not limited to design style, layout, data, business, and technical requirements ● A facilitator presents it to other possible end users ● Observers see how the end users interpret the workflow and record the results for follow up discussions and improvements	
Expected Outcome: ● A list of feedback for improving the initial workflow and get it ready to draw the high fidelity version with more details	
Notes:	

- Visit this page to understand how a high-fi prototype should be tested
- One student took the role as the facilitator and the facilitator
- Other students took the role as the observer
- Students within our team and from other teams who did not work on the prototype took the role as the end users

Test Result:

- Feedback from developers to make the final design compatible among teams
- Allow developers to visualize the design and workflows according to the prototype

Screenshot:

3.3.3 Final product demo

Test data

The data used in test cases, please refer to our dummy data within the db folder and the data sample folder

4.1 Login credential

Description: Login details required to access the website	
Field	**Value**
Username	user1
Password	squizz

4.2 Pricing model data

Source file name: CSV Filemaker Pricing.csv

Category	Sell Code	Description	day 1	day 2	day 3	day 4	day 5	day 6	day 7	day 8	Day 9
DH Beg	0	DH Budget skis, boots, poles	50	85	100	115	125	130	135	140	145
DH Int	DI	DH Inter skis, boots, poles	70	110	130	140	150	160	165	170	175
DH Per	DP	DH Perf skis, boots, poles	85	130	150	165	170	180	190	200	210
DH U14	DK	DH U15 skis, boots, poles	40	68	80	92	100	104	108	112	116
DH U6	DPS	DH U5 skis, boots, poles	25	42	50	58	63	65	68	73	
XC Cla	X	XC Classic Ski, Boots, poles	50	85	100	115	125	130	135	140	145
SB Perf	SP	SB Perf board and Boots	85	130	150	165	170	180	190	200	
XC Ska	XS	XC Skate, skis, boots, poles	60	100	120	130	140	160	165	170	

XC U14	XK	XC U15, skis, boots, poles	40	68	80	92	100	104	108	113	
XC U6	XCP	XC U5, skis boots, poles	25	42	50	58	63	65	68	73	
SB Beg	SB	SB & Boots - Budget	50	85	100	115	125	130	135	140	145
SB Only	SBO	Snowboard Only	40	75	90	105	115	120	125	130	
Per brd	SIO	Snowboard only - Inter	60	100	120	130	140	150	155	160	
SB U14	SK	SB and Boots U15	40	68	80	92	100	104	108	112	116
BC Bud	B	BC Budget Skis, boots, poles	50	85	100	115	125	130	135	140	145
BC Per	BP	BC Perf, skis, boots, poles	85	130	150	165	170	180	190	200	210
Tele	TeleP	Telemark skis, boots, poles	85	130	150	165	170	180	190	200	210
Yowies	Y	Yowies/snowshoes	25	35	45	50	55	60	66	70	73
Boots	boot	Ski/SB or BC boots only	20	35	45	50	55	60	65	70	
Beg ski	ski	DH skis/poles no boots	40	75	90	105	115	120	125	130	135
P or P	P	Pants Adult	20	30	35	40	45	50	55	60	
P&P	PP	Pants and Parka Adult	30	40	45	50	55	60	65	70	
P&PK	PPK	Pants and Parka/suit U6	18	23	27	31	35	39	43	46	
PorPK	PK	Pants or Parka U6	14	19	23	27	31	35	39	42	
PorPY	PY	Pants or Parka U14	18	23	27	31	35	39	43	46	
P&PY	PPY	Pants and Parka U14	24	34	39	44	49	53	57	60	
Apres	A	Apres Boots Adults	10	15	20	25	30	35	35	35	
ApresK	AK	Apres Boots U14	8	12	15	17	19	21	23	25	27
Helmet	H	Helmet	10	15	20	23	26	29	31	33	35
Helm U14	HK	Helmet kids	0	0	0	0	0	0	0	0	0
Wrist G	WG	Wrist Guard	5	10	12	14	16	18	20	22	24
Goggle	G	Goggles	10	15	20	23	26	29	31	33	35

Poles	Poles	Poles Only	5	10	15	20	25	30	35	40	
Helm/gogg	HG	Helmet & Goggles	15	25	35	40	44	48	50	52	
Helm/WG	HWG	Helmet & WG	15	25	32	37	42	47	51	53	
Diamond	CD	Diamond Chain Hire	20	35	45	55	60	65	70	75	80
Ladder	LC	Ladder Chain Hire	15	25	35	45	50	55	55	57	
Toboggan	TS	Toboggan	10	15	20	23	26	29	31	33	
Tob Carve	TC	Toboggan Carve	20	30	40	46	52	58	62	64	
Garmon	BPB	Backcountry Garmont/Scarpa Boot	35	55	70	80	90	100	110	120	
ski u14	SKIK	Ski only under 15	32	60	72	84	92	96	100	104	
ski u6	SKIPS	Ski Only under 6	20	37	45	52	57	60	63	66	
Bootu14	BK	Boot only Under 14	15	25	30	35	40	45	50		
Boot u6	BPS	Boot only Under 6	10	15	20	25	30	35	40		
Per ski	PSKI	Performance ski only	60	100	120	135	145	155	165	175	185
Int ski	ISKI	Intermediiate ski only	60	100	120	135	145	155	165	175	185
Int Boot	IB	Intermediate Boot only	30	40	50	55	60	65	70	75	80
SB Inter	SI	SB & Boots Interm	70	110	130	140	150	160	165	170	175
Pants	P	Pants Adults	20	30	35	40	45	50	55	60	
Jacket	J	Jacket Adults	20	30	35	40	45	50	55	60	
P & J	PP	Pants & Jacket Adults	30	40	45	50	55	60	65	70	
Pants U6	PK	Pants U6	14	19	23	27	31	35	39	42	
Jacket U6	JK	Jacket U6	14	19	23	27	31	35	39	42	
P&J U6	PPK	Pants & Jacket U6/Suit	18	23	27	31	35	39	43	46	
Pants U15	PY	Pants U15	18	23	27	31	35	39	43	46	
Jacket U15	JY	Jacket U15	18	23	27	31	35	39	43	46	

P&J U15	PPY	Pants & Jacket U15	24	34	39	44	49	53	57	60	
Per Boot	PB	Performance Boot Only	35	55	70	80	90	100	110	120	
NNNBC	B	Backcountry NNNBC	50	85	100	115	125	130	135	140	