COMP90082 2021 SM1 Software Project

Test Plan and Document for Project TO

by Team 1, Team 2, and Team 3

Version 1.3

Last update: 28 May 2021

Revision History

Date	Version	Description	Contributor
28/05/2021	1.3	Fill in test cases for frontend tests, backend tests, and user acceptance tests	Andrea Law, Haowen Shen, Xiande Wen, Yue Yue, Yuwei Tsai
30/04/2021	1.2	Update format and added Requirements section	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.

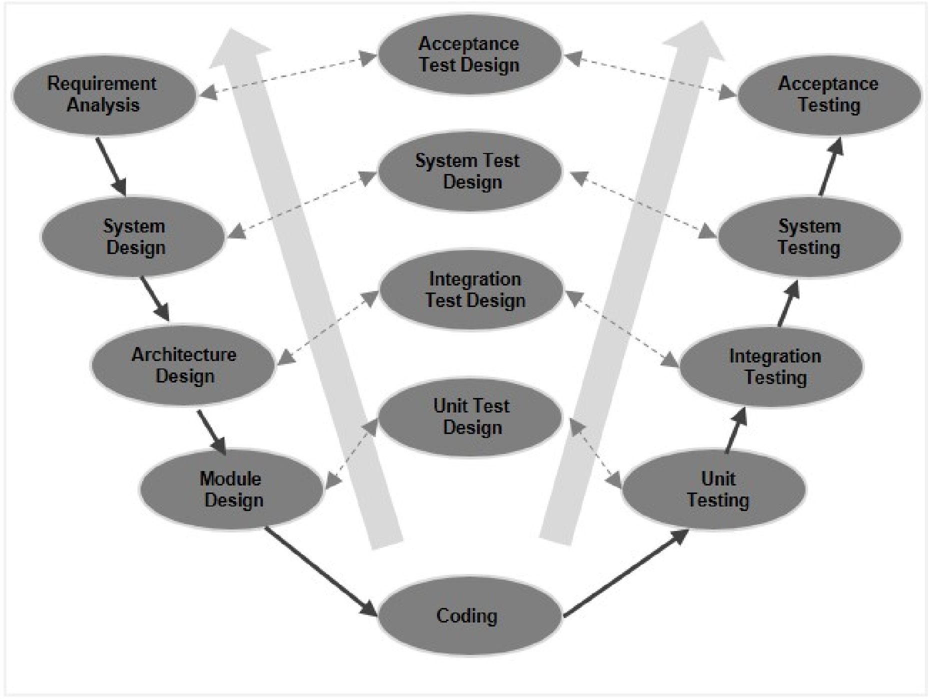

Image source: https://www.tutorialspoint.com/sdlc/sdlc_v_model.htm

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
  Cypress is a frontend testing tool built for the modern web. The tests could use it to write end-to-end tests, integration tests and unit tests. During using it, cypress also applies the method to debug the tests. The relative link is https://docs.cypress.io/guides/overview/why-cypress#Debugging-tests. Also,

cypress applies several methods to automatically generate the testing report by using Mocha for JavaScript testing framework, such as Json or Junit format report. Cypress would automatically make screenshots of failure test cases and video of entire test cases when run them.

- Unit test: Pytest, based on Python. Ref: Pytest doc

Functional Requirements

Requirements of the project could be found at https://github.com/ndhngoc91/COMP90082_S1_TO_2021/tree/master/documents/project_requirements

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.

Scenario - User registration

TEST CASE ID	TEST SCENARIO	TEST CASE	TEAM 1	TEAM 2	TEAM 3
TC_FE_1.1	Verify customer registration	Register as a customer by valid format successfully	Yue Yue		Yuwei Tsai
TC_FE_1.2	Verify customer registration	Register as a customer using another valid format successfully	Yue Yue		Yuwei Tsai
TC_FE_1.3	Verify customer registration	Register failed due to the wrong psw format	Yue Yue		Yuwei Tsai
TC_FE_1.4	Verify customer registration	Register failed due to the different value between psw and confirm ps	Yue Yue		Yuwei Tsai
TC_FE_1.5	Verify customer registration	Register failed due to the invalid email format	Yue Yue		Yuwei Tsai
TC_FE_1.6	Verify customer registration	Register failed due to the existed email account	Yue Yue		Yuwei Tsai
TC_FE_1.7	Verify customer registration	Register failed due to the existed username account	Yue Yue		Yuwei Tsai
TC_FE_1.8	Verify staff registration	Register as a staff by valid format successfully	Yue Yue		Yuwei Tsai
TC_FE_1.9	Verify staff registration	Register failed due to the wrong psw format	Yue Yue		Yuwei Tsai
TC_FE_1.10	Verify staff registration	Register failed due to the invalid email format	Yue Yue		Yuwei Tsai
TC_FE_1.11	Verify staff registration	Register failed due to the existed email account	Yue Yue		Yuwei Tsai
TC_FE_1.12	Verify staff registration	Register failed due to the existed staff username account	Yue Yue		Yuwei Tsai

Scenario - User login

TEST CASE ID	TEST SCENARIO	TEST CASE	TEAM 1	TEAM 2	TEAM 3

TC_FE_2.1	Verify login page	Login successfully as a customer	Yue Yue		Yuwei Tsai
TC_FE_2.2	Verify login page	Login fail due to wrong password	Yue Yue		Yuwei Tsai
TC_FE_2.3	Verify login page	Login successfully as a Staff	Yue Yue		Yuwei Tsai
TC_FE_2.4	Verify login page	Login failed due to inexistent username	Yue Yue		Yuwei Tsai
TC_FE_2.5	Verify login modal	Login successfully as a customer	Yue Yue		Yuwei Tsai
TC_FE_2.6	Verify login modal	Login fail due to wrong password	Yue Yue		Yuwei Tsai
TC_FE_2.7	Verify login modal	Login successfully as a Staff	Yue Yue		Yuwei Tsai
TC_FE_2.8	Verify login modal	Login failed due to inexistent username	Yue Yue		Yuwei Tsai

Scenario - Customer identity pages browsing

TEST CASE ID	TEST SCENARIO	TEST CASE	TEAM 1	TEAM 2	TEAM 3
TC_FE_3.1	Staff Home page	Enter into the staff home page	Yue Yue		Yuwei Tsai
TC_FE_3.2	Customer Home page	Enter into the customer home page	Yue Yue		Yuwei Tsai
TC_FE_3.3	Profile page	Edit customer account profile successfully	Yue Yue		Yuwei Tsai
TC_FE_3.4	Profile page	The working condition of cancel button in customer profile page	Yue Yue		Yuwei Tsai
TC_FE_3.5	Address page	Add address successfully	Yue Yue		Yuwei Tsai
TC_FE_3.6	Address page	Delete address successfully	Yue Yue		Yuwei Tsai
TC_FE_3.7	Address page	Edit address successfully	Yue Yue		Yuwei Tsai
TC_FE_3.8	Order History Page	Search the order by customer name	Yue Yue		Yuwei Tsai
TC_FE_3.9	Order History Page	Cancel the order and change the status successfully	Yue Yue		Yuwei Tsai
TC_FE_3.10	Add recipients	Add the details of the recipient.	Yue Yue		Yuwei Tsai
TC_FE_3.11	Edit recipients	The details of the recipient have been changed.	Yue Yue		Yuwei Tsai

Scenario - Customer Booking Process

TEST CASE ID	TEST SCENARIO	TEST CASE	TEAM 1	TEAM 2	TEAM 3
TC_FE_4.1	Shopping process	Add a package into the shopping cart.	Yue Yue		Yuwei Tsai
TC_FE_4.2	Shopping process	The shopping cart could cache selected packages.	Yue Yue		Yuwei Tsai
TC_FE_4.3	Shopping process	The shopping cart could cache selected packages and add the right packages.	Yue Yue		Yuwei Tsai
TC_FE_4.4	Shopping process	The shopping cart could cache selected packages and add the right packages.	Yue Yue		Yuwei Tsai
TC_FE_4.5	Shopping process	The shopping cart could cache selected packages and add the right packages.	Yue Yue		Yuwei Tsai
TC_FE_4.6	Shopping process	The shopping cart could cache selected packages and add the right packages.	Yue Yue		Yuwei Tsai
TC_FE_4.7	Hiring process	Check that the packages cached in the shopping cart could be successfully booked.	Yue Yue		Yuwei Tsai
TC_FE_4.8	Hiring process	The contract could automatically generate after booking packages in the checkout page successfully.	Yue Yue		Yuwei Tsai

<u>Scenario - Staff identity pages browsing</u>

TEST CASE ID	TEST SCENARIO	TEST CASE	TEAM 1	TEAM 2	TEAM 3
TC_FE_5.1	Staff identity browsing	Login as a staff correctly and click into Calendar page, confirm the correct url and content.	Yue Yue		Yuwei Tsai
TC_FE_5.2	Staff identity browsing	Login as a staff correctly and click into Contract page, confirm the correct url and content.	Yue Yue		Yuwei Tsai
TC_FE_5.3	Staff identity browsing	Login as a staff correctly and click into Management page, confirm the correct url and content.	Yue Yue		Yuwei Tsai
TC_FE_5.4	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.	Yue Yue		Yuwei Tsai
TC_FE_5.1	Staff identity browsing	Login as a staff correctly and click into Calendar page, confirm the correct url and content.	Yue Yue		Yuwei Tsai
TC_FE_5.2	Staff identity browsing	Login as a staff correctly and click into Contract page, confirm the correct url and content.	Yue Yue		Yuwei Tsai
TC_FE_5.3	Staff identity browsing	Login as a staff correctly and click into Management page, confirm the correct url and content.	Yue Yue		Yuwei Tsai
TC_FE_5.4	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.	Yue Yue		Yuwei Tsai

3.1.1 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_FE_1.1	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	{username: "test0", password: "123456sS", phone: "0402011031", email: "test0@gmail.com", firstName: "Yue", lastName: "Yue", birthday: '1996-06-05', gender: "Female", addressLine: "48 Bouverie St", state: 'VIC', city: "Melbourne", postcode: "3053", confirmPSW: "123456sS",}	Able to register	passed	passed	passed
TC_FE_1.2	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	{username: "test9", password: "123456sS", phone: "0402011031", email: "test9@gmail.com", firstName: "Yue", lastName: "Yue", birthday: '1996-06-05', gender: "Female", addressLine: "48 Bouverie St", state: 'VIC', city: "Melbourne", postcode: "3053", confirmPSW: "123456sS",}	Able to register	passed	passed	passed
TC_FE_1.3	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	{username: "test1", password: "litesting", phone: "0402011031", email: "test1@gmail.com", firstName: "Yue", lastName: "Yue", birthday: '1996-06-05', gender: "Female", addressLine: "48 Bouverie St",	Unable to register	failed	failed	passed

					state: 'VIC', city: "Melbourne", postcode: "3053", confirmPSW: "litesting",}				
TC_FE_1.4	Verify customer registration	Register failed due to the different value between psw and confirm ps	Need a invalid valid format of information	Register page -> Enter username-> Enter password-> Enter other informations according to fields-> Click submit	{username: "test2", password: "123456sS", phone: "0402011031", email: "test2@gmail.com", firstName: "Yue", lastName: "Yue", birthday: '1996-06-05', gender: "Female", addressLine: "48 Bouverie St", state: 'VIC', city: "Melbourne", postcode: "3053", confirmPSW: "squizzsS",}	Unable to register	failed	failed	passed
TC_FE_1.5	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	{username: "test3", password: "123456sS", phone: "0402011031", email: "test3", firstName: "Yue", lastName: "Yue", birthday: '1996-06-05', gender: "Female", addressLine: "48 Bouverie St", state: 'VIC', city: "Melbourne", postcode: "3053", confirmPSW: "123456sS",}	Unable to register	failed	failed	passed
TC_FE_1.6	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	{username: "test4", password: "123456sS", phone: "0402011031", email: "mail2@gmail.com", firstName: "Yue", lastName: "Yue", birthday: '1996-06-05', gender: "Female", addressLine: "48 Bouverie St", state: 'VIC', city: "Melbourne", postcode: "3053",	Unable to register	failed	failed	passed

					confirmPSW: "123456sS",}				
TC_FE_1.7	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	{username: "user1", password: "123456sS", phone: "0402011031", email: "xxx@gmail.com", firstName: "Yue", lastName: "Yue", birthday: '1996-06-05', gender: "Female", addressLine: "48 Bouverie St", state: 'VIC', city: "Melbourne", postcode: "3053", confirmPSW: "123456sS",}	Unable to register	failed	failed	passed
TC_FE_1.8	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	{username: "test5", phone: "0402011031", email: "test5@gmail.com", firstName: "Yue", lastName: "Yue", password: "123456sS", confirmPSW: "123456sS",}	Able to register	passed	passed	passed
TC_FE_1.9	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	{username: "test6", phone: "0402011031", email: "test6@gmail.com", firstName: "Yue", lastName: "Yue", password: "litesting", confirmPSW: "litesting",}	Unable to register	failed	failed	passed
TC_FE_1.10	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	{username: "test7", phone: "0402011031", email: "test7", firstName: "Yue", lastName: "Yue", password: "123456sS", confirmPSW: "123456sS",}	Unable to register	failed	failed	passed
TC_FE_1.11	Verify staff registration	Register failed due to the existed email account	Need a invalid valid format of information	Register page -> Enter username-> Enter password->	{username: "test8", phone: "0402011031", email:	Unable to register	failed	failed	passed

				Enter other informations according to fields-> Click submit	"mail1@gmail.com", firstName: "Yue", lastName: "Yue", password: "123456sS", confirmPSW: "123456sS",}				
TC_FE_1.12	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	{username: "ruby", phone: "0402011031", email: "test5@gmail.com", firstName: "Yue", lastName: "Yue", password: "123456sS", confirmPSW: "123456sS",}	Unable to register	failed	failed	passed

3.1.2 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_FE_2.1	Verify login page	Login successfully as a customer	Need a valid user account to login	login page -> Enter customer name-> Enter password Click the login button	{username: test0, password:123456sS }	Able to login	passed	passed	passed
TC_FE_2.2	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	{username: user1, password:iTesting}	Unable to login	failed	failed	passed
TC_FE_2.3	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	{username: ruby, password:squizz}	Able to login	passed	passed	passed
TC_FE_2.4	Verify login page	Login failed due to inexistent username	Need an invalid staff account to login	login page -> Enter staff name-> Enter password	{username: hello, password:123456}	Unable to login	failed	failed	passed

				Click the login button					
TC_FE_2.5	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	{username: test0, password:123456sS }	Able to login	passed	passed	passed
TC_FE_2.6	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	{username: user1, password:iTesting}	Unable to login	failed	failed	passed
TC_FE_2.7	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	{username: ruby, password:squizz}	Able to login	passed	passed	passed
TC_FE_2.8	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	{username: hello, password:123456}	Unable to login	failed	failed	passed

3.1.3 Scenario - Customer identity pages browsing

TEST CASE ID	TEST SCENARIO	TEST CASE	PRE-CONDITIO N	TEST STEPS	TEST DATA	EXPECTED RESULT	POST CONDITION	ACTUAL RESULT	STATUS (PASS/FAIL)
TC_FE_3.1	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

TC_FE_3.2	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
TC_FE_3.3	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
TC_FE_3.4	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
TC_FE_3.5	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
TC_FE_3.6	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
TC_FE_3.7	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	{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

				edit->input 128 Swanston St					
TC_FE_3.8	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
TC_FE_3.9	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
TC_FE_3.10	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
TC_FE_3.11	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)
TC_FE_4.1	Shopping process	Add a package into	Need a valid	Login->Click	{username:user1,	The switched	passed	Contain the right	passed

		the shopping cart.	customer account to login	Package->Click View Detail of 1st package->Click type of race track->Click add to cart->Click continue ordering	password:squizz}	webpage contains the 1st package name and the shopping cart web page should contain the 1st package name.		package name of both package detail webpage and shopping cart webpage.	
TC_FE_4.2	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
TC_FE_4.3	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
TC_FE_4.4	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
TC_FE_4.5	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
TC_FE_4.6	Shopping process	The shopping cart could cache selected packages and add the right	Add five packages in the shopping cart..	Click Package->Click View Detail of 11th package->Click	{username:user1, password:squizz}	The switched webpage contains the 11th package name and the	passed	Contain the six right package names of the shopping cart	passed

		packages.		type of race track->Click add to cart->Click input recipient		shopping cart web page should contain above 6 package names.		webpage.And contain the right package name of the package detail webpage.	
TC_FE_4.7	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
TC_FE_4.8	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 CONDITIO N	ACTUAL RESULT	STATUS (PASS/FAIL)
TC_FE_5.1	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:squiz z}	Able to login Able to preview the Calendar page and relevant content are work.	passed	passed	passed
TC_FE_5.2	Staff identity	Login as a staff correctly and	Need a valid staff	login page ->	{username:ruby	Able to login	passed	passed	passed

	browsing	click into Contract page, confirm the correct url and content.	account to login	Enter staff name-> Enter password Click the login button -> click the Calendar tab and make assertions	, password:squiz z}	Able to preview the Contract page and relevant content are work.			
TC_FE_5.3	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:squiz z}	Able to login Able to preview the Management page and relevant content are work.	passed	passed	passed
TC_FE_5.4	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:squiz z}	Able to login Able to preview direct to other pages and redirect back to home page.	passed	passed	passed

3.2 Backend API Tests

Here is a list of all the backend tests written in pytest.

Endpoint - /addresses

TEST CASE ID	TEST SCENARIO	TEST CASE	TEAM 1	TEAM 2	TEAM 3
TC_BE_1.1	Get addresses by user id	Enter a valid user id and get all the addresses created by the user	Haowen Shen		
TC_BE_1.2	Create an address	Enter valid address data and create a new address with current user	Haowen Shen		
TC_BE_1.3	Update an address	Enter valid address data and update a new address with current address id	Haowen Shen		
TC_BE_1.4	Delete an address	Delete an address with current address id	Haowen Shen		
TC_BE_1.5	Delete an address that does not exist	Delete an address with invalid address id	Haowen Shen		

TC_BE_1.6	Update an address that does not exist	Enter valid address data and update a new address with invalid address id	Haowen Shen		

Endpoint - /auth

TEST CASE ID	TEST SCENARIO	TEST CASE	TEAM 1	TEAM 2	TEAM 3
TC_BE_2.1	Login	Login with correct username and password	Haowen Shen		
TC_BE_2.2	Login	Login with correct username but wrong password	Haowen Shen		

Endpoint - /categories

TEST CASE ID	TEST SCENARIO	TEST CASE	TEAM 1	TEAM 2	TEAM 3
TC_BE_3.1	Get all **categories**	Send a GET request to the **categories** endpoint			Andrea Law

Endpoint - /packages

TEST CASE ID	TEST SCENARIO	TEST CASE	TEAM 1	TEAM 2	TEAM 3
TC_BE_4.1	Get all packages	Send a GET request to the **packages** endpoint			Andrea Law
TC_BE_4.2	Get packages with a single filter	Send a GET request to the **packages** endpoint with a **query** parameter in a filter			Andrea Law
TC_BE_4.3	Get packages with a single filter	Send a GET request to the **packages** endpoint with a **category_id** parameter in the filter			Andrea Law
TC_BE_4.4	Get packages with a single filter	Send a GET request to the **packages** endpoint with a **skill_level_id** parameter in the filter			Andrea Law
TC_BE_4.5	Get packages with a single filter	Send a GET request to the **packages** endpoint with a **age_group_id** parameter in the filter			Andrea Law
TC_BE_4.6	Get packages with a single filter	Send a GET request to the **packages** endpoint with at least one parameter in a filter and returns nothing			Andrea Law
TC_BE_4.7	Get packages with **two**	Send a GET request to the **packages** endpoint with two parameters (**category_id**, **skill_level_id**) in a filter			Andrea Law

	filters				
TC_BE_4.8	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			Andrea Law
TC_BE_4.9	Create a package	Send a POST request to the packages endpoint to create a package with **valid** data			Andrea Law
TC_BE_4.10	Create a package	Send a POST request to the packages endpoint to create a package with **invalid** data			Andrea Law
TC_BE_4.11	Delete a package	Send a DELETE request to the packages endpoint to delete an existing package			Andrea Law
TC_BE_4.12	Delete a package	Send a DELETE request to the packages endpoint to delete a package that does not exist			Andrea Law
TC_BE_4.13	Update a package	Send a PUT request to the packages endpoint to update an existing package			Andrea Law
TC_BE_4.14	Update a package	Send a PUT request to the packages endpoint to update a non-existing package			Andrea Law

Endpoint - /products

TEST CASE ID	TEST SCENARIO	TEST CASE	TEAM 1	TEAM 2	TEAM 3
TC_BE_5.1	Get all **products**	Send a GET request to the **products** endpoint			Andrea Law

Endpoint - /users

TEST CASE ID	TEST SCENARIO	TEST CASE	TEAM 1	TEAM 2	TEAM 3
TC_BE_6.1	Get all users	Send a GET request to the users endpoint	Haowen Shen		
TC_BE_6.2	Create a new user	Send a POST request to the users endpoint to create a new user with valid data	Haowen Shen		
TC_BE_6.3	Filter users	Send a GET request to the user endpoint with either parameter username or email in a filter	Haowen Shen		
TC_BE_6.4	Get user by id	Send a GET request to the user endpoint with parameter user_id	Haowen Shen		
TC_BE_6.5	Update a user	Send a PUT request to the user endpoint to update a valid user	Haowen Shen		
TC_BE_6.6	Create a user that username exists	Send a POST request to the users endpoint to create a new user with duplicate username	Haowen Shen		
TC_BE_6.7	Create a user that email exists	Send a POST request to the users endpoint to create a new user with duplicate email	Haowen Shen		

Endpoint - /orders

TEST CASE ID	TEST SCENARIO	TEST CASE	TEAM 1	TEAM 2	TEAM 3
TC_BE_7.1	Get all orders	Send a GET request to the orders endpoint to get all orders		Xiande Wen	
TC_BE_7.2	Create a new order	Send a POST request to the orders endpoint to create an order with **valid** data		Xiande Wen	
TC_BE_7.3	Filter orders	Send a GET request to the **orders** endpoint with at least one parameter in a filter		Xiande Wen	
	Get one customer's order history	Send a GET request to the orders endpoint to get history orders with a customer id.		Xiande Wen	
TC_BE_7.4	Get details of one order	Send a GET request to the orders endpoint to get details of orders with an order id.		Xiande Wen	
TC_BE_7.5	Update an order	Send a PUT request to the orders endpoint to update a existing order with an order id		Xiande Wen	
TC_BE_7.6	Cancel an order	Send a DELETE request to the orders endpoint to delete a order that does not exist		Xiande Wen	

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_BE_1.1	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

TC_BE_1.2	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 matching the filter condition	"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
TC_BE_1.3	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
TC_BE_1.4	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
TC_BE_1.5	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
TC_BE_1.6	Update an address that does not exist	Enter valid address data and update a new address with invalid address id	-	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 404	-	Returns a response code of 404	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)
TC_BE_2.1	Login	Login with correct username and password	-	Create a test client -Client calls the auth endpoint accepting user data with a POST request -Backend returns a JSON in the response -Confirm the returned JSON contains the user	"username": "user1", "password": "squizz"	-Returns a response code of 200 -Returns the user available in the database as a JSON string with username = user1	-	-Returns a response code of 200 -Returns the user available in the database as a JSON string with username = user1	PASS
TC_BE_2.2	Login	Login with correct username but wrong password	-	Create a test client -Client calls the auth endpoint accepting user data with a POST request -Backend returns a JSON in the response -Confirm the returned JSON	"username": "user1", "password": "squizz1"	Returns a response code of 404	-	Returns a response code of 404	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)
TC_BE_3.1	Get all **categories**	Send a GET request to the **categories** endpoint	The endpoint handles a GET request	-Create a test client -Test client calls the **categories** endpoint with a GET request	Categories available in the test database	-Returns a response code of 200 -Returns a list of categories available	-	-Returns a response code of 200 -Returns a list of categories available	PASS

				-Backend returns a JSON in the response -Confirm the returned JSON contains a list of categories		in the database as a JSON string		in the database as a JSON string	

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_BE_4.1	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
TC_BE_4.2	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 contains a package matching the filter condition	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

TC_BE_4.3	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
TC_BE_4.4	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
TC_BE_4.5	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 -Backend returns a JSON in the response	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

				-Confirm the returned JSON contains a package matching the filter condition					
TC_BE_4.6	Get packages with a single filter	Send a GET request to the **packages** endpoint with at least one parameter in a filter and returns nothing	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
TC_BE_4.7	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
TC_BE_4.8	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 the GET request -Backend returns a	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 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

				JSON in the response -Confirm the returned JSON contains a package matching the filter condition		AND **age_group_id** = 1		AND **age_group_id** = 1	
TC_BE_4.9	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
TC_BE_4.10	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
TC_BE_4.11	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 -Backend deletes	A package in the database	-Returns a response code of 204	-	-Returns a response code of 204	PASS

				the package in the database -Backend sends back a response of the result					
TC_BE_4.12	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	No package could be found in the database using the package id in the request	-Returns a response code of 204	-	-Returns a response code of 204	PASS
TC_BE_4.13	Update a package	Send a PUT request to the packages endpoint to update an existing package	The endpoint handles a PUT request with a valid package id and package data	-A package exists in the database -Create a test client -Client calls the packages endpoint with a PUT request and passes in package data to be used for the update -Backend updates the package in the database -Backend sends back a response of the result	A package with the same package id as in the request	-Returns a response code of 202	-	-Returns a response code of 202	PASS
TC_BE_4.14	Update a package	Send a PUT request to the packages endpoint to update a non-existing package	The endpoint handles a PUT request with an invalid package id and valid package data	-A package exists in the database -Create a test client -Client calls the packages endpoint with a PUT request and passes in package data to be used for the update -Backend fails to update a package	No packages having the same package id as in the request	-Returns a response code of 404	-	-Returns a response code of 404	PASS

				that does not exist in the database -Backend sends back a response of the result					

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)
TC_BE_5.1	Get all **products**	Send a GET request to the **products** endpoint	The endpoint handles a GET request	-Create a test client -Test client calls the **products** endpoint with a GET request -Backend returns a JSON in the response -Confirm the returned JSON contains a list of **products**	Categories available in the test database	-Returns a response code of 200 -Returns a list of **products** available in the database as a JSON string	-	-Returns a response code of 200 -Returns a list of **products** available in the database as a JSON string	PASS

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)
TC_BE_6.1	Get all users	Send a GET request to the users endpoint	-	-Create a test client -Test client calls the users endpoint with a GET request -Backend returns a JSON in the response -Confirm the returned JSON contains a list of users	-	-Returns a response code of 200 -Returns a list of users available in the database as a JSON string	-	-Returns a response code of 200 -Returns a list of users available in the database as a JSON string	PASS

TC_BE_6.2	Create a new user	Send a POST request to the users endpoint to create a new user with valid data	-	-Create a test client -Client calls the users endpoint with a POST request and valid data for the new user -Backend returns a JSON in the response -Confirm the returned JSON contains a user matching the user data sent in the request	"username": "TestUsername1", "height": 180, "weight": 75, "foot_size": 300, "first_name": "TestFirstName", "last_name": "TestLastName", "gender": "Male", "birthday": "1990-01-01", "phone": "0481111111", "email": "test1@gmail.com" , "din": 1, "is_enabled": 1, "skill_level_id": 1, "user_type_id": 1, "password": "TestPassword1"	-Returns a response code of 201 -Returns the users available in the database as a JSON string with username = TestUsername1	-	-Returns a response code of 201 -Returns the users available in the database as a JSON string with username = TestUsername1	PASS
TC_BE_6.3	Filter users	Send a GET request to the user endpoint with either parameter username or email in a filter	-	-Create a test client -Test client calls the users endpoint with a GET request -Backend returns a JSON in the response -Confirm the returned JSON contains a list of users	-	-Returns a response code of 200 -Returns a list of users available in the database as a JSON string	-	-Returns a response code of 200 -Returns a list of users available in the database as a JSON string	PASS
TC_BE_6.4	Get user by id	Send a GET request to the user endpoint with parameter user_id	-	-Create a test client -Test client calls the users endpoint with a GET request -Backend returns a JSON in the response -Confirm the returned JSON contains the user	user1	-Returns a response code of 200 -Returns the user available in the database as a JSON string with username = user1	-	-Returns a response code of 200 -Returns the user available in the database as a JSON string with username = user1	PASS
TC_BE_6.5	Update a user	Send a PUT request to the user endpoint to update a valid user	The user data is created with current user_id	-Create a test client -Test client calls the users endpoint with a POST request -Backend returns a JSON in the	"username": "TestUsername2", "height": 180, "weight": 90, "foot_size": 300, "first_name":	-Returns a response code of 201 -Returns the user available in the database as a JSON string with	-	-Returns a response code of 201 -Returns the user available in the database as a JSON string with	PASS

				response -Confirm the returned JSON contains the updated user	"TestFirstName", "last_name": "TestLastName", "gender": "Male", "birthday": "1990-01-01", "phone": "0482222222", "email": "test2@gmail.com" , "din": 2, "is_enabled": 1, "skill_level_id": 1, "user_type_id": 1	username = TestUsername2		username = TestUsername2	
TC_BE_6.6	Create a user that username exists	Send a POST request to the users endpoint to create a new user with duplicate username	The username already exists in the database	-Create a test client -Client calls the users endpoint with a POST request and valid data for the new user -Backend returns a JSON in the response -Confirm the returned JSON contains a user matching the user data sent in the request	"username": "TestUsername3", "height": 180, "weight": 75, "foot_size": 300, "first_name": "TestFirstName", "last_name": "TestLastName", "gender": "Male", "birthday": "1990-01-01", "phone": "0481111111", "email": "test4@gmail.com" , "din": 1, "is_enabled": 1, "skill_level_id": 1, "user_type_id": 1, "password": "TestPassword4"	Returns a response code of 409	-	Returns a response code of 409	PASS
TC_BE_6.7	Create a user that email exists	Send a POST request to the users endpoint to create a new user with duplicate email	The email already exists in the database	-Create a test client -Client calls the users endpoint with a POST request and valid data for the new user -Backend returns a JSON in the response -Confirm the returned JSON contains a user matching the user	"username": "TestUsername5", "height": 180, "weight": 75, "foot_size": 300, "first_name": "TestFirstName", "last_name": "TestLastName", "gender": "Male", "birthday": "1990-01-01", "phone":	Returns a response code of 409	-	Returns a response code of 409	PASS

				data sent in the request	"0481111111", "email": "test3@gmail.com" , "din": 1, "is_enabled": 1, "skill_level_id": 1, "user_type_id": 1, "password": "TestPassword5"				

3.2.7 Endpoint - /orders

TEST CASE ID	TEST SCENARIO	TEST CASE	PRE-CONDITION	TEST STEPS	TEST DATA	EXPECTED RESULT	POST CONDITION	ACTUAL RESULT	STATUS (PASS/FAIL)
TC_BE_7.1	Get all orders	Send a GET request to the orders endpoint to get all orders	The endpoint handles a GET request.	-Create a test client -Test client calls the orders endpoint with a GET request -Bankend get all orders in the database and return them in JSON response	All orders in the database	- Returns a response code of 200 -Returns a list of orders in the database in JSON format	-	- Returns a response code of 200 -Returns a list of orders in the database in JSON format	PASS
TC_BE_7.2	Create a new order	Send a POST request to the orders endpoint to create an order with **valid** data	The endpoint handles a POST request with valid data	-Create a test client -Client calls the **orders** endpoint with a POST request and invalid data for the new order -Backend returns a JSON in the response -Confirm the returned JSON contains a order match order data sent in the request	Invalid order data to be sent in the request. Example of invalid data includes having NULL for mandatory fields such as order_id	Returns a response code of 422		Returns a response code of 422	PASS
TC_BE_7.3	Filter orders	Send a GET request to the **orders** endpoint with at least one parameter	The orders endpoint handles a GET request when it has no result to return	-Create a test client -Client calls the **orders** endpoint accepting filter	Ensure all packages have a valid **order_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

		in a filter		conditions with a GET request -Client passes an invalid order_id = 0 into the GET request's filter -Backend returns a JSON in the response -Confirm the returned JSON is empty					
	Get one customer's order history	Send a GET request to the orders endpoint to get history orders with a customer id.	The order endpoint handles a GET request	-Create a test client -Test client calls the **orders** endpoint with a GET request along with the client id -Backend returns a JSON string of all orders belonging to the client	All ordered order in the database	- Returns a response code of 200 -Returns a list of orders in the database	-	- Returns a response code of 200 -Returns a list of orders in the database	PASS
TC_BE_7.4	Get details of one order	Send a GET request to the orders endpoint to get details of orders with an order id.	The orders endpoint handles a GET request	-Create a test client -Test client calls the **orders** endpoint with a GET request along with the customer id -Backend returns a JSON string of all orders belonging to the order	Details of order in the database	- Returns a response code of 200 -Returns a list of order' details		- Returns a response code of 200 -Returns a list of order' details	PASS
TC_BE_7.5	Update an order	Send a PUT request to the orders endpoint to update a existing order with an order id	The orders endpoint handles a PUT request with an valid order id and valid order data	-A order exists in the database -Create a test client -Client calls the orders endpoint with a PUT request and passes in order data to be used for the update -Backend fails to update a order that does not exist in the database -Backend sends back a response of the result	No orders having the same order id as in the request	Return a response code of 404		Return a response code of 404	PASS

TC_BE_7.6	Cancel an order	Send a DELETE request to the orders endpoint to delete a order that does not exist	The orders endpoint handles a DELETE request with invalid data	-An order exists in the database -Create a test client -Client calls the orders endpoint with a DELETE request and passes in a order id = 0 (which is invalid and would never exist) for the order to be deleted -Backend deletes the order in the database -Backend sends back a response of the result	No order could be found in the database using the order id in the request	-Returns a response code of 204	-	-Returns a response code of 204	PASS

3.3 User Acceptance Tests

3.3.1 Low fidelity prototype and workflow

TC3.3.1: 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

3.3.2 High fidelity prototype and workflow

TC3.3.2: 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 ● We use Figma for this. The file could be found at https://www.figma.com/file/PVHSxmtcFDIzTyn5Zurz5e/Project-3-Workflow?node-id=0%3A1 ● One student took the role as 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

3.3.3 Final product demo

TC3.3.3: Developers show the final product to the client and end users

Test type: Functional	**Execution Type:** Manual
Objective: To demonstrate the user-friendliness, usability, and functionality of the final product to the client and end users to get a user acceptance.	
Setup: ● Facilitator schedules a demo session between the development team and the client ● Developers attend the demo, walkthrough the product, and show the client how to use it according to the project requirements received for the project ● Client asks questions and developers address the questions ● Develop receives feedback from the client and document them for future expansions	

Expected Outcome:

- A list of feedback for expanding the final product

Notes:

- One student took the role as the facilitator
- Other students took the role as the observer
- Recording of the demo could be found here:
    - The password for these recordings is ceftt8p@LLr!
    - Mp4 (172.3 MB): https://cloudstor.aarnet.edu.au/plus/s/7NVMH0ErPPIyJmr
    - M4a (14.2 MB): https://cloudstor.aarnet.edu.au/plus/s/akgd5Y1X58UIhwz

Test Result:

- Positive feedback received from the client for user acceptance tests. Result of the tests = PASS

Test data

The data used in test cases, please refer to the db_migration_scripts directory and the data sample directory on our Github repository of this project.