PLAYING IN A SANDBOX WITH AUTHENTICATED SECURITY

(700100)

700100: REPORT AND SHORT NOTE

TRUSTWORTHY COMPUTING ACW 3

18 MAY 2020

Farah Aly - 201505395

Table of Contents

Introduction

The main goal of this assignment is to set up an online shop while ensuring the buyer is safe. An online shop for the University of Hull Catering Service was to be built. The dummy system would have a selection of products displayed for its users for them to purchase. After setting up an order, they have an option to pay by cash at collection or to pay through PayPal.

PayPal offers a developer sandbox which can be used to set up dummy buyers and facilitators to test payment transactions. A lot of questions arise about the capabilities of the sandbox, the accounts set up and the weaknesses, security and trustworthiness of the system.

This technical document is divided into two parts. It describes APIs. What information they take, how they work and how the PayPal REST API functions and explains Authenticated Encryption Modes.

The first part is the report which is divided into 3 parts. It will go in depth into APIs, the PayPal Sandbox and the weaknesses in the system. The second part is the short notes which will be depicting the concept of authenticated encryptions modes, their generic composition methods and single-pass authenticated encryption modes. The report then has a conclusion, reference and appendices which contain the pseudo code, JavaScript sample code, the tests made and results, the feedback of the heuristic evaluation and a user manual.

Report

API

What Is an API

API stands for Application Programing Interface. It is an intermediate software that allows communication between two applications (Pearlman, 2016). An API is a set of protocols that let the main software applications to be used by another application safely. APIs can be thought of as public methods of an object-oriented program that interact with other elements on the application.

There are different types of API. APIs are divided into four categories: public APIs, partner APIs, internal APIs and composite APIs. Public or Open APIs are APIs that are made publicly available by the software developers. There are to be used free meaning that the software owners give universal access to their customers to integrate this API into their own systems. For example, Facebook's API allows third-party tools to post on their user's feed. Partner APIs are not publicly available. Developers will need specific rights or licenses in order to be able to implement them. Internal APIs are also called Private APIs. They are not meant to be used outside of the bound of the company that has developed it. They are integrated among the teams so that they are able to improve their products. The last type of APIs is Composite APIs. They are a combination of APIs that speed up the execution and improve the performance (RapiAPI Staff, 2019).

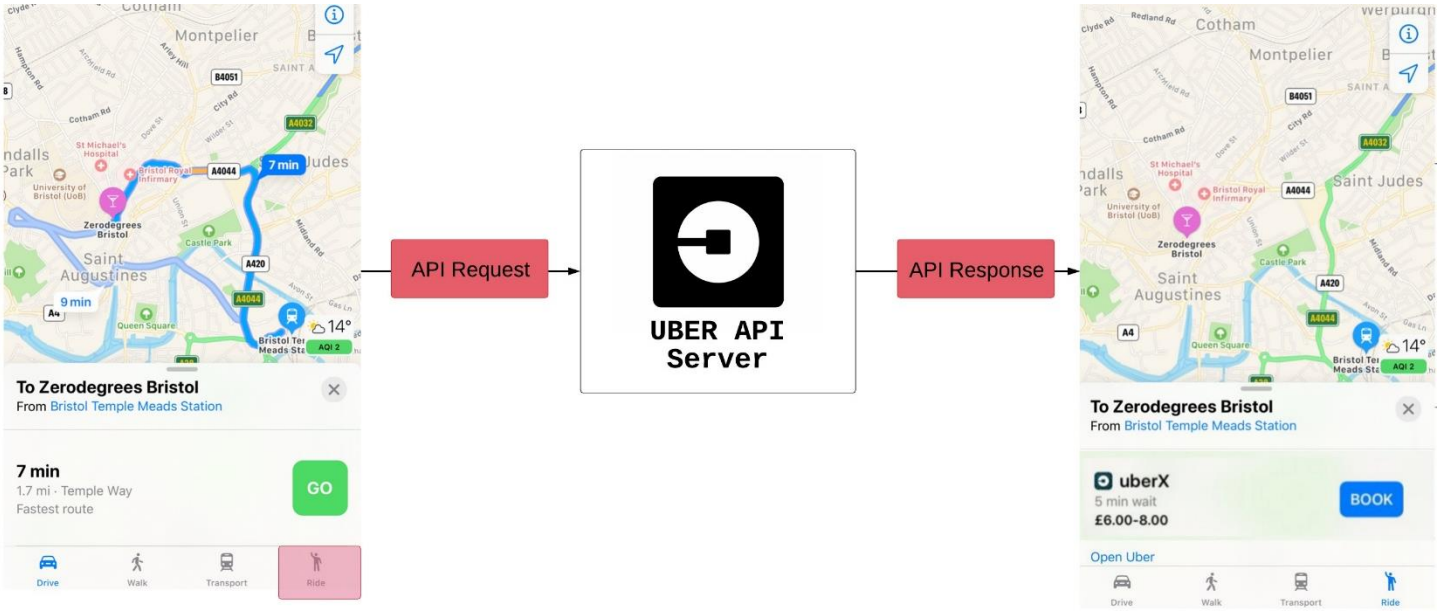
Certain protocols need to be followed in order to utilise these API types to their full potential. Protocols define a set of rules to be followed by the API calls. It specifies the accepted data types and commands for each API. There are two main protocols: SOAP and REST architectures.
SOAP stands for Simple Object Access Protocol. It is an API for web servers and browsers. This web API standardised how applications use network connections to manage services. Its downsides is that it came with strict rules. Compared to REST APIs, SOAP is more complicated to understand and less flexible.

REST stands for Representational State Transfer. It is a web API that takes advantage of existing protocols. It is used on many of the modern web applications such as Netflix, Uber and PayPal. An API is characterised as RESTful when it adheres to the following rules. A REST API is stateless. According to Richardson and Ruby (2007), "statelessness means that every HTTP request happens in complete isolation. When the client makes an HTTP request, it includes all information necessary for the server to fulfill that request. The server never relies on information from previous requests. If that information was important, the client would have sent it again in this request." A REST API also requires a uniform interface; meaning that the communication need to be done via hypertext transfer protocol using unique resources identifiers, CRUD and JSON conventions. It should be able to cache the response to improve the users' experience, meaning it would be faster and more efficient to use. This API should also support a layered architecture. Each layer should contribute to hierarchy and allow encapsulation (Castellani and Dorairajan, 2019).

How Does an API Work

The first application sends an API request in HTTP to the API server. The server then processes the requests and completes the commands before sending the results from the second application to the first application.

To explain the functionality, let's take the example of the Uber API used by Apple Maps. Apple Maps offers a 'Ride' option that lets the user select a ride services options like Uber. If the user tries to book an Uber through the Apple Maps app, the latter needs to get information from Uber such as the type of ride available, the number of rides available near the user, the price range of the rides and so on. The Uber API server receives those requests and responds by showing the needed info in the Apple Maps application. So, to show the user the price of an uberX ride from the university to the user's home, Apple would send an HTTP request to Uber with start_longtitude is the university and end latitude is home (see appendix A for pseudo-code). Uber would then send the information requested and display it in the Apple Maps (Uncubed, 2017).



Inside the PayPal Sandbox

The PayPal API is a public REST web API. PayPal offers tools that offers resources in order to test and implement this method of payment onto their platform. In order to accomplish that, the users need to set up their development environment and get their personal credentials.

In this section, steps to how to set up an account and how the PayPal API operates will be defined.

How to Set Up an Account

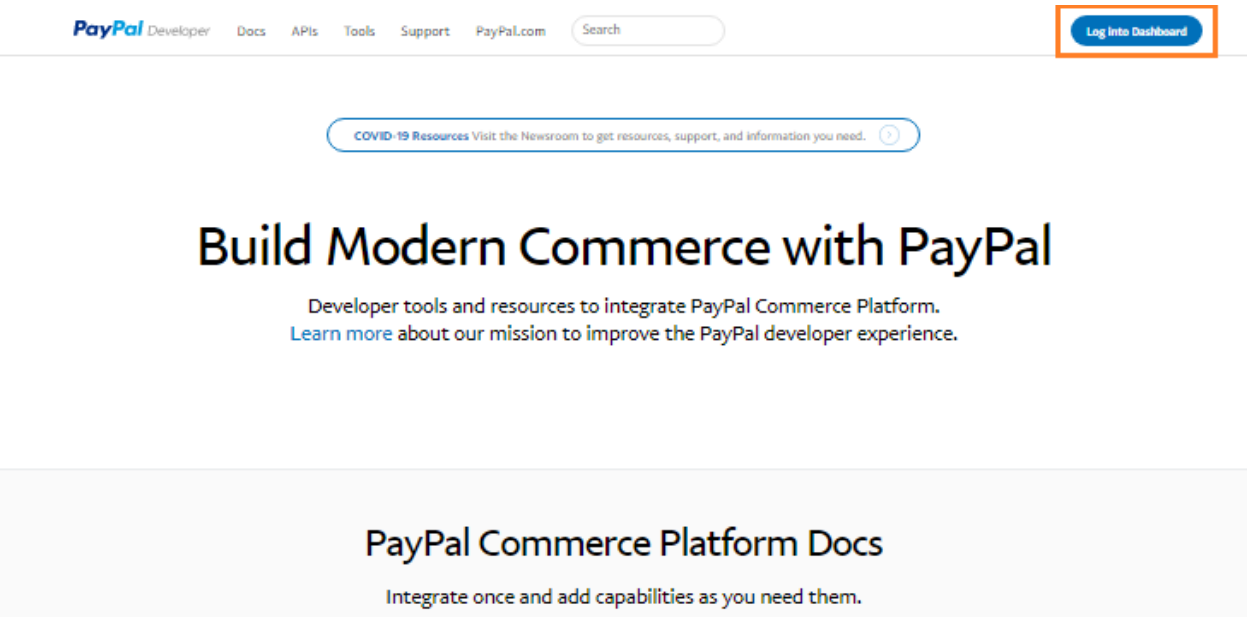
Setting up the PayPal environment occurs in three steps: getting credentials, getting an access token and creating sandbox accounts.

There are three type of accounts that can be acquired to get credentials and create sandbox accounts: a developer account, a personal account or a business account. While all these types can get the necessary credentials, they have different functionalities (PayPal, n.d.).

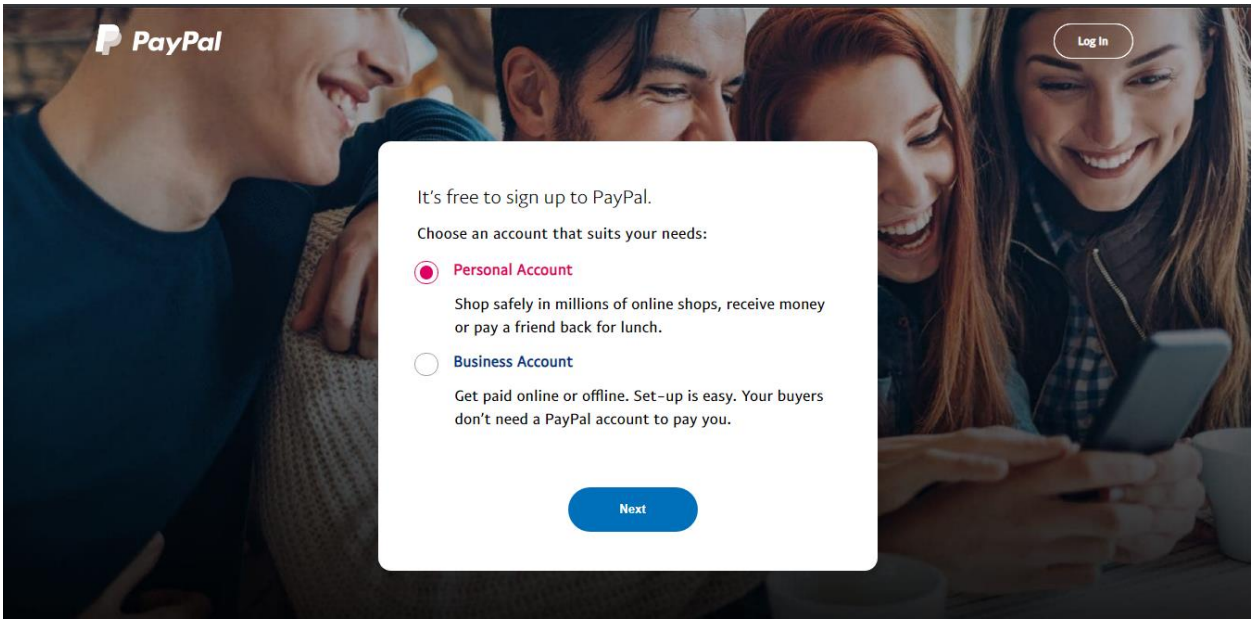
Capabilities	Developer account	Personal account	Business account
Access sandbox	X	X	X
Send and receive money		X	X
Go live			X

Sandbox accounts and app credentials were acquired into order to run purchases on the University of Hull Catering Service dummy system. The following are the steps followed so that the payments can be accomplished:

Step 1: Go to PayPal Developer's website. Click on the 'Log Into Dashboard' button.

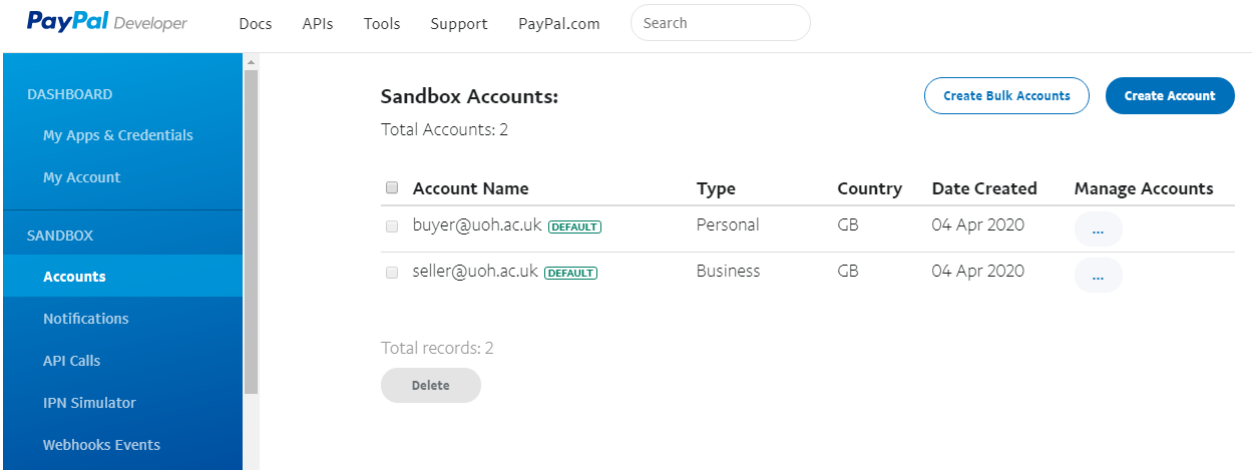


Step 2: You will be redirected to the Log In page. If you do not have an account or wish to have a new account for the developer platform, click Sign Up. You will be directed to the following page. Check for 'Personal Account'

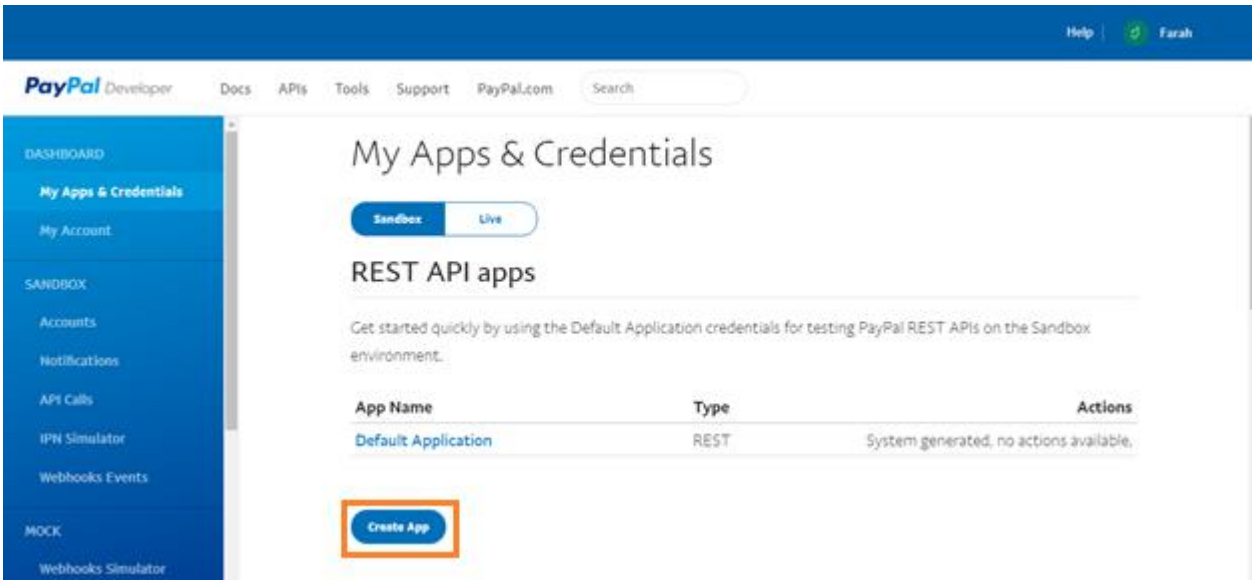


Step 3: After logging, you will be redirected to the Dashboard. The ‘My Apps & Credentials’ page is open by default. Here, you can see the REST API apps and credentials. One app is made by default. For this project, we will be creating new details for this project.

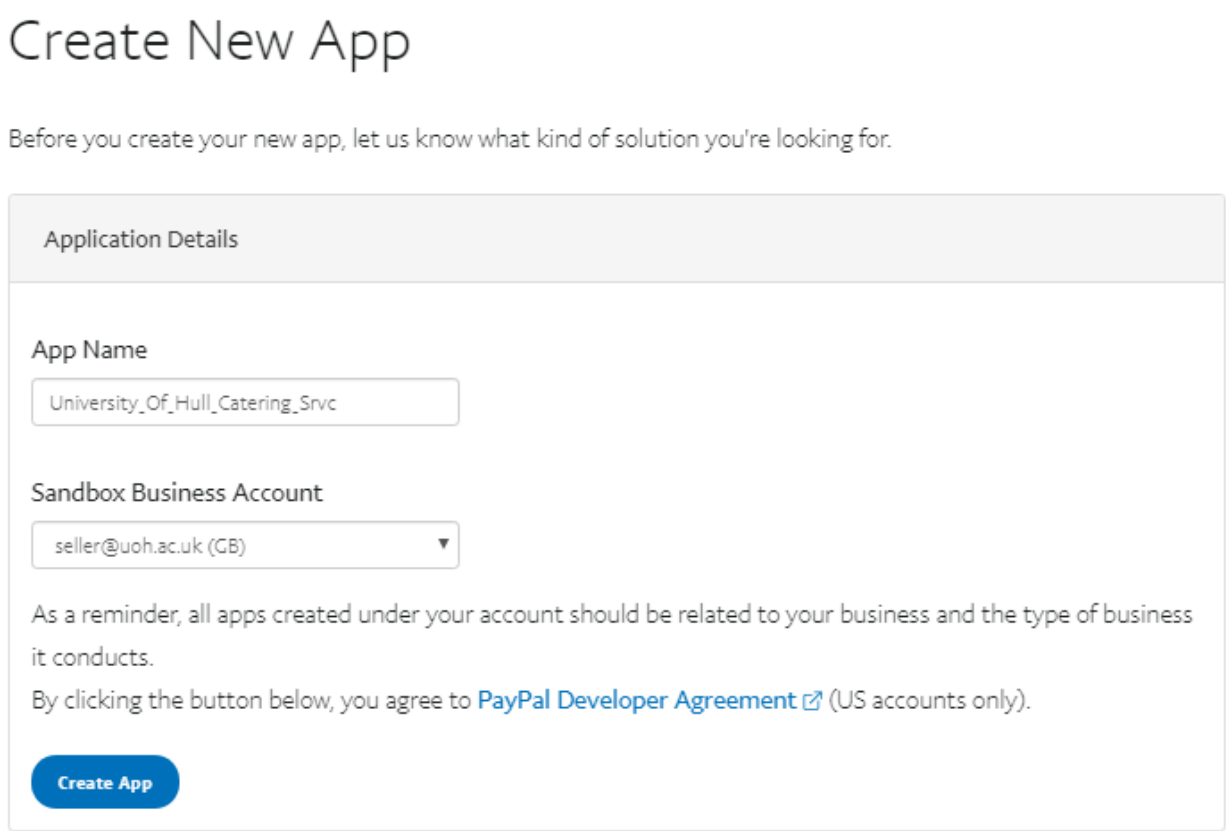
Before doing so, go to ‘Accounts’ under ‘Sandbox’. A personal and business accounts are made by default. The personal account has been renamed to 'buyer@uoh.ac.uk'; the facilitator account has been renamed to 'seller@uoh.ac.uk.'



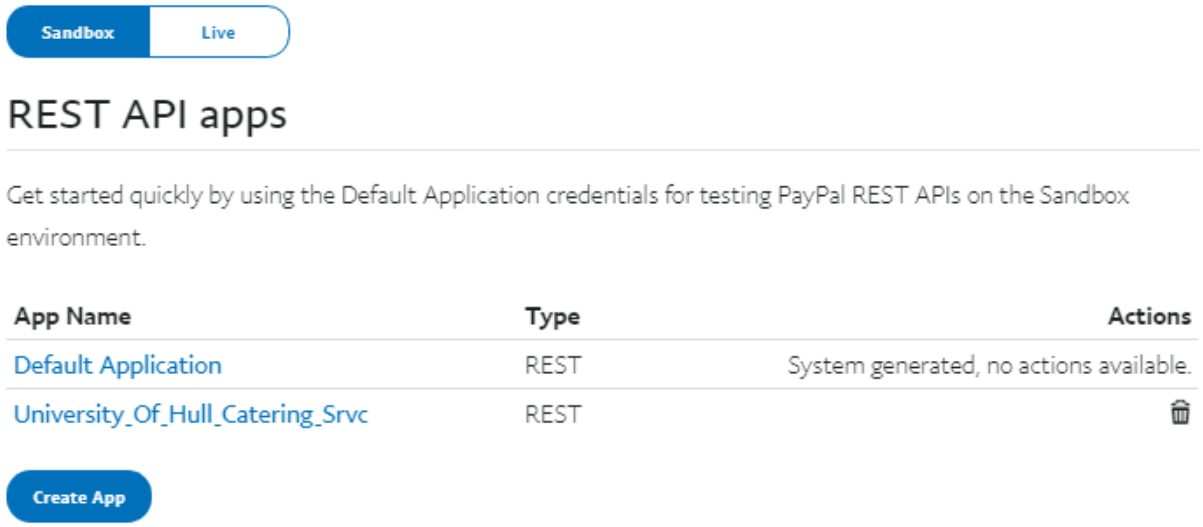
Step 4: Go back to the ‘My Apps & Credentials’ page under ‘Dashboard’ and click ‘Create App’ You will be redirected to the Create App page



Step 5: You will be redirect to the ‘Create New App’ page where you will be filling out the application details. For this project, the App Name is ‘University_Of_Hull_Catering_Srvc’ and the Sandbox Business Accounts is set as seller@uoh.ac.uk



If the creation of the app is successful, you will be redirected to the My Apps & Credentials page where the new application will now show up in the apps table and when clicked, its details are displayed.



In another page, we can see the API calls made when a purchase is done:

Sandbox API Call History

Email | seller@uoh.ac.uk

Total records: 38

Show 10 Per Page « 1 »

HTTP Status	URL	PayPal Debug ID	API Call Date
✓	/v1/payments/payment	c9bebe1bc217a	17 May 2020 07:08:05
✓	/v1/payments/payment	14afc48719a82	16 May 2020 14:01:11
✓	/v1/payments/payment	5ba4534223fcd	16 May 2020 14:00:35
✓	/v1/payments/payment	6d10e0bcfe396	16 May 2020 13:43:42
✓	/v1/payments/payment	5cfbbf08b1529	16 May 2020 13:19:06
✓	/v1/payments/payment	26ce09b0c538c	16 May 2020 12:25:52
✓	/v1/payments/payment	8ef385af0628d	16 May 2020 12:22:44

After a test payment is done, the PayPal emails with the order confirmation will show up in the 'Notification' page.

Sandbox emails and SMS notifications

Review sandbox emails and notifications as a result of REST and Classic API requests and responses.

Questions? Check out the Testing Guide. Non-U.S. developers should read our FAQ.

Search for notifications

Show 10 Per Page « 1 »

To	Subject	Date
seller@uoh.ac.uk	Notification of Payment Received	17 May 2020 15:10:54
buyer@uoh.ac.uk	Receipt for Your Payment to John Doe's Test Store	17 May 2020 15:10:44
seller@uoh.ac.uk	Notification of Payment Received	16 May 2020 22:03:14
buyer@uoh.ac.uk	Receipt for Your Payment to John Doe's Test Store	16 May 2020 22:03:12
buyer@uoh.ac.uk	Receipt for Your Payment to John Doe's Test Store	16 May 2020 21:45:22
seller@uoh.ac.uk	Notification of Payment Received	16 May 2020 21:45:22
buyer@uoh.ac.uk	Receipt for Your Payment to John Doe's Test Store	16 May 2020 20:18:12
seller@uoh.ac.uk	Notification of Payment Received	16 May 2020 20:18:02
buyer@uoh.ac.uk	Receipt for Your Payment to John Doe's Test Store	16 May 2020 18:57:35
seller@uoh.ac.uk	Notification of Payment Received	16 May 2020 18:57:34

PayPal API

After completing the development environment creation, the credentials needed to set up payments on your platform are generated and ready to be used as intended. When clicking on the app in 'My Apps & Credentials', the OAuth 2.0 credentials for the app will be displayed. The sandbox account is the account to which all the payment is directed. The Client ID and Secret behave as a username and password for the app. The Client ID is used to publicly identify your app. The Client Secret is only known to the application and the authorization server. Its purpose is to protect the communication by granting access tokens to the authorized users only.

To accept payments, the shop must integrate the PayPal mechanisms. Implementing the PayPal button is what made this possible. Once clicked, the PayPal API is called to set up the payment and URL is redirected to the PayPal pop-up where they can complete their purchase.
The PayPal open-source JS code that does the payments is presented appendix E. The information exchanged are the following:

- transaction: amount including total, currency, details of the transcation such as subtotal, tax, shipping, handling_fee, shipping_discount and insurance.
- description
- custom
- payment_options: allowed_payment_method
- soft_descriptor
- items_list: items with details including name, description, quantity, price, tax, sku, currency.
- note_to_payer

When the shop sends a request to PayPal to start a transaction, PayPal requests these information from the shop. After PayPal receives these details, they are shown to the buyer who can then continue or cancel their purchase (see appendix A for pseudocode).

With the PayPal integration, the trustworthiness of the system build increases. However, the PayPal sandbox system isn't flawless and the catering shop system itself still contains a collection of weaknesses.

Weakness

A quick heuristic evaluation was conducted to get some user feedback on the catering system. The results can be found in appendix C. Most users has no previous developing experience mentioned that the fact that the shop offered PayPal payment made them feel that it is trustworthy. However, this system lacks proper security.

As this is a dummy system, everything can be considered as a simulation. Instead of having session for the purchases, the order was stored in the local storage and could be deleted manually. The login has no encryption. The user ID is simply stored in the localstorage as long as the user is logged in. Although this simplified the testing and hosting this system, in a real system, having the details stored in the local storage is a critical action. Not only is the user at risk, but also, if the localstorage is cleared, all the progress the user had made is deleted and they would need to start over.

A series of tests were conducted evaluating the security and safety of this system. The results can be found in appending B.

A way to improve the system later on is to host sessions. This can be achieved by creating persistent cookies with a suitable lifespan. This would provide a safer environment for the user. The users can also create and store accounts' details. This can be attained by using SHA encryption. According to Cobb (2006), 'The SHA (Secure Hash Algorithm) family is a set of related cryptographic hash functions designed by the algorithm creates a hash value from any kind of data, such as a file, password, or in this case, a credit card number. This value is virtually unique to the input data, so even a small change in the data will result in a completely different hash due to the avalanche effect.'

Short Note

Authenticated Encryption Modes

Authenticated Encryption mode is a type of encryption that assures the privacy and integrity of a message or data. Authenticated Encryption has two functions: Encrypt and Decrypt. When it comes to encryption, it takes in plaintext and a key and output a ciphertext with an authentication tag. For decryption, it takes in a ciphertext, a key and an authentication tag ad outputs plaintext or an error if the authentication tag and the ciphertext don't match. It entails protecting the communication against chosen ciphertext attack; where the attacker tries to access information by sending ciphertext to the decryption oracle and analysing the decrypted results. This is what authenticated encryption prevents as it can recognize these ciphertexts and refuse to decrypt them therefore preventing the attacker from compromising the system (Wikipedia, n.d.).

Generic Composition Methods of Authenticated Encryption Modes

There are three type of generic composition methods of authenticate encryption: Encrypt-then-MAC (EtM), Encrypt-and-MAC (EaM) and MAC-then-Encrypt (MtE)

A MAC is a message authentication code and it is used to ensure integrity. EtM defines that the ciphertext is generated then a message authentication code is produced from it, meaning that the plaintext can't be verified or revealed by the MAC. As for EaM, the MAC can reveal information about the plaintext as it is generated from it. MtE the MAC and the plaintext as encrypted together, therefore the MAC won't reveal any information about the plaintext (Wikipedia, n.d.).

Single-Pass Authenticated Encryption Modes

A single-pass AE mode is a mode that provided authentication and privacy in a single pass. Two of its main types were IAPM and XCBC.

Integrity Aware Parallelizable Mode was one of the first single-pass AE modes. Although it is more difficult to implement, it reduces the number of block-cipher invocations compared to the generic compositions of AE. (Black, 2004)

eXtended CipherText Block Chaining, just like the REST APIs implemented are stateless and was a lot faster than the generic composition methods of AE modes (Gligor and Donescu, 2002).

References

Black, J. 2004. *Authenticated Encryption* [online] University of Colorado Boulder. Available at: https://www.colorado.edu/faculty/black/sites/default/files/attached-files/ae.pdf [Accessed 17 May 2020]

Castellani, S. and Dorairajan, A., 2019. *Types of Apis | Learn About the Different Types of APIs in 2020*. [online] API Friends. Available at: https://apifriends.com/api-creation/different-types-apis/ [Accessed 13 May 2020].

Cobb, M., 2006. *Use SHA to Encrypt Sensitive Data.* [online] Available at https://searchsecurity.techtarget.com/answer/Use-SHA-to-encrypt-sensitive-data [Accessed 17 May 2020]

Gligor and Donescu, 2002. *Fast Encryption and Authentication: XCBC Encryption and XECB Authentication Modes* [online]. Available at: https://link.springer.com/content/pdf/10.1007%2F3-540-45473-X_8.pdf [Accessed 17 May 2020].

PayPal, n.d. *Get started with PayPal Commerce Platform.* PayPal Developer. [online] Available at: https://developer.paypal.com/docs/api/overview/# [Accessed 29 April 2020]

PayPal, n.d. *PayPal Checkout Integration Guide.* PayPal Developer. [online] Available at: https://developer.paypal.com/docs/archive/checkout/integrate/# [Accessed 25 April 2020]

Pearlman, S., 2016. *What Are APIs And How Do APIs Work?* [online] MuleSoft Blog. Available at: https://blogs.mulesoft.com/biz/tech-ramblings-biz/what-are-apis-how-do-apis-work/ [Accessed 13 May 2020].

RapidAPI Staff., 2019. *Types of APIs*. [online] RapidAPI. Available at: https://rapidapi.com/blog/types-of-apis/ [Accessed 13 May 2020]

Richardson, L. and Ruby, S., 2007. *RESTful Web Services.* Beijing: O'Reilly, p. 86.

Uncubed. 2017. *What Is An API? Learn The Basics In 3 Minutes*. [online] Available at: https://www.youtube.com/watch?v=Rha1_St_9kw [Accessed 13 May 2020].

Wikipedia, n.d. *Authenticated Encryption.* [online]. Available at: https://en.wikipedia.org/wiki/Authenticated_encryption#Security_guarantees [Accessed 17 May 2020]

Appendix

Appendix A: PayPal Checkout Pseudo Code (app.js file – line 434 to like 473)

```
Initialize button render
     Get env
     Get client
          Get sandbox
     Get locale
     Get style
          Get size
          Get color
          Get shape
Initialize payment
     Get total cost
     Get order reference
     Initialize payment create
          Get transaction
          Get amount
          Get description
          Get note to payer
     Initialize onAuthorize
          Return execute
          Initialize payedByPayPal
```

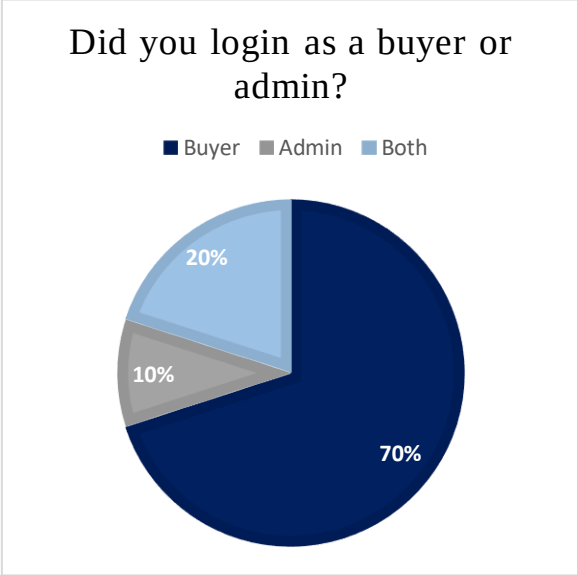
Appendix B: Tests

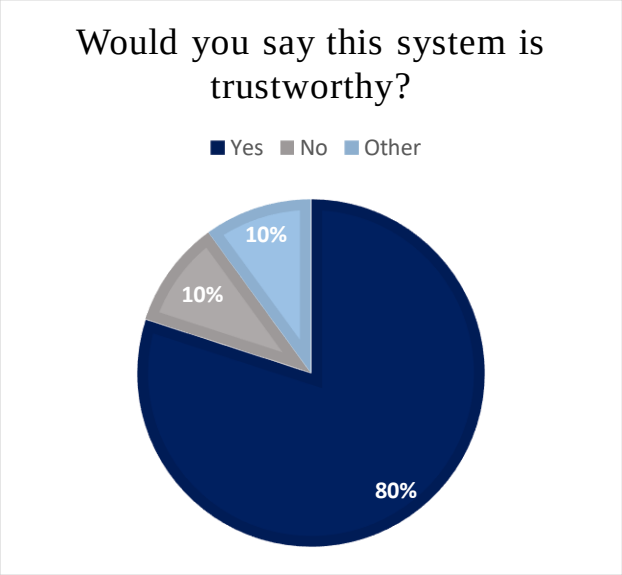
PAGE	***#***	***TEST CONDUCTED***	***EXPECTED RESULT***	***ACTUAL RESULT***
Home	1	Click login button	Get login pop up	As expected
	2	Click login button with no details	Error shows up	As expected
	3	Click login button with no user ID	Error show up	As expected
	4	Click login button with no password	Error show up	As expected
	5	Click log in with wrong password	Error show up	As expected
	6	Click log in with non-existent account	Error show up	As expected
	7	Login as admin	Redirected to admin	As expected
	8	Login as buyer1	Redirected to index	As expected
	9	Login as buyer2	Redirected to index	As expected
	10	Logout button	Redirect to home	As expected
	11	Go to Admin button	Redirect to admin	As expected
	12	Continue Shopping button	Redirect to index	As expected
Admin	13	Switch between tabs	Show different admin options	As expected
	14	Products – Click on 'Change Details'	Go to coming soon	As expected
	15	Stock – Open accordion tab	Show relevant table	As expected
	16	Stock – Open multiple accordion tabs	Show multiple tables	As expected
	17	Stock – Close accordion tab	Close relevant table	As expected
	18	Order – Switch through tabs	Show relevant table	As expected
	19	Go back to login page	Tell the user they're already logged in	As expected
	20	Logout button on nav bar	Go to home	As expected
Index	21	Switch between tabs	Show relevant pages	As expected
	22	Click on Home	Go to home	As expected
	23	Click on About	Go to about	As expected
	24	Click on Contact	Go to contact	As expected
	25	Click on cart	Go to cart	As expected
	26	Hover on item	Add to Cart popup	As expected
	27	Add item to cart	Cart icon + 1	As expected
	28	Add same item to cart	Increment in local storage	As expected
	29	Add multiple items to cart	Add in local storage	As expected
ShoppingCart	30	Go back to login page	Tell the user they're already logged in	As expected
	31	Logout button on nav bar	Go to home	As expected

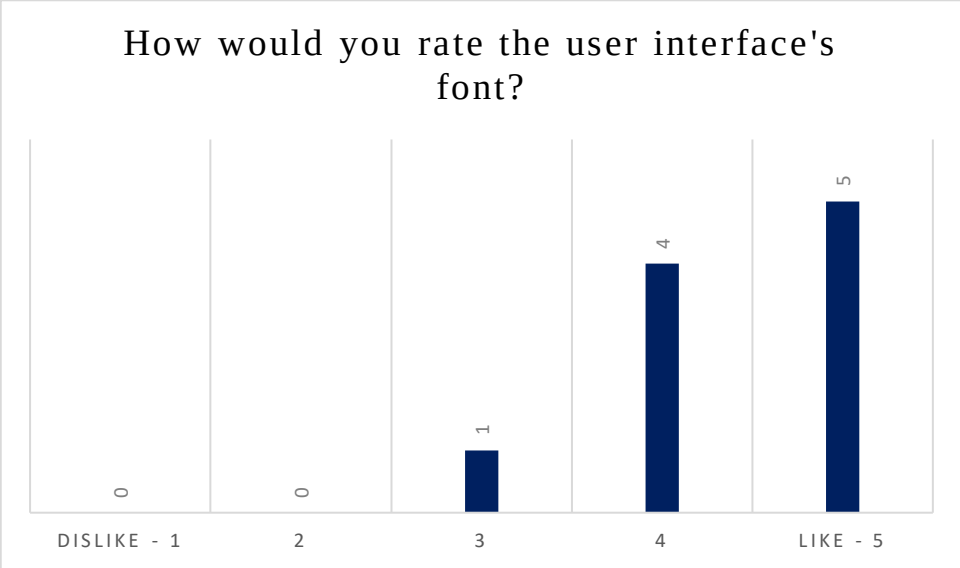
PAGE	*#*	*TEST CONDUCTED*	*EXPECTED RESULT*	*ACTUAL RESULT*
ShoppingCart	32	Tests 22 - 25	Go to relevant page	As expected
	33	Click on Continue Shopping	Go to index	As expected
	34	Click on Proceed to Payment	Go to payment	As expected
	35	Click on logout	Go to home	As expected
Payment	38	Tests 22 -25	Go to relevant page	As expected
	39	Click on Pay by Cash	Go to cashCheckout	As expected
	40	Click on Pay by Paypal	Get Paypal pop up	As expected
	41	Click on logout	Go to home	As expected
CashCheckout Successful	42	Make a new order	Go to index and clear cart	As expected
	43	Click on logout	Go to home	As expected
PayPal payment pop-up	44	Click on top-right drop down	See product details	As expected
	45	Click proceed to payment	Go to PaypalCheckout	As expected
PaypalCheckout Successful	46	Make a new order	Go to index and clear cart	As expected

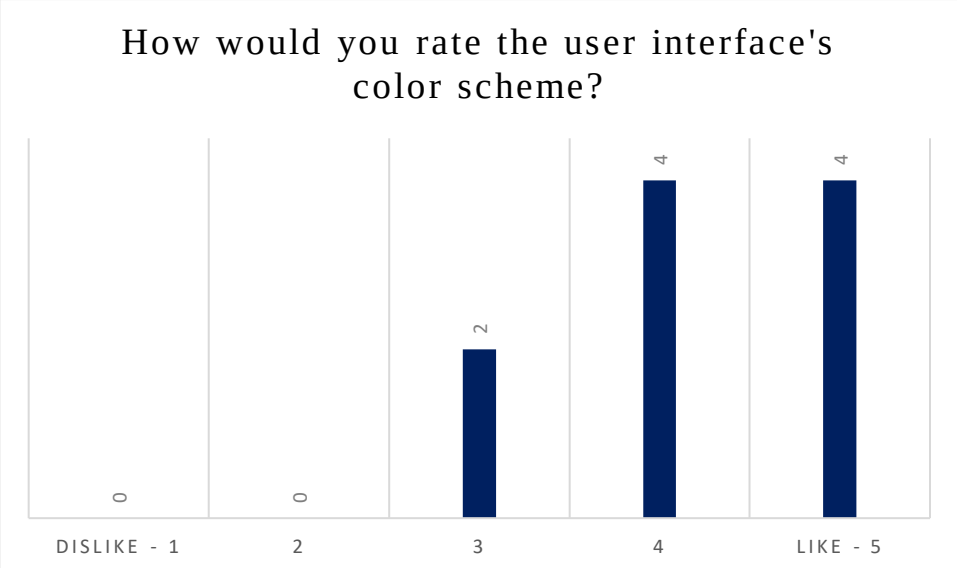
Appendix C: Heuristic Evaluation Results

Link to survey: https://docs.google.com/forms/d/1QNDf1uk3RrwmaduUDsYc9yOgY5UuiVfwsBvsCaKt6cs/edit



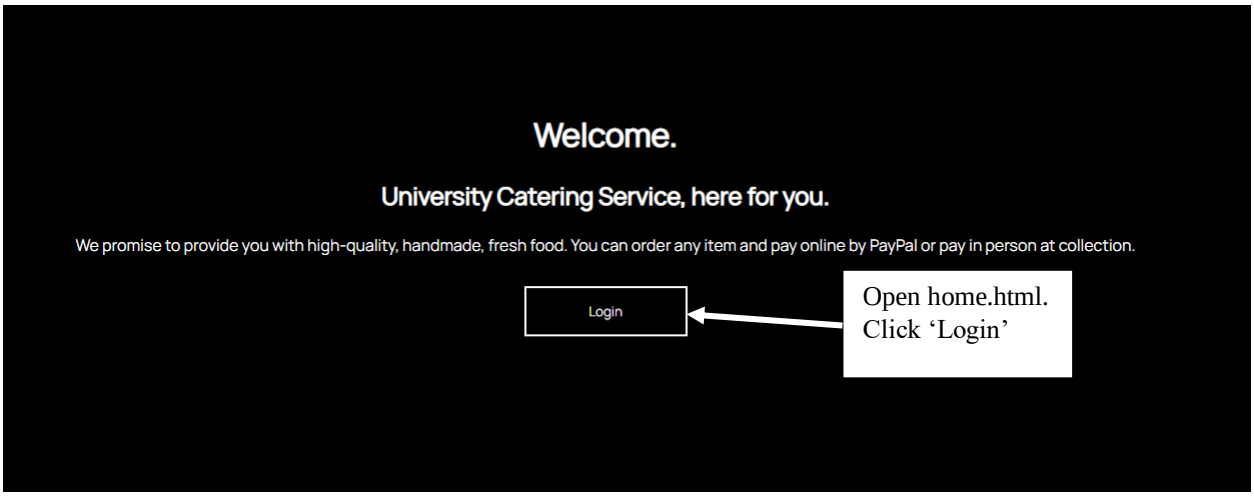


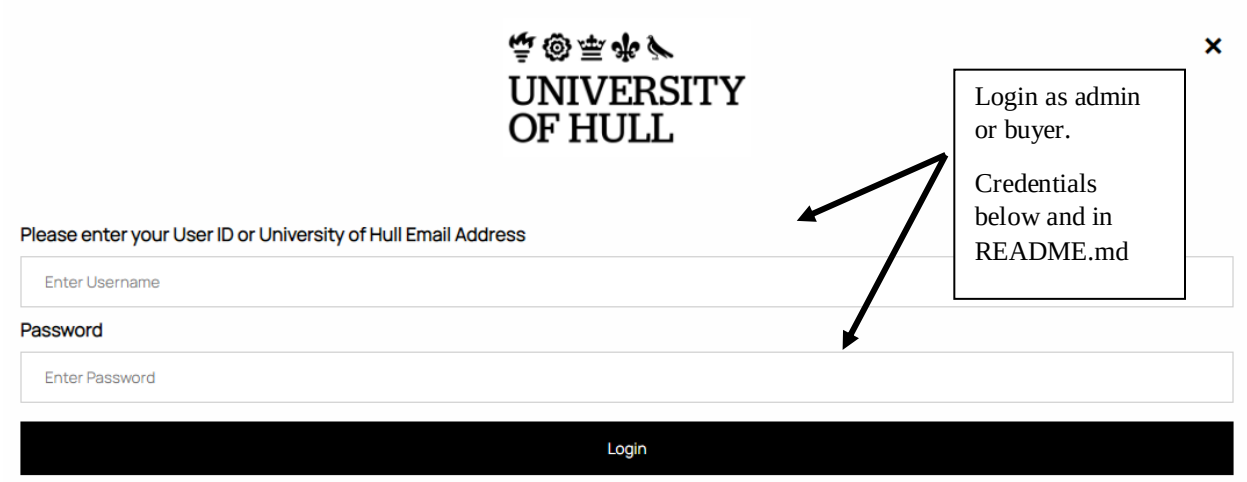




ID	Please provide a short comment about the interface.	Did you pay by cash or PayPal?	Any comments on the system's trustworthiness?
Participant 1	The interfaces choice of colour and font made the content easy to navigate and interact with as it was straightforward and not overwhelming	I didn't make a purchase	I found the system to be secure due to the option of purchasing with paypal
Participant 2	Easy and simple	PayPal	Because of the websites association with an actual institution, I thought the system was quite trustworthy.
Participant 3	they are easy to navigate through, but on mobile the fonts tend to look compressed.	PayPal	none that i can think of
Participant 4	The website looks sleek and very clean.	PayPal	n/a
Participant 5	The website uses simple colors and fonts but it's effective and nice to look at.	PayPal	PayPal already has an easy to use, safe interface so the fact that I could connect it to the site was great.
Participant 6	The user interface colors of choice and the font are very pleasing to the eye.	Cash	None
Participant 7	Simple and easy to use	PayPal	Could use a little more of a professional look
Participant 8	I like how simple and straightforward it was. There was also no confusion or mishaps when it was time to pay.	PayPal	No!
Participant 9	the design is simple and elegant	I didn't make a purchase	perfect
Participant 10	UI simple and nice	Both	No

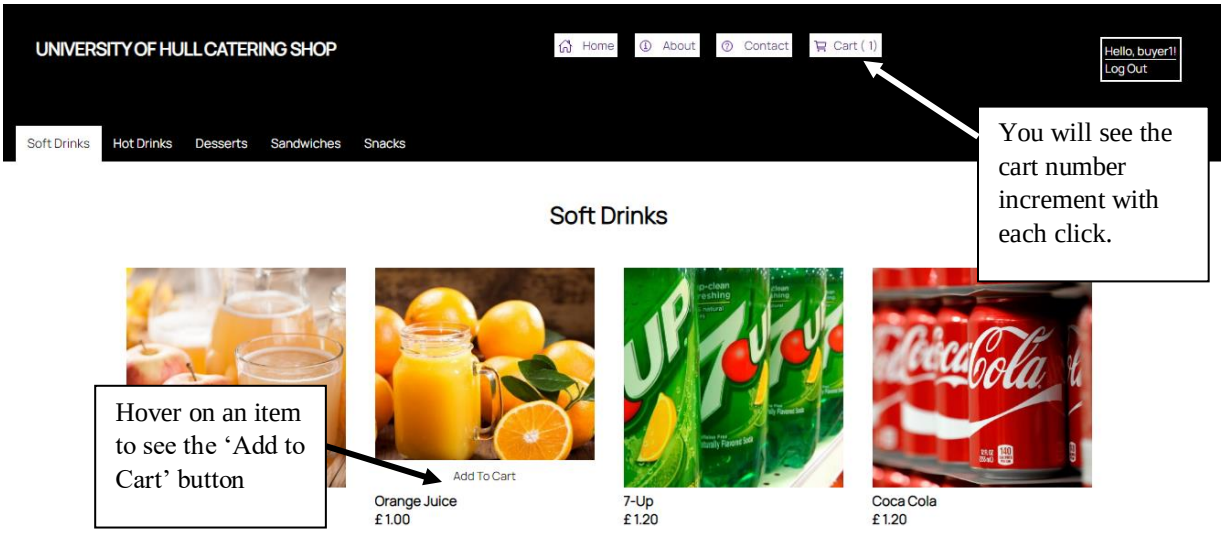
Appendix D: User Manual





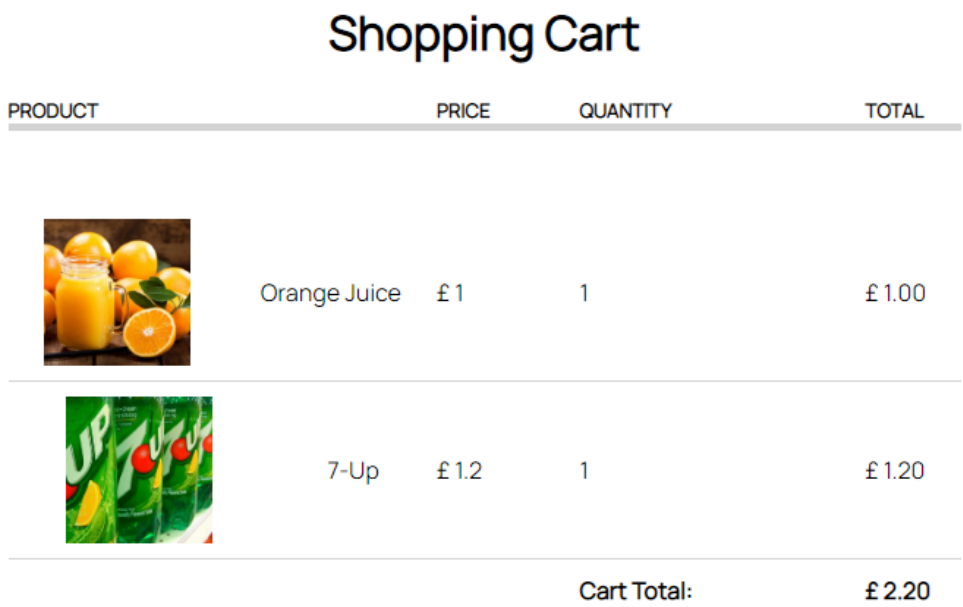
Credentials to use:

For admin:	For buyers:	
Username: admin	Username: buyer1	Username: buyer2
Password: admin	Password: buyer1	Password: buyer2



You will see the cart number increment with each click.

Hover on an item to see the 'Add to Cart' button

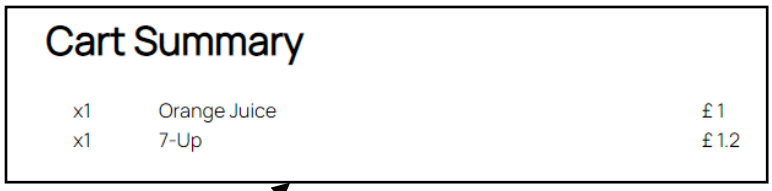


Click to go back to index page

Continue Shopping

Proceed to Payment

Click to proceed to payment page

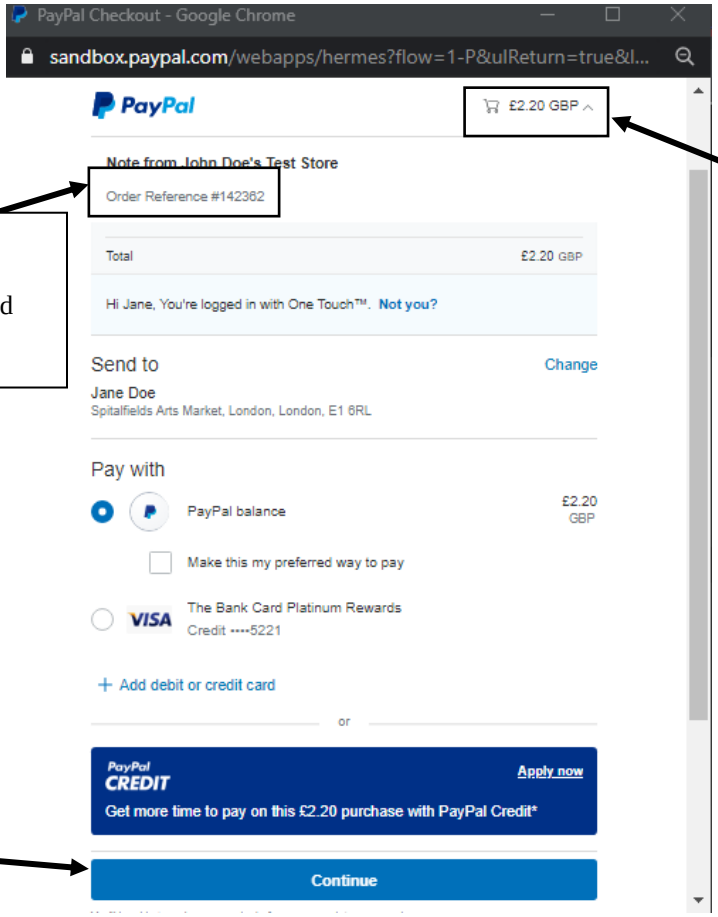


This is the summary of your purchase to confirm before payment

Total to Pay: £ 2.20

Click to pay by cash at collection

Click to pay by PayPal



Total cost of the order

A random order reference for the purchase is generated and displayed

Click to pay

UNIVERSITY OF HULL CATERING SHOP

Hello, buyer1!
Log Out

A payment confirmation page is then generated for the user.

Thank you for your purchase!

Order Reference #142362
Payment Method: Paypal
Transaction ID: 62A68556JW177224T

Order Summary

x1 Orange Juice £ 1
x1 7-Up £ 1.2
Total to pay: £ 2.20

Make A New Order

PayPal

18 May 2020 07:14:51 BST
Transaction ID: 62A68556JW177224T

This is the payment confirmation PayPal provides the customer with.

Dear Jane Doe,

You sent a payment of £2.20 GBP to John Doe's Test Store.

It may take a few moments for this transaction to appear in your account.

Merchant
John Doe's Test Store
sb-bmt47v1376350@business.example.com

Instructions to merchant
You haven't entered any instructions.

Delivery address – confirmed
Jane Doe
Spitalfields Arts Market
London,London
E1 6RL
United Kingdom

Dispatch details
The seller hasn't provided any dispatch details yet.

Description	Unit price	Qty	Amount
Order Reference #644541: x1 - Orange Juice - £ 1 x1 - 7-Up - £ 1.2	£2.20 GBP	1	£2.20 GBP

Subtotal	£2.20 GBP
Total	£2.20 GBP

Appendix E: PayPal Checkout Integration JS Code (PayPal, n.d.)

```
payment: function(data, actions) {
  return actions.payment.create({
    transactions: [{
      amount: {
        total: '30.11',
        currency: 'USD',
        details: {
          subtotal: '30.00',
          tax: '0.07',
          shipping: '0.03',
          handling_fee: '1.00',
          shipping_discount: '-1.00',
          insurance: '0.01'
        }
      },
      description: 'The payment transaction description.',
      custom: '90048630024435',
      //invoice_number: '12345', Insert a unique invoice number
      payment_options: {
        allowed_payment_method: 'INSTANT_FUNDING_SOURCE'
      },
      soft_descriptor: 'ECHI5786786',
      item_list: {
        items: [
        {
          name: 'hat',
          description: 'Brown hat.',
          quantity: '5',
          price: '3',
          tax: '0.01',
          sku: '1',
          currency: 'USD'
        },
        {
          name: 'handbag',
          description: 'Black handbag.',
          quantity: '1',
          price: '15',
          tax: '0.02',
          sku: 'product34',
          currency: 'USD'
        }],
        shipping_address: {···
        }
      }
    }],
    note_to_payer: 'Contact us for any questions on your order.'
  });
},
```