University of Bolton

Student Name	Thomas Plumpton
Student Number	1500936
Student Email	TP2AMT@Bolton.ac.uk

Programme	Computing
Module Code	CPU6013
Module Title	Web Application Project Development
Tutor Name	Brennan Tighe

Assignment Title	Supporting Documentation
Date of Submission	Tuesday 29th May 2018
Word Count	2092

Table of Contents

Table of Figures

Important Links

GitHub Repository
https://github.com/TomPlum/thewillwritingservice.git

Live Web Application URI
https://www.thewillwritingpartnership.tomplumpton.me/

Video Demonstration
https://youtu.be/1QkBTLtjepw

[1] Justification of Chosen Technologies
The Software Requirements Specification specified that the web application was to be developed in OctoberCMS using the LAPP Stack. This stack of technologies includes the Laravel framework, Apache web-server, PHP programming language and PgSQL Object-Relational Database Management System. The application was to be hosted on a dedicated web-server running Linux CentOS x64.

[1.1] MySQL Relational Database Management System
MySQL is an open source relational database management system (RDBMS) based on Structured Query Language (SQL). MySQL runs on virtually all platforms, including Linux, UNIX, and Windows. Although it can be used in a wide range of applications, MySQL is most often associated with web-based applications and online publishing and is an important component of an open source enterprise stack called LAMP. LAMP is a Web development platform that uses Linux as the operating system, Apache as the Web server, MySQL as the relational database management system and PHP as the object-oriented scripting language. (Sometimes Perl or Python is used instead of PHP.) (Rouse, n.d.)

MySQL uses the relational model just like PgSQL. Although MySQL is not object-orientated, they share a number of similarities that are relevant for the web application. For example, the LAMP stack is the same as the LAPP stack, except PgSQL has been replaced by MySQL. The web application that has been developed requires only simple data structures to be stored and does not require the user of complex queries to perform the necessary operations required for the functionality. Therefore, this choice of DBMS makes no difference to the development but works will with the rest of the revised technologies.

[1.2] NodeJS Runtime Environment
NodeJS is a runtime environment built upon Google Chrome's JavaScript V8 engine and allows for the execution of JavaScript code server-side. I choice this runtime environment and a long with the ExpressJS framework because I am familiar with it and there is a huge online community that provides documentation, open-source packages and libraries and active forums. Conversely, OctoberCMS is not very popular and therefore lacks an active online presence. This meant that during the early stages of development, research was very difficult to conduct as there was very little information available online. I had never used OctoberCMS before, or the Laravel framework. Several hours were invested into the research of OctoberCMS by myself and the rest of the class to no avail. Too much time had already elapsed and so the decision to switch the major technologies over was made in order to ensure that the clients functionality could still be delivered successfully and on time.

[1.3] ExpressJS Framework

The ExpressJS framework was created by the NodeJS foundation as is a lightweight, minimalistic framework made for developing web applications. I am also very familiar with this framework and it made the development a lot easier. Considering that the initial approved from end of the website was written in HTML5 mark-up and was styling with CSS3 with the bootstrap library, I could simply copy the HTML mark-up into a HTML to Jade converter and the PugJS mark-up could be copied into the relevant files. (StrongLoop, 2017)

[1.4] Stripe Payment System

The original specification specified that the payments were to be handled via a UTP (Universal Transaction Processing) Payment Gateway which uses PHP to handle the requests to the API. Considering that the revised stack of technologies are full-stack JavaScript, this of course was not an options as NodeJS does not contain a PHP interpreter.

Stripe is a technology company. Its software allows individuals and businesses to receive payments over the Internet. Stripe provides the technical, fraud prevention, and banking infrastructure required to operate on-line payment systems. (Wikipedia, 2018) Stripe has been audited by a PCI-certified auditor and is certified to PCI Service Provider Level 1. This is the most stringent level of certification available in the payments industry. To accomplish this, we make use of best-in-class security tools and practices to maintain a high level of security at Stripe. It also forces HTTPS for all services using TLS (SSL), including the public website and dashboard. (Stripe, 2017)

Stripe charges 2.9% + 30cents per transaction, which is the same as PayPal. It also free to charge cards from the website, unlike PayPal. (Strojny, 2016)

[1.5] Amazon Web Services Hosting

As previously mentioned, the Stripes payment gateway runs over HTTPS. Therefore, an SSL certificate is required for the website so that all data sent to and from the server is encrypted by HTTPS over SSL. I decided to host the web application on Amazon Web Services on an EC2 (Elastic Compute Cloud) container that is running Ubuntu x64. Ubuntu is a Linux distribution just like CentOS, so the majority of the skills and knowledge is transferrable. I already have multiple NodeJS web applications running on this server, so it was easy to setup another one and map it to a subdomain of one of my existing domain names. The SSL certificate is a free one provided by LetsEncrypt.org.

[1.6] Git Version Control System & GitHub

Git is a free and open source distributed version control system that works well for both small and large projects. It has a small footprint and very high performance. (git, 2018) GitHub is a web-based hosting service for version control using git. It is mostly used by programmers to store managed repositories of code that can be collaborated upon by multiple developers.

I decided to use Git for as the development VCS, a long with GitHub to store a repository of code. This creates a central place for all the code base, documentation, version history and acts as a backup of the web application should the local development storage becomes corrupt or be destroyed.

[2] Objectives Achieved

Below is a list of objectives achieved during development. They are either state in the Software Requirement Specification or have been implemented by virtue of another feature being added. For example, protecting the page routes from unauthorised users was not explicitly stated in the SRS. However, I have assumed that due to the nature of the project regarding data security that it would need to be implemented.

- User Registration
- User Login
- User Authentication with Protected Page Routing
- User Login Sessions & Cookies
- Last Will & Testament Multi-Page Form
- User Profile with History
- Stripe Payment Gateway Integration
- HTTPS & SSL Certificate
- Front-End Form Validation
- Server-Side SQL Injection Prevention

[3] Deviation from the Software Requirements Specification

[3.1] Technology Choice

As mentioned in Chapter 1, the technology stack was revised and changed to a more suitable one for the development.

[3.2] Database Structure Modifications & Additions

Chapter 3.4.1.1 of the Software Requirements Specification defined the MySQL data structure for the Last Will & Testament. The figure below shows the original hierarchal structure of tables that are relevant for the Will form.

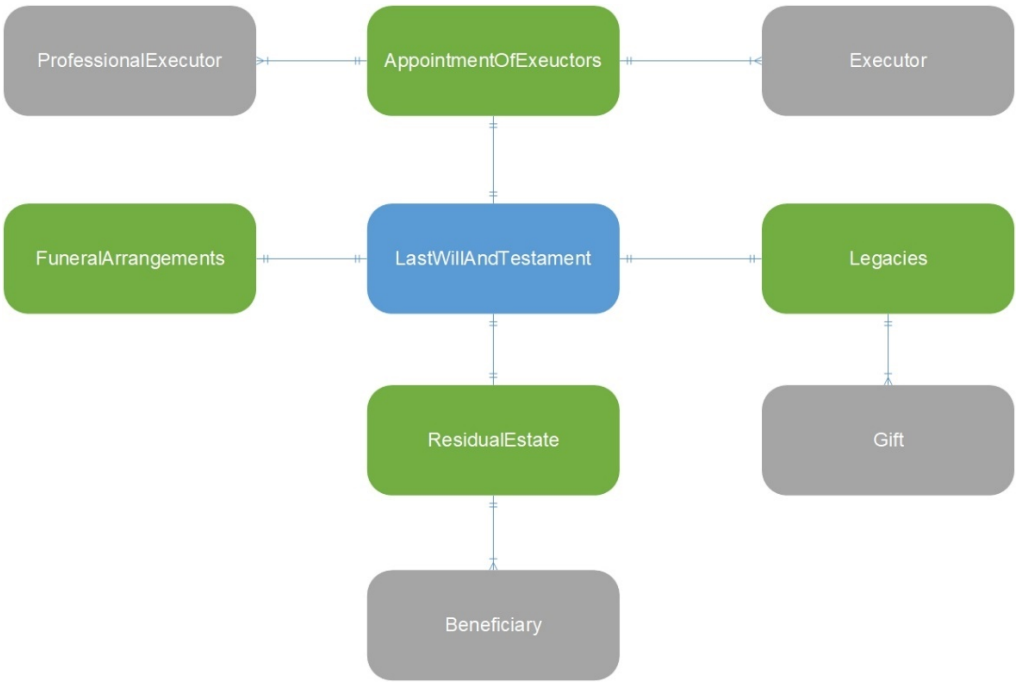


Figure 1 - Last Will & Testament Table Structure

During the development of the software changes were made to the overall structure of tables and to the structure of particular tables and columns. The Legacies and Gift tables were dropped as they were no longer needed due to the client not wanting them in the Will. Furthermore, extra fields were added to tables where necessary. For example the screenshot below shows the main 'LastWillAndTestament' table that links all the child tables together. I added the user_id, completed and progress fields as they were not included in the SRS.

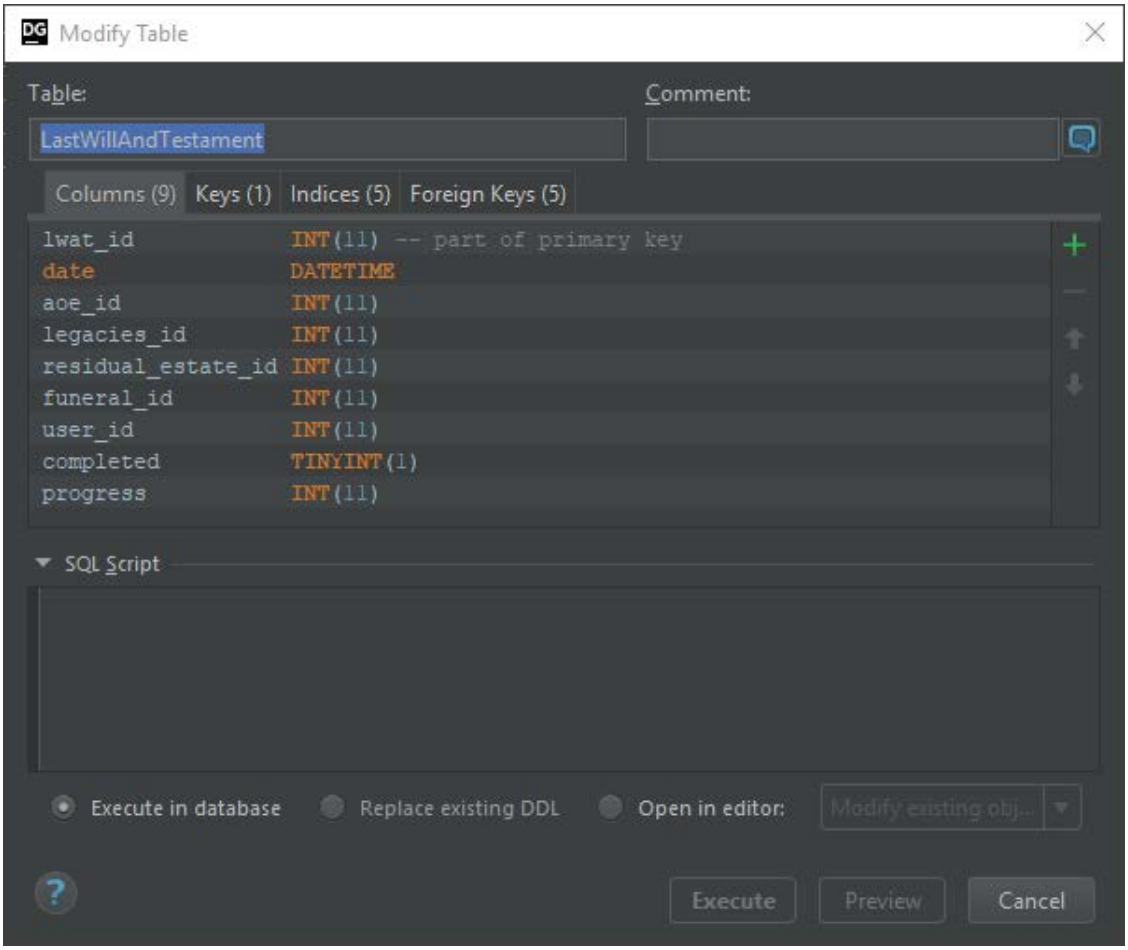


Figure 2 - LastWillAndTestament Table Structure

[4] Code Documentation & Inline Comments

[4.1] Inline Code Comments

Throughout the development of the project, I ensured inline comments where put into the code where needed. For example, some of more complex and ambiguous server-side functions needed comments for future developers. I used a JavaDoc style comment syntax above the function declaration. The screenshot below shows an example of this comments on one of the large asynchronous waterfall functions on the server-side. The rest of the function code is missing for clarity.

```
/* POST Database Last Will & Testament - Page 1 (Executors)
* This function uses an asynchronous waterfall from the npm library 'async' to do the following;
* 1: Insert an AppointmentOfExecutors row into the database.
* 2: Select the last inserted ID from the database (from the record above).
* 3: Uses npm 'node-async-loop' to iterate over the array of Executors and inserts a row for each one.
* 4: Uses node-async-loop again to iterate over the array of ProfessionalExecutors and inserts a row for each.
* 5: Updates the LastWillAndTestament record by setting it's AoE ID to that of the record we inserted above.
* 6: Updates the LastWillAndTestament record by setting its progress values to 2 (For the next stage, residual estate)
* 7: Finally, if all previous functions succeeded, it will send back an object to the AJAX call on the page.
* */
router.post('/save-last-will-and-testament-executors', isAuthenticated, (req, res) => {
    async.waterfall([
        callback => {
```

Figure 3 - JavaDoc Style Function Comment

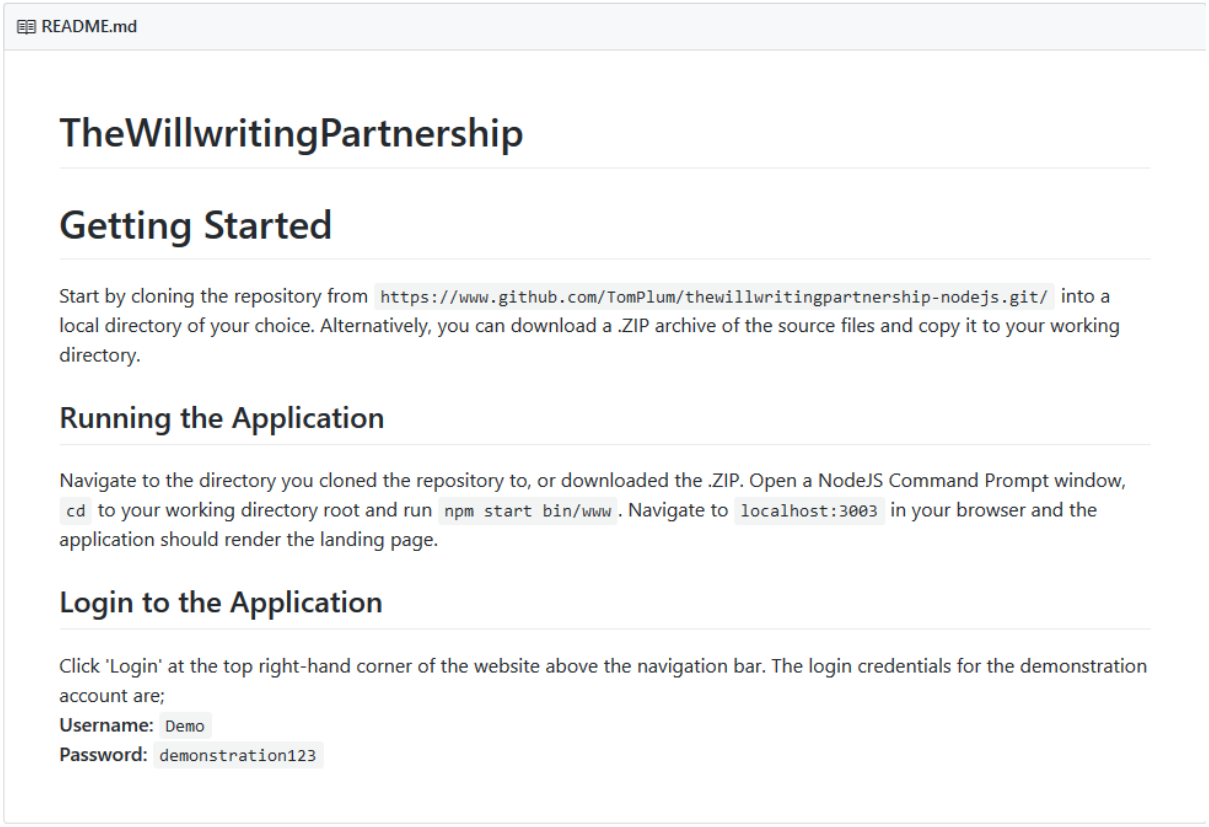
I also added shorter inline comments inside the function body where required. The screenshot below shows a for loop that iterates over all of the authenticated users' wills and builds a table of information and options. I have left inline comments in the code to briefly explain what easy part does for quick and easy reference for future developers.

```
for (let i = 0; i < data.length; i++) {
    //Open Row
    tBody += "<tr>";

    //Add Last Will & Testament ID
    tBody += "<td>" + data[i].lwat_id + "</td>";

    //Add Date
    tBody += "<td>" + formatDate(data[i].date) + "</td>";
    //If will is completed, add a check mark and "Complete"
    if (data[i].completed === 1) {
        tBody += "<td><i class='fas fa-fw fa-check-circle'></i> Completed</td>";
    } else {
        //Else, add in the current progress
        tBody += "<td>" + formatProgress(data[i].progress) + "</td>";
    }
    //Add in "Yes" or "No" for complete 1 and 0.
    tBody += "<td>" + (data[i].completed === 1 ? "Yes" : "No") + "</td>";
```

Figure 4 - Function Body Inline Comments

[4.2] GitHub Documentation

Further documentation regarding the web application was written in the project README.md file. This file uses 'markdown', a lightweight mark-up language with plan text formatting syntax. It contains information regarding the set-up process of the application and how to log-in.



README.md

TheWillwritingPartnership

Getting Started

Start by cloning the repository from `https://www.github.com/TomPlum/thewillwritingpartnership-nodejs.git/` into a local directory of your choice. Alternatively, you can download a .ZIP archive of the source files and copy it to your working directory.

Running the Application

Navigate to the directory you cloned the repository to, or downloaded the .ZIP. Open a NodeJS Command Prompt window, `cd` to your working directory root and run `npm start bin/www`. Navigate to `localhost:3003` in your browser and the application should render the landing page.

Login to the Application

Click 'Login' at the top right-hand corner of the website above the navigation bar. The login credentials for the demonstration account are;
Username: `Demo`
Password: `demonstration123`

Figure 5 - GitHub Repository readme.md

[5] Testing Phase

Manual testing was carried out on the web application

[5.1] Form Validation

As demonstrated in the video, the website has validation on all form fields. One feature I mentioned but didn't demonstrate was the jQuery form validation on the Will pages. The figure below shows a snippet of the Client Data Form. The rest of the form on the page has been filled in and validated. However, leaving the Address Line 2 blank and click 'Next' to submit causes the validation code to flag up the field as it is required. I have marked all required fields with a red asterisk.

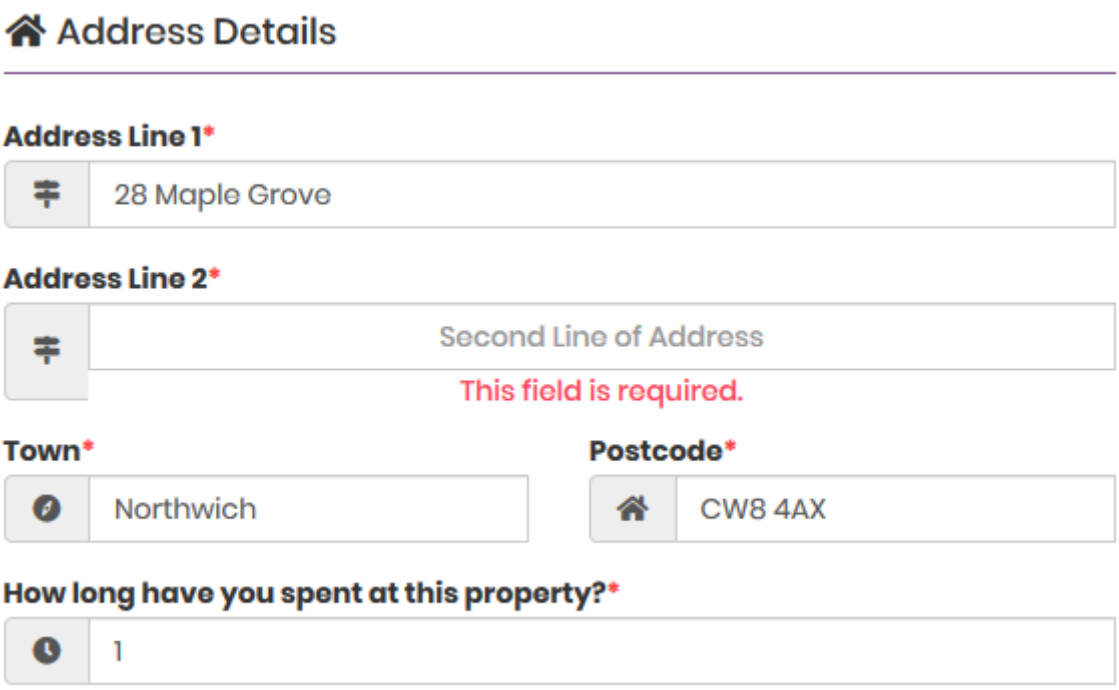
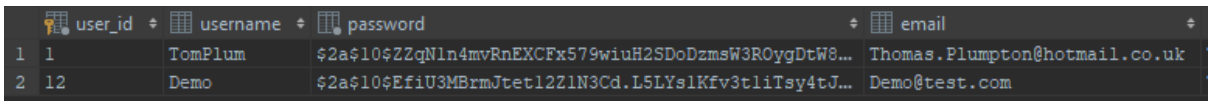


Figure 6 - Client Data Form - Address Details Form Validation

[5.2] User Data Security

In order to keep the users' data secure I took several measures. The first of which is the implementation of an SSL certificate on the sites domain to ensure that the HTTPS protocol is used at all times. This ensures that the users' sensitive form data and personal information is encrypted at all times when being transferred to and from the server. As mentioned in the video, unfortunately there is currently an issue at the time of writing this that affects the renewal of the existing certificate with the new subdomain. I have demonstrated that the certificate does work on another subdomain and will be fixed upon release of the next patch.

Furthermore, I ensured that all user passwords are hashed to further increase security. This means they can never be 'de-hashed' as there is no key, like encryption. I used an NPM (Node Package Manager) package called 'bcrypt' to achieve this.



	user_id	username	password	email
1	1	TomPlum	$2a$10$ZZqNln4mvRnEXCFx579wiuH2SDoDzmsW3ROygDtW8…	Thomas.Plumpton@hotmail.co.uk
2	12	Demo	$2a$10$EfiU3MBrmJtetl2ZlN3Cd.L5LYslKfv3tliTsy4tJ…	Demo@test.com

Figure 7 - Hashed Passwords

I also implemented preventative techniques to stop SQL injection. I did this by replacing the values in the queries with question marks and then passing an array of variables in the respective order to the MySQL query function.

```
async.waterfall([
    callback => {
        mysql.connection.query(
            "INSERT INTO ResidualEstate " +
            "(notes, pass_to_spouse, distribute_residue, excluded_from_will, add_failed_gift) " +
            "VALUES (?, ?, ?, ?, ?)",
            [
                req.body.notes,
                req.body.pass_to_spouse,
                req.body.distribute_residue,
                req.body.excluded_from_will,
                req.body.gift_fail
            ],
```

Figure 8 - MySQL Package Query Function

[6] Bibliography

Amazon, 2018. *Amazon Web Services.* [Online]
Available at: https://aws.amazon.com/
[Accessed 15 05 2018].

chenka, n.d. *jade converter.* [Online]
Available at: http://www.html2jade.org/
[Accessed 15 05 2018].

git, 2018. *git.* [Online]
Available at: https://git-scm.com/
[Accessed 15 05 2018].

Rouse, M., n.d. *MySQL.* [Online]
Available at: https://searchoracle.techtarget.com/definition/MySQL
[Accessed 16 05 2018].

Stripe, 2017. *Secuirty at Stripe.* [Online]
Available at: https://stripe.com/docs/security/stripe
[Accessed 15 05 2018].

Strojny, D., 2016. *Stripe vs PayPal: Who should you choose?.* [Online]
Available at: https://memberful.com/blog/stripe-vs-paypal/
[Accessed 15 05 2018].

StrongLoop, I., 2017. *Express.* [Online]
Available at: https://expressjs.com/
[Accessed 15 05 2018].

Wikipedia, 2018. *Stripe.* [Online]
Available at: https://en.wikipedia.org/wiki/Stripe_(company)
[Accessed 15 05 2018].