

1809709 - Web Services and Web Development

by Rhys Harding

FILE	12016_RHYS_HARDING_1809709_- _WEB_SERVICES_AND_WEB_DEVELOPMENT_181292_1604862527.DOC X (69.56K)		
TIME SUBMITTED	08-MAY-2019 11:55PM (UTC+0100)	WORD COUNT	8332
SUBMISSION ID	106707147	CHARACTER COUNT	55290

Introduction

REST stands for Representational State Transfer and is a common technology employed by developers building web applications that is considered as being easily scalable and maintainable [1]. RESTful web services are a client-server architecture meaning that workloads are divided into requests between a client and a server where the client requests access to a particular service or resource that the server holds [1]. The client can access this resource using HyperText Transfer Protocol (HTTP), although there are other transfer protocols which can be used [2]. HTTP enables the communication between these two where the client sends HTTP requests to the server with the intention of accessing or manipulating a resource that the server is exposing [2]. The server will then return a HTTP response back to the client based on the particular request that the client made [2]. A RESTful web service is the implementation of a REST Application Programming Interface (API) [3]. An API governs the communication between a client and server and acts as the mediator between the two allowing them to communicate [1], [3]. A REST API dictates a particular way in which HTTP is used by a service where the service exposes its resources using Uniform Resource Identifiers (URI's) [3]. These URI's expose a resource that is associated with that URI which can be represented in various forms such as JavaScript Object Notation (JSON) which is a way to format data that is to be stored or transported across a service [4]. The API supports the use of basic Create, Read, Update and Delete operations, also known as CRUD operations, upon these resources [3]. The RESTful API convention indicates particular ways in which these operations are supported and outline best practices when implementing a RESTful web service [5]. This report will aim to critically analyse the best practice in the implementation of RESTful web services. Supported with various pieces of literature advantages and disadvantages for its use will be considered and a conclusion will be made with the objective of summarising the effectiveness and usefulness of the best practices discussed.

RESTful API Best Practices

For a web service to be a RESTful web service it must adhere to core principles and aim to achieve best practices which model the idealistic approach to designing a web service . The core principles act as a footing to the best practices and should be found in all REST API's [6]. A REST web service follows a client-server design meaning that the server exposes the resources that the client wants to access whilst separating the user interface concerns with data storage concerns [6]. This architecture enables the loose coupling of components that can then be redesigned, reused and scaled effectively [6]. A REST web service is also defined to be stateless [6]. This principle describes how each response encapsulates all the necessary information to fulfil the client's request such that no data is stored between requests and does not need to persist past the point at which the request has been fulfilled [6]. This reduces the need to store any data besides cookies and reduces the complexities of managing memory [6]. Another defining principle is the act of caching effectively and efficiently to prevent old data and also lower the number of requests that have to be made to the server subsequently improving performance [6]. A uniform interface is another principle and provides the client with an easy-to-use API that describes itself easily such that it adheres to the following best practices [6].

URI handbook

The resource exposed by a web server for a client to access via request must be uniquely identifiable and the constraints that describe a good RESTful web service, regarding URI's mostly describe its format and its design [7].

```
{
  "_links" : {
    "self" : {
      "href" : "http://localhost:8080/profile"
    },
    "bids" : {
      "href" : "http://localhost:8080/profile/bids"
    },
    "batches" : {
      "href" : "http://localhost:8080/profile/batches"
    },
    "bidders" : {
      "href" : "http://localhost:8080/profile/bidders"
    },
    "regs" : {
      "href" : "http://localhost:8080/profile/regs"
    },
    "auctions" : {
      "href" : "http://localhost:8080/profile/auctions"
    }
  }
}
```

Firstly, with regards to the structure of a URI, the path construction of the path must adhere to a few basic rules including avoiding the use of underscores due to obscurity in decorated URI elements on HTML pages, consistently using lowercase letters to avoid confusion between URI's which are the same but which have uppercase letters within, and the prohibited inclusion of file extensions for which instead should be left to be indicated in the header as to how the data should be processed [6], [7]. In addition, there also remains other rules such as the use of hyphens to improve readability and avoid including the forward slash to help identify resources clearly [6], [7]. Although these set of rules may seem tedious, they help clearly define the structure of how a RESTful API should generate its URI's, helping to efficiently and cleanly communicating the exposed resources to the client [7].

GET <http://localhost:8080/valuate?plate=HO11IES>

```
{"plate":"HO11IES","cuss":"allie, dollie, hallie, holley, holli, hollie, hollis, holly, mollie, ollie, ","offensive":true}
```

Secondly, the design of a URI must follow a particular format of archetype structure. When designing a REST API, it's resources should inevitably fall into one of the following archetypes: Document, collection, store and controller [8]. Respectively, these refer to: A single instance of an object, a directory of resources managed by the server, a resource repository managed by the client, and any procedural concepts which are operations that are beyond the functionality of the basic CRUD operations [8]. For such archetypes, there exists certain practices that are intended to clarify the meaning of the URI's generated in order to make it clear when using or consuming the API what exactly is being done and this is done through

the specific use of nouns or verbs [8]. Nouns should be used for documents, whilst plural nouns should be used to name collections and stores [8]. This done to increase the meaningfulness of the URI with respect to the resource that it represents [8]. In the case of a controller, a verb should be used which describes the functionality of its action. These careful considerations to API help to clearly manifest order within how the resources can be navigated and how accurately they depict the resource that they represent [8]. Thus, creating clear instruction for what and how the API should be used. There is, however, a limit to which a good API bounds itself as to how meaningful its generated URIs are. Where a URI declares transparency of its back-end implementations it is not only meaningless to the client side, but it also hinders the ease of scalability or change since any future changes to the back-end may inaccurately be reflected in the transparency of the URI used [8]. A URI should hence be designed such that it is loosely coupled with the back-end so that it can accommodate to change and reusability [8].

Operations

The best practice for RESTful web services describe that the API should use the operations of the HTTP standard whilst complying to each of their semantics [8], [9], [10]. Thus, REST APIs must use a uniform interface in which to communicate between the client and server and should not abuse the use of the standard operations [8], [9], [10]. Hence, a GET request should purely be used for the use of retrieving information from the server. It should not, then, be used subsequently to place any other type of request through the misuse of design of a URI. Furthermore, POST and PUT requests should explicitly be used to create a new resource in a collection, and to update or create resources respectively [8], [9], [10]. Also, DELETE must explicitly delete a resource such that once the operation has been completed the resource can no longer be obtained through any request to the client [8], [9], [10]. Best practices describe that any derivatives of deletions of resources should be employed through a separate controller and invoked through a POST instead [8], [9], [10].

Response Codes

```
$.ajax({
  type: "POST",
  url: "/bids",
  contentType: "application/json",
  data: formData,
  dataType: "json",
  success: makeBidSuccess,
  error: ajaxError
});

function ajaxError(req, status, error)
{
  window.alert("Got error: " + status + "\nrequestStatus: " + req.status);
}
```

Associated with these operations are response status codes. These are codes that are included within a response that the server sends back to a client when a request has been made [11]. These codes indicate a variety of different messages and there exists particular etiquette of which errors are handled and how response codes are used within a best practice

REST API exemplar [11]. The 2xx code indicates a success and 200, 201 and 204 are derivatives of this response code [11]. Specifically, 200 should be used to indicate success of a nonspecific nature [11]. Response code 201 should be used to indicate the success of a created resource and 204 indicates a successful deletion following a DELETE request and can be used when there is no response body [11]. The 3xx response code is used to indicate redirection of a variety of different manners [11]. Specifically, the 302 code is not recommended. Its use is often misinterpreted and should only redirect a client if a GET or HEAD request has been made [11]. It is thus, recommended that a 303 or 307 code be issued instead in order to reference a URI in the Location header of the response for which the client should send a GET request to [11]. The 4xx code is intended to be used to indicate that there has been a client error, when no other derivative is considered a 400 indicates non-specific failure [11]. However, there are other codes that should be used to represent specific errors such as when the client fails authentication and has provided invalid credentials, when a client requests an unmappable URI, and when a client tries to request an unsupported operation on a particular resource where the particular resource does not allow it [11]. The response codes should be 401, 403 and 404 returned respectively. It is important to note however, that in the case where a client has requested authentication correctly however the access is beyond the scope of permission, a response code of 403 "Forbidden" should be sent, as oppose to a sometimes misconceived 401 [11]. On the contrary to a client-side error, a 500 error code should be used to indicate a generic server error. Best practices recommend the thorough use of response codes to name just a few and although a web service may be able to handle responding with response codes they can easily be misinterpreted or inadvertently misused [8], [9], [10], [11]. It is often the case where best practices forbid the use of such response codes or recommend the avoidance of typical response codes for nontypical scenarios [8], [9], [10], [11]. More specifically, developers of a RESTful web service should try to avoid using 500 response codes that are nonspecific and should only be used for API malfunction [8], [9], [10], [11].

Hypermedia as the Engine of Application State (HATEOAS)

```
"batches" : [ {
  "buy_now_price" : 0.0,
  "winning_bid" : 0,
  "batch_name" : "government agency",
  "_links" : {
    "self" : {
      "href" : "http://localhost:8080/batches/1"
    },
    "batch" : {
      "href" : "http://localhost:8080/batches/1"
    },
    "winning_bidder" : {
      "href" : "http://localhost:8080/batches/1/winning_bidder"
    },
    "regs" : {
      "href" : "http://localhost:8080/batches/1/regs"
    },
    "auction" : {
      "href" : "http://localhost:8080/batches/1/auction"
    },
    "bids" : {
      "href" : "http://localhost:8080/batches/1/bids"
    }
  }
}
```

HATEOAS constrains a RESTful web service to include hypermedia links in its response bodies in order to connect different resources where appropriate and thus providing an effective way in which a client can navigate the resources and thus the API [12].

Conclusion

This report has outlined some of the principles that found the design and implementation of a RESTful web service and has discussed the use of recommended best practices for its RESTful API. This report has analysed the effectiveness of such practices and to what extent they are applicable. There are a range of important standards that are perceived to best follow when constructing such a web service such as the meticulous design of the URIs that an API generates. Each outline how implementation of basic constraints help the consumption and client-side usage and navigation of the API become easier and simpler through concepts such as nouns for certain resource types, basic CRUD operations applicable to them and the use of HATEOAS. All of which can be considered testament to the effectiveness of such a protocol that is RESTful web services that are essentially easier for a user to decipher and access information that a service provides through its API.

Bibliography

- [1] Paik, Hye-young, A. L. Lemos, M. C. Barukh, B. Benatallah and A. Natarajan, "Web Services—REST or Restful Services," *Web Service Implementation and Composition Techniques*, pp. 67-91, 2017.
- [2] R. Ghosh, V. Soolapani and R. Ayothi, "Providing access via hypertext transfer protocol (HTTP) request methods to services implemented by stateless objects". United States of America Patent 9,860,298, 2 January 2018.
- [3] S. P. Ong, S. Cholia, A. Jain, M. Brafman, D. Gunter, G. Ceder and K. A. Persson, "The Materials Application Programming Interface (API): A simple, flexible and efficient API for materials data based on REpresentational State Transfer (REST) principles," *Computational Materials Science*, vol. 97, pp. 209-215, 2015.
- [4] T. W. Moore and P. R. Wardrop, "Method and system for encrypting JavaScript object notation (JSON) messages". United States of America Patent 9,461,817, 4 October 2016.
- [5] L. J. Mitchell, *PHP Web Services: APIs for the Modern Web*, O'Reilly Media, Inc, 2016.
- [6] D. Renzel, P. Schlebusch and R. Klamma, "Today's Top 'RESTful' Services and Why They Are Not RESTful," *Web Information Systems Engineering - WISE*, pp. 354-367, 2012.
- [7] M. Masse, *REST API Design Rulebook*, 2011.
- [8] V. Sahni, "Best Practices for Designing a Pragmatic RESTful API," [Online]. Available: <http://www.vi-naysahni.com/best-practices-for-a-pragmatic-restful-api#hateoas>. [Accessed 29 April 2019].

- [9] S. Jauker, "10 Best Practices for Better RESTful API | Thinking Mobile," 5 June 2014. [Online]. Available: <https://blog.mwaysolutions.com/2014/06/05/10-best-practices-for-better-restful-api/>. [Accessed 1 April 2019].
- [10] V. Sahni, "Best Practices for Designing a Pragmatic RESTful API,,," [Online]. Available: <http://www.vinaysahni.com/best-practices-for-a-pragmatic-restful-api#hateoas>. [Accessed: 22-Dec-2017]. [Accessed 1 April 2019].
- [11] C. e. a. Rodriguez, "REST APIs: A Large-Scale Analysis of Compliance with Principles and Best Practices," *Web Engineering*, vol. 9671, pp. 21-39, 2016.
- [12] A. Ciortea, O. Boissier, A. Zimmermann and A. M. Florea, "Give agents some REST: hypermedia driven agent environments," *International Workshop on Engineering Multi-Agent Systems*, pp. 121-141, 2017.

Introduction

Serverless technologies generally refer to the outsourcing of responsibility for the maintenance and management of backend server logic and state of a particular application [1]. These cloud-hosted applications make use of third-party services to purely manage the running and allocation of servers [1]. Before serverless providers such as giants Amazon Web Services (AWS) Lambda, Microsoft Azure and Google Cloud Functions offered up their services, web applications were often burdened with the maintenance of their own servers and its resources and were charged for doing so even during times when the demand on the particular servers was lower than the capacity of those servers [1]. This was the case especially for small companies and development teams and so for these predicaments it gave way for serverless technologies to take their place amongst the development of web services architectures and is used currently by many companies like Netflix across the industry [1]. This report will be carried out with the objective to outline the need for server-less technologies and will critically analyse the effectiveness of such technologies. In addition, this report will then follow by introducing and outlining the use of circuit breaker patterns and will again critically analyse their effectiveness.

Web Hosting

Web hosting is the act through which web services are housed by servers across the internet made available by serverless technologies [2]. Through this process developers are able to construct web applications that can be accessed via the internet whilst the hosting service maintains the server and resources that has been allocated for that application however much is needed at any given time [2]. Serverless technologies are also referred to as Function as a Service (FaaS) [3]. Web hosting is thus the accommodation of such FaaS which serve to run server logic in stateless containers where data does not persist past the point at which the event-triggering function has completed its task [3]. All that is required for that particular functionality is encapsulated into the connection made between client and server but once that connection has been fulfilled the server holds no acknowledgement to previous connections [1], [3]. Through the use of these technologies comes many advantages and reasons why development might want to incorporate these architectures in its design. On the other hand, there are instances where this might not be the case.

Discussion regarding advantages and disadvantages of serverless technologies

One of the first possible arguments in favour of serverless applications might be the cost reduction or cost effectiveness that it can grant. A serverless web service that only makes a few executions will only be charged for the time that was taken and for the memory that is required [1], [4], [5]. When an application uses its own server, however, the cost associated with the running of that server accounts for the constant up-time for that server regardless of whether the full capacity is being used if at all [1], [4], [5]. It is clear then to see why serverless is favourable with regards to being cost effective in coherence to the demand that is being made to the server. Furthermore, serverless technology is adaptable to higher

demands and an application can make use of the hosting service to reflect the demand that is being made in terms of more execution and more memory [1], [4], [5]. This makes the scalability of an application using a hosting service much easier since this can be done automatically [1], [4], [5]. Another cost benefit comes as a consequence of less of a need for architectural engineers since most of this is handled by the hosting service [1], [4], [5].

This leads onto another argument that concerns the responsibility of the back-end infrastructure. With serverless, the responsibility no longer lies with the development team rather it is the support granted that comes from the service providers [1], [4], [5]. However, the disregard of back-end maintenance leaves greater reliability upon the service provider and lesser control of the system, lack of transparency for server-side architecture and possible security risks [1], [4], [5]. Multi-tenancy is the concept of multiple applications running on the same servers. For an application that handles sensitive data, there is a possibility of exposure of this data due to the misconfiguration of servers on behalf of the provider [1], [4], [5]. Additionally, multi-tenancy can have performance implications [1], [4], [5].

Another advantage is the ability to deploy parts of an application as collections of executables quickly and wherever needed, as part of a complete transition to serverless or just parts at a time [1], [4], [5]. Furthermore, this ease of deployment is the same for redevelopment. Developers can quickly change, update or add features to the already existing architecture making it easier for companies, especially those who employ agile methodologies for software development [1], [4], [5]. However, from a developer's perspective this transition to serverless can be complex and can pose difficulties for testing code locally [1], [4], [5]. Furthermore, it is important to mention that an executable is limited to a 300 second timeout, meaning that a feature that has complex functions or functions that run over that limit are hindered and cannot be hosted [1], [4], [5]. For an application where the run time of particular functions can vary this can be problematic.

A further disadvantage comes from serverless code that remains unused for long periods of time [1], [4], [5]. Frequently executed functions are held in a state that the provider can quickly boot up, referred to as warm starts. Infrequently used functions, however, are booted up in 'cold' starts and endure a slightly slower response leading to possible performance degradation [1], [4], [5].

Introduction to Circuit Breaker Pattern (CBP)

The CBP is a pattern that is used to monitor the availability and responsiveness of specific functionalities of a web service, and acts as a mediator for the communications between client and server; in-case of the event that the microservice becomes unusable [6], [7], [8]. The nature of communications between client and server describes a series of sequential transactions where a client communicates to a server across a network in anticipation of a response [6], [7], [8]. The server then services the request and responds back with the necessary information. However, this is not always the case in events that render the microservice unusable. In such happenings the client, unaware of the server-side problem, awaits a response until the request exceeds a time-out limit at which point client deems the transaction as a failure [6], [7], [8]. However, the problem with this arrangement is that waiting for a time-out value to be exceeded is a relatively long time, especially considering

that this timeout may consequently be waited upon every time requests are made to the particular microservice and can be made by multiple clients [6], [7], [8]. The CBP attempts to bid a solution to this problem that dramatically reduces the time in which a client can deduce a service failure [6], [7], [8]. The process the CBP carries out to achieve this closely mimics the operation of an electrical circuit breaker, hence the name, and does so by functioning within the realms of three states; open, closed and half-open [6], [7], [8].

Each request and its expected response pass through the circuit breaker. In the scenario where the communications between the two is intact the circuit breaker is closed and allows the typical communications to occur [6], [7], [8]. In the case where the circuit breaker detects no response from the server, the breaker will become open meaning that no requests are able to succeed to the microservice and any that are made are responded to with the appropriate errors [6], [7], [8]. Once the breaker is able to detect a response from the microservice the breaker will revert to a closed state to continue communications as desired [6], [7], [8]. The third state, half-open, refers to the period between the closed and open states and serves as the period for which only a few requests are succeeded to the server in order to test the responsiveness of the microservice [6], [7], [8]. If those requests are serviced and sufficient number of responses are made, the breaker deems that the service has been restored and will revert to the closed state [6], [7], [8].

The circuit breaker is able to be constantly aware of the state of the microservice by continuous and constant monitoring of three different types [6], [7], [8]. The first is 'heartbeat' monitoring and is such that it constantly tests the availability of the service, checking whether the service can be reached [6], [7], [8]. The second monitoring uses synthetic transactions which mock the data typically received to test the microservices ability to function [6], [7], [8]. The final monitoring is called real-user monitoring and uses trend analysis of the duration of the particular operation to observe and determine the responsiveness of the microservice and whether it exceeds a point at which it is deemed unusable, and in this case will open the breaker [6], [7], [8].

The use of a CBP reduces the client wait time in the event of an unusable service and can reject a request that is deemed likely to fail dramatically quicker than it would take to wait upon a timeout [6], [7], [8]. Also, the pattern inevitably supports the system during recovery by managing load and reduces the impact on system performance [6], [7], [8]. The way in which a breaker pattern can monitor the health of a microservice can greatly improve the effectiveness of the pattern [6], [7], [8]. In particular, real-use monitoring provides accurate data in order to determine the responsiveness whereas heartbeat monitoring will only serve to determine the availability of the service [6], [7], [8]. The latter can be a much quicker identifier of its measure but lacks the ability to test true functionality. The former is a relatively more comprehensive test being one of responsiveness but in some cases might not be considered as a true representation of the health of the system since some operations may experience fluctuations in performance [6], [7], [8]. However, a pattern can possibly be used effectively to determine whenever it is likely to fail, and this sensitivity can be set by the developer [6], [7], [8]. The monitoring that refers to the mocking of request data also comes with its drawbacks in the sense that the microservice will have to accommodate and distinguish such mock data from live data. In addition, the mocking of data tests just a specific path and might not be representative of the entire microservice [6], [7], [8]. There are some

possible advantages that come from the handling of errors that come from the breaker with regards to how the application handles these errors and where applicable, can invoke alternate functionality in place of the downed service [6], [7], [8]. Nevertheless, the pattern has to be designed to be able to interpret the errors it is handling and effectively communicate the information in order to best serve the client and reap these benefits. In addition, the breaker can be used to also log failures which can aid the monitoring of health. Another point to be made is that during half-open states, the breaker may be programmed to only revert to a closed state when a number of successful requests have been responded to. This means that for a system that might receive low volumes of requests to a certain microservice, the pattern may not be able to accurately use real-user monitoring to determine the health of the service due to its lack of incoming data.

Conclusion

In conclusion, similar can be said for both technologies in the sense that their usage really depends on the scenario for which they are considered. Serverless technologies can bring exciting possibilities with the benefits such as cost reduction and less of an association with back-end responsibilities, if of course, is what one is looking for. However, this lack of control may be the reason an application would abstain from serverless as well as complex applications that are not suited to be transitioned to serverless. For the CBP the effectiveness depends on how it is designed and how well it used within an application. As it provides stability and reassurance it must also be well married with important concepts such as error handling and appropriate monitoring.

Bibliography

- [1] M. Fowler, "Serverless Architecture," [Online]. Available: <https://martinfowler.com/articles/serverless.html>. [Accessed 28 April 2019].
- [2] D. Zanon, Building Serverless Web Applications., Packt Publishing Ltd, 2017.
- [3] G. C. I. V. M. V. & S. A. Fox, Status of serverless computing and function-as-a-service (faas) in industry and research, 2017.
- [4] E. van Eyk, L. Toader, S. Talluri, L. Versluis, U. Alexandru and I. Alexandru, Serverless Is More: From PaaS to Present Cloud Computing, IEEE Computer Society, 2018.
- [5] M. Sewak and S. Singh, Winning in the era of Serverless Computing and Function as a Service, 2018.
- [6] V. Malcher, "Design Patterns in Cloud Computing," *IEEE*, 2015.
- [7] F. & W. J. Montesi, "Circuit breakers, discovery, and API gateways in microservices.," 2016.
- [8] M. Fowler, "Circuit Breaker," [Online]. Available: <https://www.martinfowler.com/bliki/CircuitBreaker.html>. [Accessed 29 April 2019].

Appendix

```
package com.example.regaution;

import java.io.File;
import java.io.FileNotFoundException;
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;

public class Generator {

    private String[] plates;
    private int size;
    private String matchedWord = "";

    public String getMatchedWord() {
        return matchedWord;
    }

    public void setMatchedWord(String matchedWord) {
        this.matchedWord = matchedWord;
    }

    // Constructors
    public Generator(int size) {
        plates = new String[size];
        this.size = size;
    }

    // Methods

    /**
     * A method that generates a custom 'batch' of plates discarding any that contain or
     * are likely to contain offensive words.
     * @param customization custom letters to be generated
     * @return a batch of plates
     * @throws FileNotFoundException
     */
    public String[] platesGen(String customization) throws FileNotFoundException {
        String temp;
        String[] cussArray = readCussFile();
        for (int i = 0; i < size; i++) {
            temp = plateGen(customization);
            if (!isBanned(temp, cussArray)) {
                plates[i] = temp;
            } else {
                i--;
            }
        }
        return plates;
    }

    public boolean testPlate(String plate) throws FileNotFoundException {
        String[] cussArray = readCussFile();
        return isBanned(plate, cussArray);
    }
}
```

```

    }

    public boolean valuatePlate(String plate) throws FileNotFoundException {
        String[] namesArray = readNameFile();
        return isBanned(plate, namesArray);
    }

    /**
     * A method that will cycle through the array and flags any as false if offensive.
     * @param plate
     * @param cussArray
     * @return true for offensive false otherwise
     */
    public boolean isBanned(String plate, String[] cussArray) {
        boolean found = false;
        if (plate.indexOf('1') >= 0) {
            String refactoredPlateWith1AsL = omitNumbersAndSpacesFor1AsL(plate);
            for (String s : cussArray) {
                System.out.println(s);
                if (isOffensive(refactoredPlateWith1AsL, s)) {
                    setMatchedWord(getMatchedWord()+s+", ");
                    found = true;
                }
            }
        }
        String refactoredPlateWith1Asl = omitNumbersAndSpacesFor1Asl(plate);
        for (String s : cussArray) {
            //System.out.println(s); used for debug
            if (isOffensive(refactoredPlateWith1Asl, s)) {
                setMatchedWord(getMatchedWord()+s+", ");
                found = true;
            }
        }
        return found;
    }

    /**
     * A method that will generate a reg plate in current style with the customization of the
     * last three characters
     *
     * @param customization custom letters
     * @return generated reg plate
     */
    public String plateGen(String customization) {
        StringBuilder builder = new StringBuilder();

        char[] locTags = locationTagGen();
        builder.append(locTags[0]);
        builder.append(locTags[1]);

        char[] ageld = ageldGen();
        builder.append(ageld[0]);
        builder.append(ageld[1]);

        builder.append(" ");

        builder.append(customization.toUpperCase());
        return builder.toString();
    }

    /**
     * A method that will generate the age identifiers of the current reg style
     *
     * @return two age identifiers
     */
    public char[] ageldGen() {

```

```

char[] ageld = {'0', '0'};
char digit;

digit = (char) (Math.random() * 2 + '0');
switch (digit) {
    case '0':
        break;
    case '1':
        ageld[0] = '1';
        break;
    case '2':
        ageld[0] = '6';
        break;
}
digit = (char) (Math.random() * 8 + '0');
ageld[1] = digit;
return ageld;
}

/**
 * A method that will generate the memory tags for the current reg style plates
 *
 * @return memory tags
 */
public char[] locationTagGen() {
    char[] locTags = {'A', 'A'};
    for (int i = 0; i < 2; i++) {
        char character;
        do {
            character = (char) (Math.random() * 25 + 'A');
        } while (character == 'Q' || character == 'I' || character == 'Z');
        locTags[i] = character;
    }
    return locTags;
}

/**
 * A method that reads offensive words from a file into an array.
 * @return array of offensive words
 * @throws FileNotFoundException
 */
public String[] readCussFile() throws FileNotFoundException {
    Scanner sc = new Scanner(new File("cussFile.txt"));
    List<String> cussFile = new ArrayList<>();
    while (sc.hasNextLine()) {
        cussFile.add(sc.nextLine());
    }
    return cussFile.toArray(new String[0]);
}

public String[] readNameFile() throws FileNotFoundException {
    Scanner sc = new Scanner(new File("names.txt"));
    List<String> nameFile = new ArrayList<>();
    while (sc.hasNextLine()) {
        nameFile.add(sc.nextLine());
    }
    return nameFile.toArray(new String[0]);
}

/**
 * A method that will flag a plate as offensive if it either contains, or is similar to an
 * offensive word in the case where the offensive word is greater than three characters.
 * @param refactoredPlate plate which has been refactored
 * @param cuss offensive word
 * @return true if offensive false otherwise
 */
public boolean isOffensive(String refactoredPlate, String cuss){

```

```

    if(refactoredPlate.contains(cuss)) // exact match
        return true;
    else // if similar to a degree
        if (cuss.length() < 4) { // if 3 letters or less, similarity is debatable
            return false;
        }
        else return isSimilar(refactoredPlate, cuss);
    }
}

/**
 * A method that will flag an offense if the plate is considered to be similar to
 * the offensive word unless in the cases where the plate shows no likeness, the matches are
 * not frequent enough to be considered similar or otherwise where the similarities are rated too
 * low to be deemed offensive.
 *
 * The deciding factor could be changed depending on how strict the comparison needs to be.
 *
 * @param plate plate to be checked
 * @param cuss offensive word
 * @return true if deemed similar to offensive word false otherwise
 */
public boolean isSimilar(String plate, String cuss) {
    String matches = searchForMatches(plate, cuss);
    if ( matches.isEmpty() || (matches.length() < cuss.length() - 1) || (matches.length() <= 3))
        return false;
    int rating = calculateRating(plate, cuss, matches);
    return rating >= cuss.length(); // The deciding factor
}

/**
 * A method that will search through the plate for any characters that match those of the
 * offensive word.
 *
 * @param plate plate to be analysed
 * @param cuss offensive word
 * @return any characters that match
 */
public String searchForMatches (String plate, String cuss) {
    char[] plateArray = plate.toCharArray();
    char[] cussArray = cuss.toCharArray();
    StringBuilder builder = new StringBuilder();
    int placeholder = 0;
    for (char c : cussArray) {
        for (int i = placeholder; i < plateArray.length; i++) {
            if (c == plateArray[i]) {
                builder.append(c);
                placeholder = i+1;
                break;
            }
        }
    }
    return builder.toString();
}

/**
 * A method that compares the plate and its matches with the offensive word, giving an indication
 * via a rating that is scored by the following:
 *
 * - number of characters that are shared by both plate and offensive word
 * - the order in which each matching character proceeds the one before and how closely that succession
 * matches that of the offensive word. Scoring highly for continuous succession.
 *
 * @param plate plate to be compared
 * @param cuss offensive word
 * @param matches common characters
 * @return rating indication
 */
public int calculateRating(String plate, String cuss, String matches) {
    int rating = 0;
    if (matches.length() == cuss.length())

```

```

        rating += 3;
    else if (matches.length() == cuss.length() - 1 )
        rating += 2;
    else if (matches.length() == cuss.length() - 2 )
        rating += 1;

    // evaluating the order of the matches
    char[] matchesArray = matches.toCharArray();
    char[] plateArray = plate.toCharArray();
    int accumulator = 0;
    int y = 0;
    for (int i=0; i<matchesArray.length - 1; i++){
        char letter = matchesArray[i];
        char letterInPlate = plateArray[plate.indexOf(letter, y)];
        char nextLetter = matchesArray[i+1];
        char nextLetterInPlate = plateArray[plate.indexOf(letter, y) + 1];
        if (letter == letterInPlate && nextLetter == nextLetterInPlate){
            if (accumulator == 1)
                rating += 2;
            else if (accumulator == 2)
                rating += 3;
            else if (accumulator > 2)
                rating += 3;
            else {
                rating += 1;
            }
            accumulator += 1;
        } else {
            accumulator = 0;
        }
        y = plate.indexOf(letter, y) + 1;
    }
    return rating;
}

/**
 * A method that refactors a common reg plate into one that replaces numbers that can be perceived as
 * alphabetical characters. Also changes plate to lowercase
 * @param plate
 * @return refactored plate
 */
public String omitNumbersAndSpacesFor1AsL(String plate) {
    return omitNumbersAndSpaces(plate, '1');
}

public String omitNumbersAndSpacesFor1AsL(String plate) {
    return omitNumbersAndSpaces(plate, '1');
}

private String omitNumbersAndSpaces(String plate, char c) {
    String replacement = plate.replace(" ", "");
    char[] numbers = {'0', '1', '3', '4', '5', '6', '7', '8', '9'};
    char[] replacements = {'o', 'c', 'e', 'a', 's', 'g', 't', 'b', 'g'};
    for (int i=0; i < replacements.length; i++)
        replacement = replacement.replace(numbers[i], replacements[i]);
    replacement = replacement.toLowerCase();
    return replacement.toLowerCase();
}
}
package com.example.regauction;

import com.example.regauction.model.auction;
import com.example.regauction.model.batch;
import com.example.regauction.repository.*;
import com.example.regauction.model.reg;
import org.springframework.beans.factory.annotation.Autowired;

```

```

import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

import javax.annotation.PostConstruct;
import java.io.FileNotFoundException;

@Service
public class DataFillerService {
    private final RegRepository regR;
    private final AuctionRepository auctionR;
    private final BatchRepository batchR;
    private final BidRepository bidR;
    private final BidderRepository bidderR;

    @Autowired
    public DataFillerService(BidderRepository bidderR, RegRepository regR, AuctionRepository auctionR,
BatchRepository batchR, BidRepository bidR) {
        this.regR = regR;
        this.auctionR = auctionR;
        this.batchR = batchR;
        this.bidR = bidR;
        this.bidderR = bidderR;
    }

    @PostConstruct
    @Transactional
    public void fillData() throws FileNotFoundException {
        // Create auctions
        auction firstAuction = new auction("2019-04-18T12:00:00", "2019-04-20T12:00:00");
        auctionR.save(firstAuction);
        auction secondAuction = new auction("2019-04-12T12:00:00", "2019-07-12T12:00:00");
        auctionR.save(secondAuction);
        auction thirdAuction = new auction("2019-07-15T12:00:00", "2019-08-12T12:00:00");
        auctionR.save(thirdAuction);

        //GUM
        // Create batch
        Generator gen = new Generator(10);
        String[][] regBatchArray = {
            {"government agency", "GOV"},
            {"welsh water business", "DWR"},
            {"plumbing business", "LOO"},
            {"telecoms company", "COM"},
            {"gas tanker company", "GAS"},
            {"delivery service company", "DPD"},
            {"soft-drinks company", "POP"},
            {"van hire company", "VAN"},
            {"pest control business", "RAT"}
        };

        for (int i=0; i<regBatchArray.length; i++){
            auction _auction;
            if (i<3)
                _auction = secondAuction;
            else if (i<6)
                _auction = firstAuction;
            else
                _auction = thirdAuction;
            batch batch = new batch(regBatchArray[i][0], _auction);
            batchR.save(batch);

            String[] plates = gen.platesGen(regBatchArray[i][1]);
            for (int j=0; j < plates.length; j++) {
                reg plate = new reg(plates[j], batch);
                regR.save(plate);
            }
        }
    }
}

```

```

    }
}
$(document).ready(function() {

    displayBatches();

    $('#postBid').prop('disabled', true);
    $('#amount').change(function() {
        if($(this).val() != "" && $('#selectGroup').val() != 'Choose Batch' && $('#status').val() === 'in progress') {
            $('#postBid').prop('disabled', false);
        } else {
            $('#postBid').prop('disabled', true);
        }
    });
    $('#newBidForm').submit(function (e) {
        var formData = JSON.stringify( {
            bid_amount : $('#amount').val(),
            batch : $('#selectGroup').val(),
            bidder : $('#selectUser').val()
        });
        POSTBid(formData);
        e.preventDefault();
    });
    $('#newPlateForm').submit(function (e) {
        var plate = $('#plate').val();
        checkPlate(plate);
        e.preventDefault();
    });
    $('#valueForm').submit(function (e) {
        var plate = $('#plateToBeValued').val();
        valuePlate(plate);
        e.preventDefault();
    });
    $('#loadingDiv')
        .hide() // Hide it initially
        .ajaxStart(function() {
            $(this).show();
        })
        .ajaxStop(function() {
            $(this).hide();
        });

    loadBids(0);

});

function refresh(){
    $('#bidList').load("landingPage.html #bidList", loadBids(500)).fadeIn("slow");
}

function loadBids(delay){
    $('#loading').css('display', 'block');
    setTimeout(function() {
        $('#loading').css('display', 'none');
        displayYourBids();
    }, delay);
}

//BATCHES
function displayBatches() {
    $.ajax({
        type: "GET",
        url: "/batches",
        contentType: "application/json",
        dataType: "json"
    });
}

```

```

    }).then(function (data) {
        // initialise card deck and batch menu selection
        $('#batchList').append("<div id = 'batchCardDeck' class='card-deck justify-content-center' >");
        $('#formBatchSelect').append("<label class='card-title text-uppercase' for='selectGroup'>Batch
Name</label><br>" +
            "<select class='form-control card-price text-muted text-uppercase' id='selectGroup'
name='selectGroup' onchange='updateForm($('#selectGroup').val())'>" +
            "<option>Choose Batch</option>");

        // cycle through each batch
        for (var dataitems = 0; dataitems < data._embedded.batches.length; dataitems++) {
            // add card to deck for each batch
            addBatchCard(data._embedded.batches[dataitems]._links.self.href, dataitems);
            // for each batch add the batch name to the selection menu
            $('#selectGroup').append("<option id=''" + data._embedded.batches[dataitems]._links.self.href + "'"
value=''" + data._embedded.batches[dataitems]._links.self.href + "'>" +
data._embedded.batches[dataitems].batch_name + "</option>");
        }
    });
}

function addBatchCard(batch, items) {
    $.ajax({
        type: "GET",
        url: batch,
        contentType: "application/json",
        dataType: "json"
    }).then(function (data) {
        $('#batchCardDeck').append(
            "<div class='col-lg-4'>" +
            "<div class='card' id='card' + items + '>" +
            "<div class='card-body'>" +
            "<h5 class='card-title text-muted text-uppercase text-center'>Private plates for your</h5>" +
            "<h6 class='card-price text-center text-uppercase text-center'>" + data.batch_name + "</h6>" +
            "<hr>" +
            "<ul class='fa-ul' id='myList' + items + '>" +
            "</ul>" +
            "<button id='goToBid' class='btn btn-block btn-primary text-uppercase' onClick='onClickBid('" + batch
+ "', )" + ">Make Bid</button>" +
            "</div>" +
            "</div>" +
            "</div>");
        appendRegList(data._links.regs.href, items);
    });
}

function appendRegList(reg, items) {
    $.ajax({
        type: "GET",
        url: reg,
        contentType: "application/json",
        dataType: "json"
    }).then(function (data) {
        for (var dataitems = 0; dataitems < data._embedded.regs.length; dataitems++) {
            $('#myList' + items).append(
                "<li><strong>" + data._embedded.regs[dataitems].reg_no + "</strong></li>");
        }
    })
}

// MAKE BID

function onClickBid(batch) {
    $('#selectGroup').val(batch)
    updateForm($('#selectGroup').val());
    $('html, body').animate({
        scrollTop: $('#makeBid').offset().top
    }, 1000);
}

```

```

    }, 2000, "easeInOutExpo");
}

function updateForm(batch){
    setAuctionTime(batch);
    // calculateHighestBid(batch);
    getHighestBid(batch);
    setAuctionStatus(batch);
    $('#amount').val('');
    $('#postBid').prop('disabled', true);
}

function getHighestBid(batch){
    $.ajax({
        type: "GET",
        url: batch,
        contentType: "application/json",
        dataType: "json"
    }).then(function (data) {
        $('#highestBid').val(data.winning_bid);
        $('#amount').attr({'min': data.winning_bid+1 });
    })
}

function calculateHighestBid(batch) {
    $.ajax({
        type: "GET",
        url: batch,
        contentType: "application/json",
        dataType: "json"
    }).then(function (batchData) {
        var batchUrl = batchData._links.self.href;
        $.ajax({
            type: "GET",
            url: batchData._links.bids.href,
            contentType: "application/json",
            dataType: "json"
        }).then(function (bidData) {
            var highestBid = 0;
            var highestBidder = null;
            //window.alert("about to compare");
            for (var dataitems = 0; dataitems < bidData._embedded.bids.length; dataitems++) {
                //window.alert("inside for loop");
                //window.alert("if " + bidData._embedded.bid[dataitems].bid_amount + " > " + highestBid);
                if (bidData._embedded.bids[dataitems].bid_amount > highestBid) {
                    highestBid = bidData._embedded.bids[dataitems].bid_amount;
                    //window.alert("highestBid set to: " + highestBid);
                    highestBidder = bidData._embedded.bids[dataitems]._links.bidder.href;
                    //window.alert("highestBidder set to: " + highestBidder);
                }
            }
            //window.alert("finished comparing");
            $.ajax({
                type: "GET",
                url: highestBidder,
                contentType: "application/json",
                dataType: "json"
            }).then(function (bidderData) {
                highestBidder = bidderData._links.self.href;
                var batchData = JSON.stringify({
                    winning_bid: highestBid,
                    winning_bidder: highestBidder
                });
                //window.alert("sending off patch for " + batchUrl);
                $.ajax({
                    type: "PATCH",
                    url: batchUrl,
                    contentType: "application/json",
                    dataType: "json",

```

```

        data: batchData
    });
});
});
});
}

```

```

function setAuctionStatus(batch) {
    $.ajax({
        type: "GET",
        url: batch + "/auction",
        contentType: "application/json",
        dataType: "json"
    }).then(function (data) {
        var startTime = Date.parse(data.start_time);
        var stopTime = Date.parse(data.stop_time);
        var currentDateTime = new Date();
        if (currentDateTime < startTime){
            $('#status').css({
                'background-color': 'rgba(114,114,114,0.99)',
                'border-color': 'rgba(114,114,114,0.99)'
            });
            $('#status').val("not yet started");
            $('#postBid').prop('disabled', true);
            $('#postBid').css({
                'background-color': 'rgba(114,114,114,0.99)',
                'border-color': 'rgba(114,114,114,0.99)'
            });
        } else if (currentDateTime > stopTime){
            $('#status').css({
                'background-color': 'rgba(249, 0, 0, 0.54)',
                'border-color': 'rgba(249, 0, 0, 0.08)'
            });
            $('#postBid').prop('disabled', true);
            $('#postBid').css({
                'background-color': 'rgba(114,114,114,0.99)',
                'border-color': 'rgba(114,114,114,0.99)'
            });
            $('#status').val("finished");
        } else {
            $('#status').css({
                'background-color': 'rgba(33, 136, 36, 0.54)',
                'border-color': 'rgba(33, 136, 36, 0)'
            });
            $('#status').val("in progress");
        }
    })
}

```

```

function setAuctionTime(batch) {
    $.ajax({
        type: "GET",
        url: batch + "/auction",
        contentType: "application/json",
        dataType: "json"
    }).then(function (data) {
        $('#startTime').val(data.start_time);

        $('#finishTime').val(data.stop_time);
    })
}

```

```

function POSTBid(formData) {
    $.ajax({
        type: "POST",
        url: "/bids",
        contentType: "application/json",
        data: formData,
    })
}

```

```

        dataType: "json",
        success: makeBidSuccess,
        error: ajaxError
    });
}

function makeBidSuccess(json, status, req) {
    window.alert("Successful bid made for £" + json.bid_amount);
    calculateHighestBid(json._links.batch.href);
    refreshBids();
}

function refreshBids() {
    restartForm();
    $('html, body').animate({
        scrollTop: $('#bids').offset().top
    }, 2000);
    $('#bidList').load("landingPage.html #bidList", loadBids(2000)).fadeIn("slow");
}

function restartForm() {
    $('#amount').val("");
    $('#selectGroup').val("");
    $('#status').val("");
    $('#status').css({
        'background-color': 'rgba(114,114,114,0.99)',
        'border-color': 'rgba(114,114,114,0.99)'
    });
    $('#postBid').prop('disabled', true);
    $('#startTime').val("");
    $('#finishTime').val("");
    $('#highestBid').val("");
}

function ajaxError(req, status, error)
{
    window.alert("Got error: " + status + "\nrequestStatus: " + req.status);
}

// DISPLAY BIDS
function displayYourBids() {
    $.ajax({
        type: "GET",
        url: $('#selectUser').val()+"/bids",
        contentType: "application/json",
        dataType: "json"
    }).then(function (data) {
        // initialise card deck and batch menu selection
        $('#bidList').append("<div id = 'bidCardDeck' class='card-deck justify-content-center' >");
        // cycle through each batch
        for (var dataitems = 0; dataitems < data._embedded.bids.length; dataitems++) {
            // add card to deck for each batch
            addBidCard(data._embedded.bids[dataitems]._links.self.href, dataitems);
        }
    });
}

function addBidCard(bid, item) {
    $.ajax({
        type: "GET",
        url: bid,
        contentType: "application/json",
        dataType: "json"
    }).then(function (data) {
        $('#bidCardDeck').append(
            "<div class='bid-col'>" +
            "<div class='card' id='card' + item + '>" +
            "<div class='card-body'>" +
            "<h5 class='card-title text-muted text-uppercase text-center'>Your Bid For: </h5>" +
            "<div id='batchInfo' + item + '></div>" +

```

```

        "<hr>" +
        "<label for='\"thisBidAmount\"'+item+'\"' class='\"card-title text-uppercase text-muted col-form-label\">Bid Amount</label>" +

        "<input type='\"text\"' readonly class='\"card-price text-uppercase bid-amount text-center form-control-plaintext\"' id='\"thisBidAmount\"'+item+'\" value='\"$\"'+data.bid_amount+'\">" +
        "<hr>" +

        "<input type='\"text\"' readonly='\"\"' class='\"btn card-price text-uppercase form-control-plaintext\"' id='\"status\"'+item+'\" name value='\"\"' disabled>" +
        "<br>" +
        "<input type='\"text\"' readonly='\"\"' class='\"btn bid-btn card-price text-uppercase form-control-plaintext\"' id='\"winOrLoss\"'+item+'\" name value='\"\"' disabled>" +
        "</div>" +
        "</div>" +
        "</div>");
    getBatchInfo(data._links.batch.href, item, data.bid_amount);
}

function getBatchInfo(batch, item, bidAmount) {
    $.ajax({
        type: "GET",
        url: batch,
        contentType: "application/json",
        dataType: "json"
    }).then(function (data) {
        $('#batchInfo'+item).append("<h6 class='\"card-price text-center text-uppercase text-center\">" +
        data.batch_name + "</h6>");
        calculateHighestBid(batch);
        setTimeout(function() {
            setBidStatusButton(data._links.auction.href, data.winning_bid, item, bidAmount, data), 2000);
        });
    })
}

function setBidStatusButton(auction, batchWinningBid, item, bidAmount, batchData) {
    $.ajax({
        type: "GET",
        url: auction,
        contentType: "application/json",
        dataType: "json"
    }).then(function (data) {
        var startTime = Date.parse(data.start_time);
        var stopTime = Date.parse(data.stop_time);
        var currentDateTime = new Date();
        if (currentDateTime > stopTime){
            $('#status'+item).css({
                'background-color': 'rgba(114,114,114,0.99)',
                'border-color': 'rgba(114,114,114,0.99)'
            });
            $('#status'+item).val("finished");
            determineWinOrLoss(batchWinningBid, item, bidAmount);
        } else {
            $('#status'+item).css({
                'background-color': 'rgba(33, 136, 36, 0.54)',
                'border-color': 'rgba(33, 136, 36, 0)'
            });
            $('#status'+item).val("in progress");
            determineWhetherHighestBid(batchWinningBid, item, bidAmount, batchData);
        }
    })
}

function determineWinOrLoss(batchWinningBid, item, bidAmount) {
    //alert("winning bid: " + batchWinningBid + ", bid: " + bidAmount);
    if(batchWinningBid === bidAmount){
        $('#winOrLoss'+item).css({

```

```

        'background-color': 'rgba(33, 136, 36, 0.54)',
        'border-color': 'rgba(33, 136, 36, 0)'
    });
    $('#winOrLoss'+item).val("Won");
} else {
    $('#winOrLoss'+item).css({
        'background-color': 'rgba(249, 0, 0, 0.54)',
        'border-color': 'rgba(249, 0, 0, 0.08)'
    });
    $('#winOrLoss'+item).val("Outbid");
}
}

function determineWhetherHighestBid(batchWinningBid, item, bidAmount, batchData) {
    //window.alert("batch number: " + item + ", winning bid: " + batchWinningBid + ", bid amount: " + bidAmount);
    if(batchWinningBid === bidAmount){
        $('#winOrLoss'+item).css({
            'background-color': 'rgba(33, 136, 36, 0.54)',
            'border-color': 'rgba(33, 136, 36, 0)'
        });
        $('#winOrLoss'+item).val("Highest Bid");
    } else {
        $('#winOrLoss'+item).css({
            'background-color': 'rgba(249, 0, 0, 0.54)',
            'border-color': 'rgba(249, 0, 0, 0.08)'
        });

        $('#winOrLoss'+item).val("Outbid");
        $('#winOrLoss'+item).attr({ 'name': batchData.winning_bid+1 });
    }
}

function checkPlate(plate) {
    $.ajax({
        type: "GET",
        url: "/check?plate=" + plate,
        contentType: "application/json",
        dataType: "json"
    }).then(function (data) {
        if (data.offensive === true) {
            window.alert("Offensive word detected for " + plate + " for its likeness to " + data.cuss + "");
        } else {
            window.alert("" + plate + " is OK");
        }
    })
}

function valuatePlate(plate) {
    $.ajax({
        type: "GET",
        url: "/valuate?plate=" + plate,
        contentType: "application/json",
        dataType: "json"
    }).then(function (data) {
        var rating;
        if (data.offensive === true) {
            if (data.cuss.length > 15){
                rating = "High";
            } else {
                rating = "Moderate";
            }
        }
        window.alert(rating+" valuation calculated for " + plate + " for its likeness to " + data.cuss + "");
    } else {
        window.alert("Low valuation calculated for " + plate + "");
    }
}

```

```

    })
  }
<!DOCTYPE html>
<html lang="en">

<head>

  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
  <meta name="description" content="">
  <meta name="author" content="">

  <title>Registration Plate Auction House</title>

  <!-- Bootstrap Core CSS -->
  <link href="bootstrap-template/vendor/bootstrap/css/bootstrap.min.css" rel="stylesheet">

  <!-- Custom Fonts -->
  <link href="bootstrap-template/vendor/fontawesome-free/css/all.min.css" rel="stylesheet" type="text/css">
  <link href="https://fonts.googleapis.com/css?family=Source+Sans+Pro:300,400,700,300italic,400italic,700italic"
rel="stylesheet" type="text/css">
  <link href="bootstrap-template/vendor/simple-line-icons/css/simple-line-icons.css" rel="stylesheet">

  <!-- Custom CSS -->
  <link href="bootstrap-template/css/stylish-portfolio.min.css" rel="stylesheet">
  <link rel="stylesheet" href="landingPage.css">

</head>

<body id="page-top">

  <!-- Navigation -->
  <a class="menu-toggle rounded" href="#">
    <i class="fas fa-bars"></i>
  </a>
  <nav id="sidebar-wrapper">
    <ul class="sidebar-nav">
      <li class="sidebar-brand">
        <a class="js-scroll-trigger" href="#page-top">Auction House</a>
      </li>
      <select class="form-control card-price text-muted text-uppercase" id="selectUser" name="selectUser"
onchange="refresh()">
        <option value="http://localhost:8080/bidders/1">Select User</option>
        <option value="http://localhost:8080/bidders/1">User 1</option>
        <option value="http://localhost:8080/bidders/2">User 2</option>
      </select>
      <li class="sidebar-nav-item">
        <a class="js-scroll-trigger" href="#page-top">Home</a>
      </li>
      <li class="sidebar-nav-item">
        <a class="js-scroll-trigger" href="#batches">Latest Batches</a>
      </li>
      <li class="sidebar-nav-item">
        <a class="js-scroll-trigger" href="#bids">Your Bids</a>
      </li>
      <li class="sidebar-nav-item">
        <a class="js-scroll-trigger" href="#testPlate">Test for offenses</a>
      </li>
      <li class="sidebar-nav-item">
        <a class="js-scroll-trigger" href="#valuation">Valuate</a>
      </li>
    </ul>
  </nav>

  <!-- Header -->
  <header class="masthead d-flex">

```

```

<div class="container text-center my-auto">
  <h1 class="mb-1 myshadow">Private Registration Plate Auction House</h1>
  <h3 class="mb-5 myshadow">
    <em>Bid on private plates for your company's fleet now</em>
  </h3>
  <a class="btn btn-primary btn-xl js-scroll-trigger" href="#batches">View Batches</a>
</div>
<div class="overlay"></div>
</header>

<!-- Latest Batches -->
<section class="content-section bg-primary text-white text-center" id="batches">
  <div class="container">
    <div class="content-section-heading">
      <h3 class="text-secondary mb-0">Batches</h3>
      <h2 class="mb-5 text-white">Latest Batches</h2>
    </div>
    <section class="pricing py-5">
      <div class="container">
        <div class="row justify-content-center">
          <!-- Batch List -->
          <div id="batchList" class="col-lg-12"></div>
        </div>
      </div>
    </section>
  </div>
</section>

<!-- Make Bid -->
<section class="content-section" id="makeBid">
  <div class="container">
    <div class="content-section-heading text-center">
      <h3 class="text-secondary mb-0">Bids</h3>
      <h2 class="mb-5">Make Bid</h2>
    </div>
    <section class="pricing py-5">
      <div class="container">
        <div class="row justify-content-center">
          <div class="col-lg-8">
            <div class="card">
              <div class="card-body">
                <form id="newBidForm">
                  <!-- Batch Selection -->
                  <div id="formBatchSelect"></div>
                  <br/>
                  <!-- auction start time -->
                  <label for="startTime" class="card-title text-uppercase text-muted col-form-label">Start Time</label>
                  <input type="text" readonly class="card-price text-uppercase form-control-plaintext" id="startTime"
value="">
                  <!-- auction finish time -->
                  <label for="finishTime" class="card-title text-uppercase text-muted col-form-label">End Time</label>
                  <input type="text" readonly class="card-price text-uppercase form-control-plaintext" id="finishTime"
value="">
                  <!-- Highest Bid -->
                  <label for="highestBid" class="card-title text-uppercase text-muted col-form-label">Highest
Bid</label>
                  <input type="text" readonly class="card-price text-uppercase form-control-plaintext" id="highestBid"
value=0>
                  <!-- auction status -->
                  <label for="status" class="card-title text-uppercase text-muted col-form-label">Status</label>
                  <input type="text" readonly="" class="btn card-price text-uppercase form-control-plaintext"
id="status" name="status" value="" disabled>
                  <!-- Bid amount -->
                  <label for="amount" class="card-title text-uppercase text-muted col-form-label">Bid
Amount</label><br>
                  <!-- Amount Selection -->
                  <div class="input-group mb-3 card-price text-uppercase">

```

```
<div class="input-group-prepend">
  <span class="input-group-text">£</span>
</div>
<input type="number" min="0" class="form-control" id="amount" name="amount">
<div class="input-group-append">
  <span class="input-group-text">.00</span>
</div>
</div>
<!-- Make Bid -->
<button type="submit" class="btn btn-block btn-primary text-uppercase" id="postBid">Bid</button>
</form>
</div>
</div>
</div>
</div>
</div>
</section>
</div>
</section>

<!-- Your Bids -->
<section class="content-section bg-primary text-white text-center" id="bids">
  <div class="container bids-cont">
    <div class="content-section-heading">
      <h3 class="text-secondary mb-0">Bids</h3>
      <h2 class="mb-5 text-white">Your Bids</h2>
    </div>
    <section class="pricing py-5">
      <div class="container bids-cont">
        <div class="row justify-content-center" id="bids">
          <div class="text-center text-light" id="loading">
            <div class="spinner-border" role="status">
              <span class="sr-only">Loading...</span>
            </div>
          </div>
          <!-- Bid List -->
          <div id="bidList" class="col-lg-12"></div>
        </div>
      </div>
    </section>
  </div>
</section>

<!-- Test for offenses -->
<section class="content-section" id="testPlate">
  <div class="container">
    <div class="content-section-heading text-center">
      <h3 class="text-secondary mb-0">Misc</h3>
      <h2 class="mb-5">Test a plate for offenses</h2>
    </div>
    <section class="pricing py-5">
      <div class="container">
        <div class="row justify-content-center">
          <div class="col-lg-8">
            <div class="card">
              <div class="card-body">
                <form id="newPlateForm">
                  <h6 class="card-price text-center text-uppercase text-center">ENTER PLATE TO TEST</h6>
                  <input type="text" class="card-price text-uppercase text-center form-control-plaintext textBoxStyling"
id="plate" placeholder="registration plate" value="" pattern="[A-Za-z]{2}[0-9]{2}[A-Za-z]{3}" title="e.g. AB12CDE">
                  <button type="submit" class="btn btn-block btn-primary text-uppercase" id="postPlate">Check for
offenses</button>
                </form>
              </div>
            </div>
          </div>
        </div>
      </div>
    </section>
  </div>
</section>
```

```

        </div>
    </div>
</div>
</section>
<h3 class="container-footer-black text-secondary mb-0">OUR WEB SERVICE USES AN ALGORITHM TO
DECIPHER OFFENSIVE WORDS IN ALL OUR REGISTRATION PLATES</h3>
</div>
</section>

<!-- Get valuation -->
<section class="content-section bg-primary text-white text-center" id="valuation">
<div class="container bids-cont">
    <div class="content-section-heading">
        <h3 class="text-secondary mb-0">MISC</h3>
        <h2 class="mb-5 text-white">Get a valuation for a plate</h2>
    </div>
    <section class="pricing py-5">
        <div class="container">
            <div class="row justify-content-center">
                <div class="col-lg-8">
                    <div class="card">
                        <div class="card-body">
                            <form id="valuateForm">
                                <h6 class="card-price text-center text-uppercase text-center">ENTER PLATE TO TEST</h6>
                                <input type="text" class="card-price text-uppercase text-center form-control-plaintext textBoxStyling"
id="plateToBeValued" placeholder="registration plate" value="" pattern="[A-Za-z]{2}[0-9]{2}[A-Za-z]{3}" title="e.g.
AB12CDE">
                                <button type="submit" class="btn btn-block btn-primary text-uppercase"
id="valuePlate">Valuate</button>
                            </form>
                        </div>
                    </div>
                </div>
            </div>
        </div>
    </section>
    <h3 class="container-footer-white text-secondary mb-0">OUR WEB SERVICE CAN USE AN ALGORITHM
TO VALUATE REGISTRATION PLATES</h3>
</div>
</section>

<!-- Footer -->
<footer class="footer text-center">
<div class="container">
    <p class="text-muted small mb-0">Bootstrap along with its templates and snippets were used in the making
of this web-page and can be found at <a href="https://startbootstrap.com/themes/stylish-
portfolio/">https://startbootstrap.com/themes/stylish-portfolio/</a>, <a
href="https://startbootstrap.com/snippets/pricing-table/">https://startbootstrap.com/snippets/pricing-table/</a>
and <a href="https://startbootstrap.com/snippets/sign-in-split/">https://startbootstrap.com/snippets/sign-in-
split/</a></p>
    <p class="text-muted small mb-0">FontAwesome was used in the making of this web page and can be found
at <a href="https://fontawesome.com">https://fontawesome.com</a></p>
</div>
</footer>

<!-- Scroll to Top Button-->
<a class="scroll-to-top rounded js-scroll-trigger" href="#page-top">
    <i class="fas fa-angle-up"></i>
</a>
<!-- Bootstrap core JavaScript -->
<script src="bootstrap-template/vendor/jquery/jquery.min.js"></script>
<script src="bootstrap-template/vendor/bootstrap/js/bootstrap.bundle.min.js"></script>

<!-- Plugin JavaScript -->
<script src="bootstrap-template/vendor/jquery-easing/jquery.easing.min.js"></script>

```

```
<!-- Custom scripts for this template -->
<script src="bootstrap-template/js/stylish-portfolio.min.js"></script>

<script src="home.js"></script>

</body>

</html>
```

```
.pricing .card {
  border: none;
  border-radius: 1rem;
  transition: all 0.2s;
  box-shadow: 0 0.5rem 1rem 0 rgba(0, 0, 0, 0.1);
}
```

```
.pricing .card-body{
  color: rgba(0, 0, 0, 0.99);
  padding-bottom: 10px;
}
```

```
.pricing .card-title {
  margin: 0.5rem 0;
  font-size: 0.9rem;
  letter-spacing: .1rem;
  font-weight: bold;
}
```

```
.pricing .card-price {
  font-size: .9rem;
  margin: 0;
  letter-spacing: .1rem;
  font-weight: bold;
}
```

```
.pricing .card-price .period {
  font-size: 0.8rem;
}
```

```
.pricing ul li {
  margin-bottom: 0.5rem;
}
```

```
.pricing .text-muted {
  opacity: 0.7;
}
```

```
.card-deck .card {
  padding-bottom: .5rem;
  padding-left: .5rem;
  padding-right: .5rem;
}
```

```
.pricing .btn {
  font-size: 80%;
  border-radius: 5rem;
  letter-spacing: .1rem;
  font-weight: bold;
  padding: 1rem;
  opacity: 0.7;
  transition: all 0.2s;
}
```

```
.fa-ul{list-style-type:none;margin-left:0em;padding-left:0}
```

```
.col-lg-4{
  padding-bottom: 50px;
}
```

```
.content-section {
  padding-top: 5rem;
  padding-bottom: 7.5rem;
}
```

```

}

/* Hover Effects on Card */

@media (min-width: 992px) {
  .pricing .card:hover {
    margin-top: -.25rem;
    margin-bottom: .25rem;
    box-shadow: 0 0.5rem 1rem 0 rgba(0, 0, 0, 0.3);
  }
  .pricing .card:hover .btn {
    opacity: 1;
  }
}

#postBid.disabled {
  background-color: rgba(114,114,114,0.99) !important;
  border-color: rgba(114,114,114,0.99) !important;
}

.hr {
  margin-top: 0rem;
  margin-bottom: 0rem;
  border: 0;
  border-top: 1px solid rgba(0,0,0,.1);
}

#status{
  color: rgba(255,255,255,0.99)
}

.myshadow {
  text-shadow: -1px -1px 0 rgba(255, 255, 255, 0.26), 1px -1px 0 rgba(255, 255, 255, 0.29), -1px 1px 0
  rgba(255, 255, 255, 0.2), 1px 1px 0 rgba(255, 255, 255, 0.2) !important;
}

.btn {
  color: #ffffff !important;
}

.bid-btn {
  margin-top: 1rem !important;
}

.bids-cont {
  max-width: 100% !important;
}

.bid-col {
  padding-right: 15px;
  padding-left: 15px;
  padding-bottom: 50px;
  width: 380px;
}

.textBoxStyling{
  margin-top: 1.5rem !important;
  margin-bottom: 1.5rem !important;
  font-size: 200% !important;
  border: 1px solid #ced4da;
}

::placeholder {
  color: rgba(158, 165, 169, 0.51)
}

#loading{
  display: none;
}

.container-footer-white{
  font-size: 1rem !important;
}

```

```
text-align: center;
color: #ffffff !important;
}
.container-footer-black{
font-size: 1rem !important;
text-align: center;
color: #000000 !important;
}
#selectUser {
margin-left: .5rem;
width: 230px;
}
```

1809709 - Web Services and Web Development

GRADEMARK REPORT

FINAL GRADE

91 / 100

GENERAL COMMENTS

Instructor

Part 1(50%): 47

Part 2 (25%): 20

Part 3 (25%): 24

Part 1

Excellent bad language elimination and spotting of high value number plates. A good, well-considered approach to grouping plates. Login system spoofed, which is acceptable for a prototype. An excellent, detailed implementation which shows a robust knowledge of the principles.

Part 2

Good analysis of the implementation of REST, but I wanted a little more evaluation of what is considered to be the "best" practice e.g. you may common design patterns.

Part 3

Very detailed discussion of serverless technology and the CB pattern.

PAGE 1

PAGE 2

PAGE 3

PAGE 4

PAGE 5

PAGE 6

PAGE 7

PAGE 8

PAGE 9

PAGE 10

PAGE 11

PAGE 12

PAGE 13

PAGE 14

PAGE 15

PAGE 16

PAGE 17

PAGE 18

PAGE 19

PAGE 20

PAGE 21

PAGE 22

PAGE 23

PAGE 24

PAGE 25

PAGE 26

PAGE 27

PAGE 28

PAGE 29

PAGE 30
