

**PetDesk Appointments Web API**

**Revision Record**

|  |  |  |
| --- | --- | --- |
| Date | Sections | Comments |
| 09/13/2019 | All | Document Created |

**Contributors**

|  |  |
| --- | --- |
| Name | Organization |
| Kevin Henzel | n/a |
|  |  |
|  |  |
|  |  |

|  |  |
| --- | --- |
| **Preliminary Estimate (hours):** | 10 |
| **Final Estimate (hours):** | 12 |

|  |  |
| --- | --- |
| Prepared by:  Kevin Henzel |  |
| Last Revision Date: 09/13/2019 |  |
|  |  |

Contents

[1.1 Solution Overview 3](#_Toc19294415)

[1.2 Project Scope 3](#_Toc19294416)

[1.3 Assumptions 3](#_Toc19294417)

[1.4 Constraints 3](#_Toc19294418)

[1.5 Dependencies (if applicable) 3](#_Toc19294419)

[2. High Level Design 3](#_Toc19294420)

[2.1 Background 3](#_Toc19294421)

[2.2 Security 3](#_Toc19294422)

[2.3 Scalability 3](#_Toc19294423)

[2.4 BCP\Availability 4](#_Toc19294424)

[2.5 Process Flows 4](#_Toc19294425)

[2.5.1 Pre-Implementation Flow Diagrams 5](#_Toc19294426)

[3. Design Specifications 6](#_Toc19294427)

[3.1 Solution Description 6](#_Toc19294428)

[3.1.1 JSON Service – petDeskGetAppointments.aspx : 6](#_Toc19294429)

[3.1.2 Web API – WebPetDeskApi.sln : 6](#_Toc19294430)

[3.1.3 SQL SPROC’s: 6](#_Toc19294431)

[3.2 Presentation Layer: User Interface Design 6](#_Toc19294432)

[3.2.1 Feature #1 (JSON Service) 6](#_Toc19294433)

[3.2.2 Feature #2 (Database SPROCS for storing/gathering data) 6](#_Toc19294434)

[3.2.3 Feature #3 (Web API) 6](#_Toc19294435)

[3.2.4 Feature #4 (Web UI) 6](#_Toc19294436)

[3.3 Report Requirements 6](#_Toc19294437)

[3.4 Data Layer: Physical Data Model Representations 7](#_Toc19294438)

[3.4.1 Data Base Specifications 7](#_Toc19294439)

[3.5 Stored Procedures/Functions/Views 7](#_Toc19294440)

[4. Interface Layout 7](#_Toc19294441)

[5. Testing 9](#_Toc19294442)

[6. Appendix A. References Documents 9](#_Toc19294443)

Introduction

## Solution Overview

At PetDesk one thing we do is give pet owners the ability to easily request appointments (appts) with their veterinarian (vet) through a mobile app. Currently, the vet receives and responds to these requests through a web application. The veterinarian would like some insight / reporting about the requests they’ve been receiving.

## Project Scope

This process will consist of a JSON Service (aspx page) to collect client data from an open authentication API, a database to store only the relevant client data, and a web API to deliver reporting data to the caller. The reporting API must utilize bearer token validation. The presentation layer will be designed on an ASP.NET web platform that will provide a seamless login process for the vet to request their reports.

## Assumptions

It is assumed that the data collected is for one customer that is the target of the reporting, and that no additional datapoints are required at this time beyond the clearly stated requirements.

## Constraints

No constraints.

## Dependencies (if applicable)

This process relies on availability of the host open validation API get to deliver the payload to the JSON service for this process to function. Without this service no data will be collected, resulting in an empty report.

# High Level Design

## Background

We need to collect key datapoints from the client system to present to them the frequency of appointment types and visits per month to help them analyze their business. Through requirements gathering it was discussed that two endpoints would be a feasible approach at collecting/returning this information to the client.

## Security

Though the target system has open authentication to retrieve the datapoints for this process, requirements have been presented to safeguard the newly created API endpoints against open access. For this point, we need to present Bearer token security for access.

Additionally, as the host database authentication access should not be visible at any place in this process, the database connection will be secured through a private utility library for security.

## Scalability

The current requirements for this process target only one single vet clinic. To future proof the design of this application to support additional clinics, adjustments to the requirements would need to be considered allowing dynamic endpoints for the base API retrieval to be able to process other clients. Additional changes would need to be made as well to the database, web API, and UI to support multiple users.

## BCP\Availability

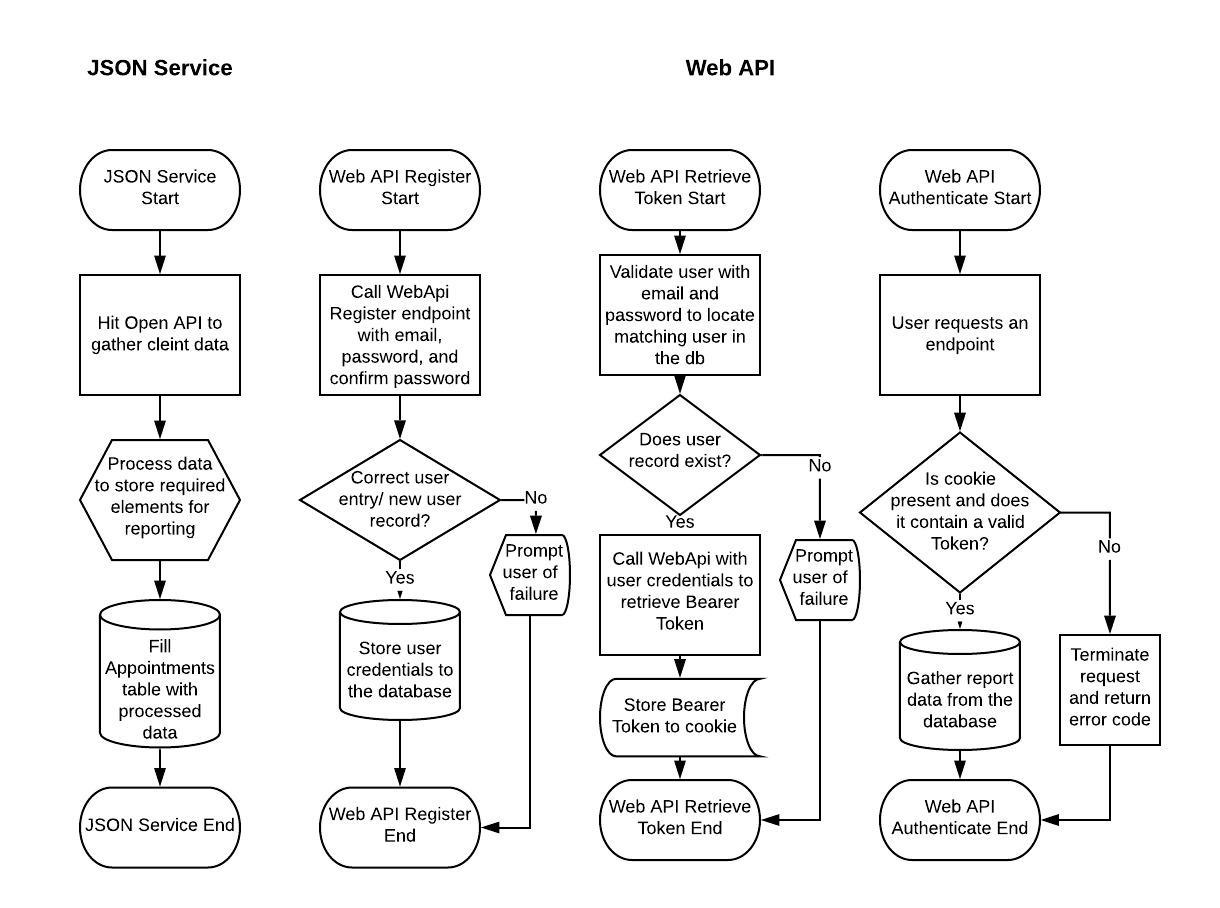
As no sensitive data is ever stored in the database, there is no need for business continuity planning as there is no risk associated to the process. In the event of a disaster there is no threat of data loss as the source tables are truncated with every pass of the application lifecycle.

## Process Flows

The process flow consists of 4 parts:

* JSON Service – pulls the client data from an open auth API GET call
* MS SQL Database Store – stores the data collected from the JSON Service
* Web API – two endpoints created to return both sets of data collected from the database to the process caller via Bearer Token Authentication
* Presentation Layer – Web UI interface that allows the user to carry out the following functions:
  + Create User – provided email and password
  + Retrieve Token (Login) –
    - Retrieves the Bearer Token from the API for reporting calls
    - Upon successful login, hits the JSON Service to refresh the client data to the database
  + Display report to GridView upon Bearer Token Authentication

### Pre-Implementation Flow Diagrams



# Design Specifications

## Solution Description

### JSON Service – petDeskGetAppointments.aspx :

1. Read the data from the API endpoint <https://sampledata.petdesk.com/api/appointments>
   1. An Appointment contains the following:
      1. Details on the appointment (type and requested datetime)
      2. Client Data (name of the person requesting appt)
      3. Patient Data (name and type of pet the appt is for)
2. Save required data to a database/datastore
3. Provide aggregate reporting data through API endpoints you create
4. API endpoints should require a valid HTTP bearer token to be accessed

### Web API – WebPetDeskApi.sln :

The reporting endpoints you create should be able to answer the following questions a vet might have:

* + - 1. What is the distribution in types of appointments we have?
      2. How many appointment requests do we get a month?

### SQL SPROC’s:

1. usp\_get\_petdesk\_appointments.sql
2. usp\_petdesk\_insert\_appointments.sql

## Presentation Layer: User Interface Design

The interface will utilize the existing SolutionsWeb website interface to host end user interaction with this reporting process. This will control user account creation, account login validation, token generation, and report rendering.

### Feature #1 (JSON Service)

Service that pulls data from the open api, processes it to acceptable form, and stores it to the database for the Web API to request/deliver.

### Feature #2 (Database SPROCS for storing/gathering data)

Sprocs that will commit processed data from the JSON service to the database and retrieve that data via parameter for one of two data collections for the two application end points.

### Feature #3 (Web API)

API offering two endpoints for gathering of data, as well as account creation and token generation for user authentication via Bearer Token.

### Feature #4 (Web UI)

Web interface to string the JSON service and web API together to simplify user interactions.

## Report Requirements

Three fields are to be returned for both reports:

1. Type – Report type. Reportable values are
   1. apptRequest (for the appointment request endpoint)
   2. apptType (for the appointment type endpoint).
2. Value – Stores the value of the record. Records consist of
   1. Appointment month-year (for the appointment request endpoint)
   2. Appointment type (for the appointment type request endpoint)
3. Frequency – number of occurrences for the current record (applies to both endpoints)

## Data Layer: Physical Data Model Representations

### Data Base Specifications

#### New Tables

One new table will be introduced with this process:

##### ***dbo.tblApplicationLog***

| Column | Nullable | DataType | Key (P/F) | Indexes |
| --- | --- | --- | --- | --- |
| Type | Not null | Varchar(15) |  |  |
| Value | Not null | Varchar(50) |  |  |
| Frequency | Not null | Int |  |  |

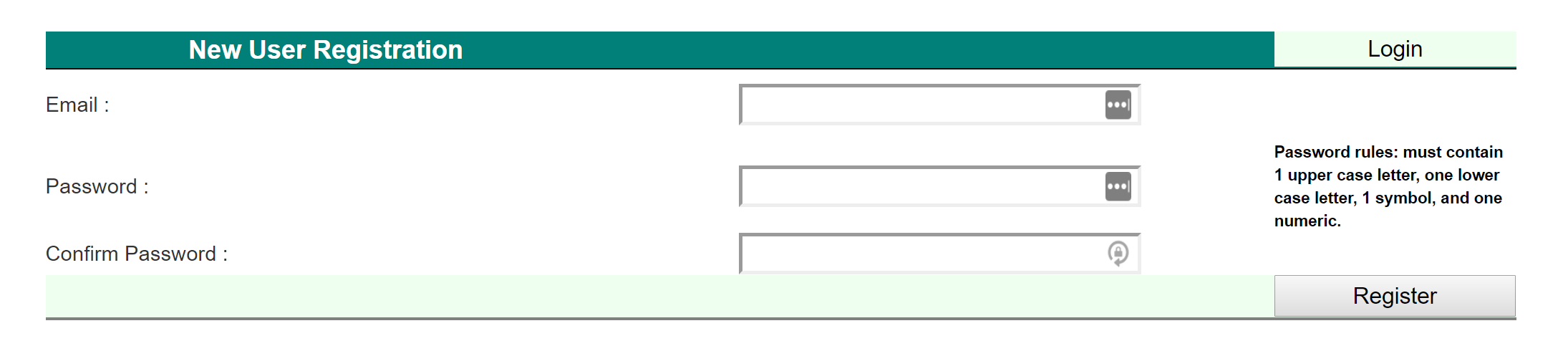
## Stored Procedures/Functions/Views

1. usp\_get\_petdesk\_appointments.sql
   1. Returns results from the tbl\_PetDesk\_Appointments\_Frequency table for use with PetDeskAPI
   2. Parameters:
      1. @TypeFilter TINYINT = 0 -- 0 = all, 1 = Request, 2 = Type
2. usp\_petdesk\_insert\_appointments.sql
   1. Update or insert bulk records to the tbl\_PetDesk\_Appointments\_Frequency temp table. This is called by petDeskGetAppointments service to store frequency values for AppointmentTypeFrequency API and AppointmentRequestFrequency API

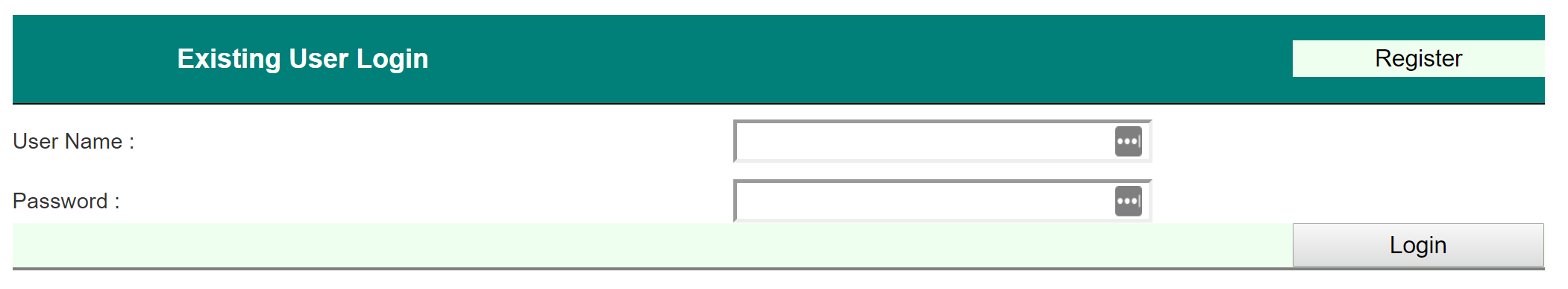
Parameters: n/a

# Interface Layout

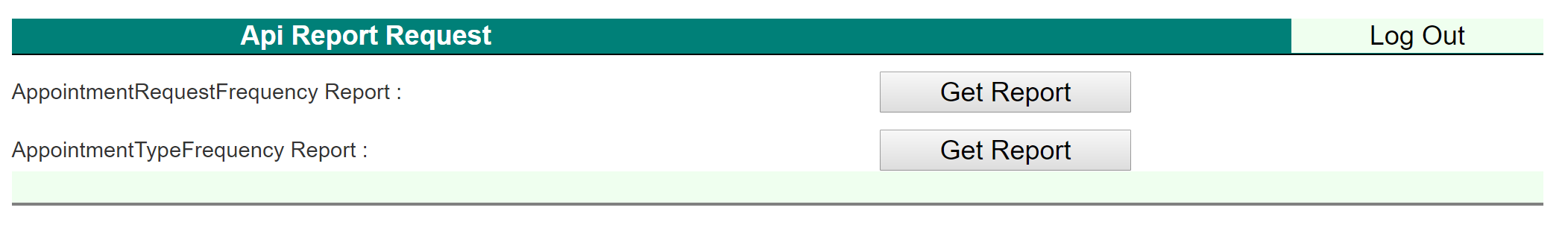
* Below is a printout of the web UI design:
* New User Registration:



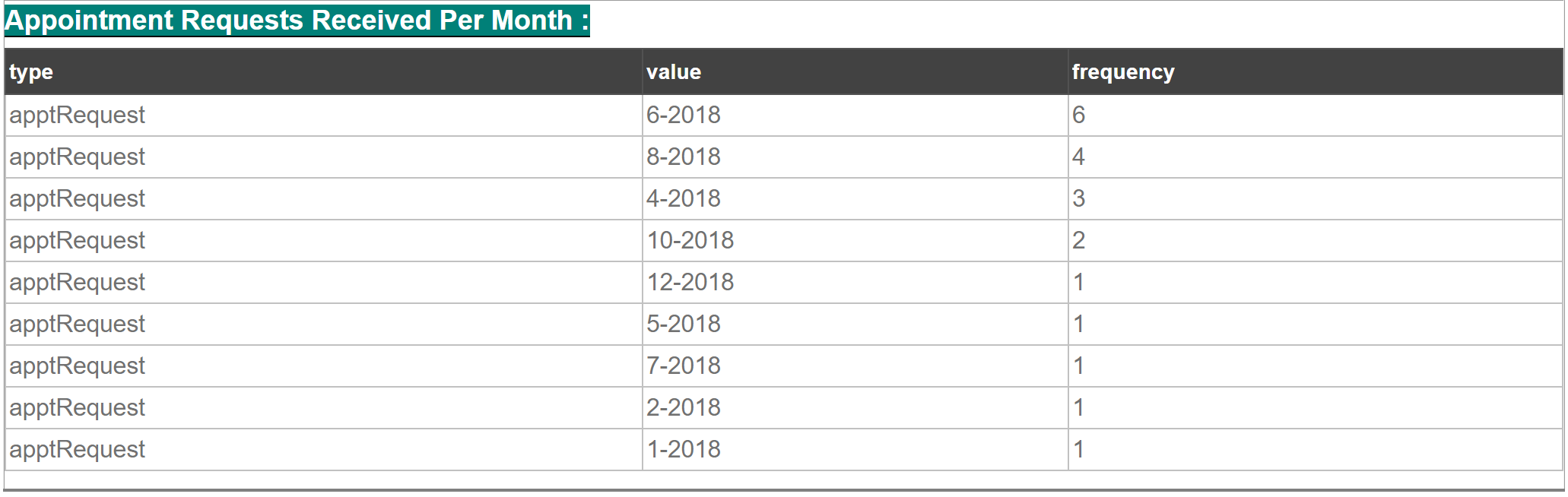
* Existing User Login:



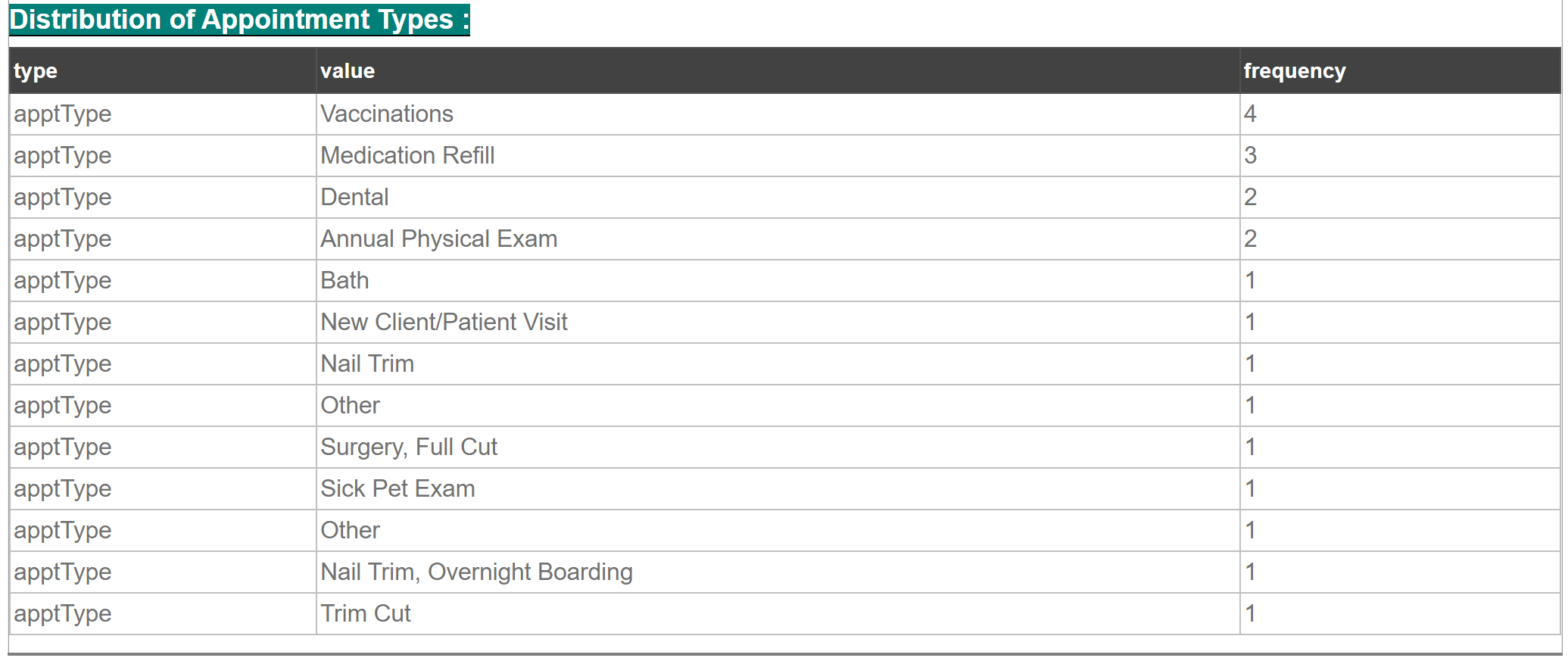
* API Report Request:



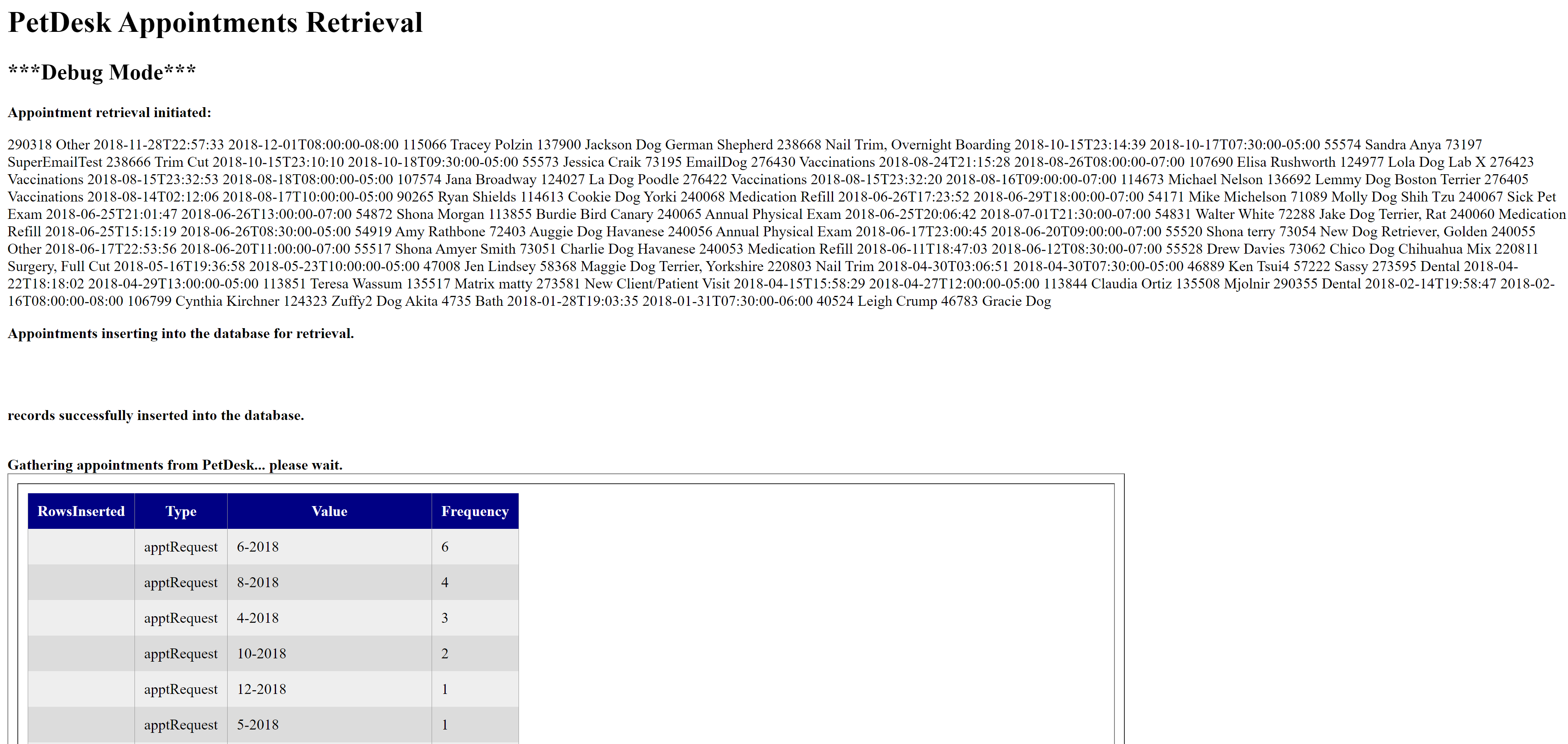
* Appointment Requests Received Per Moth (Report):



* Distribution of Appointment Types (Report):



* JSON Service output:
* Debug mode output:



* Standard run-time output:



# Testing

The following testing will be done during Unit Testing. These tests may be duplicated by QA & UAT, however, additional tests will need to be drafted up.

1. The JSON service is pulling records from the open API.
   1. **Testing Expectation:** when the call is made to the provided source API, records are retrieved with no authentication needed from the API.
2. The JSON service is performing appropriate calculations to the deserialized data and is inserting them to the database for retrieval.
   1. **Testing Expectation:** Expected frequency calculation is being performed on the source data.
   2. **Testing Expectation:** The groomed data is being inserted to the database without error.
3. The Web API is pulling records from the database for both endpoints.
   1. **Testing Expectation:** For both endpoints a record count of the return from the database is producing records.
4. The Web UI is allowing users to properly create new accounts.
   1. **Testing Expectation:** When a user enters an invalid email format a rejection occurs.
   2. **Testing Expectation:** When a user enters an invalid password format (requires one upper case, one lower case, one numeric, and one symbol) a rejection occurs.
   3. **Testing Expectation:** When a user enters an invalid confirmation password a rejection occurs.
   4. **Testing Expectation:** When a user attempts to create a duplicate user already existing in the database, an error occurs.
   5. **Testing Expectation:** When all above requirements are met, the user can successfully create an account.
5. The Web UI Allows users to log in successfully.
   1. **Testing Expectation:** When the user improperly validates, login is rejected.
   2. **Testing Expectation:** When the user properly validates, login is granted.
   3. **Testing Expectation:** Upon successful login, the user is passed to the JSON service for data refresh.
   4. **Testing Expectation:** Upon successful login, the access\_token is stored as a cookie to the user’s browser under key “PetDeskApiToken”.
   5. **Testing Expectation:** When the user logs out, the cookie is removed from the browser.
   6. **Testing Expectation:** When the user requests a report from the API Report Request panel, the user is successfully authenticated against the API to produce a report

**Explorative Items:**

n/a

# Appendix A. References Documents

Document Location

| Project Name | Project Directory path |
| --- | --- |
| PetDeskAppointmentsAPI | PetDesk application repository location: <https://github.com/khenzel/PetDeskAppointmentsAPI> |
|  |  |

This page intentionally left blank