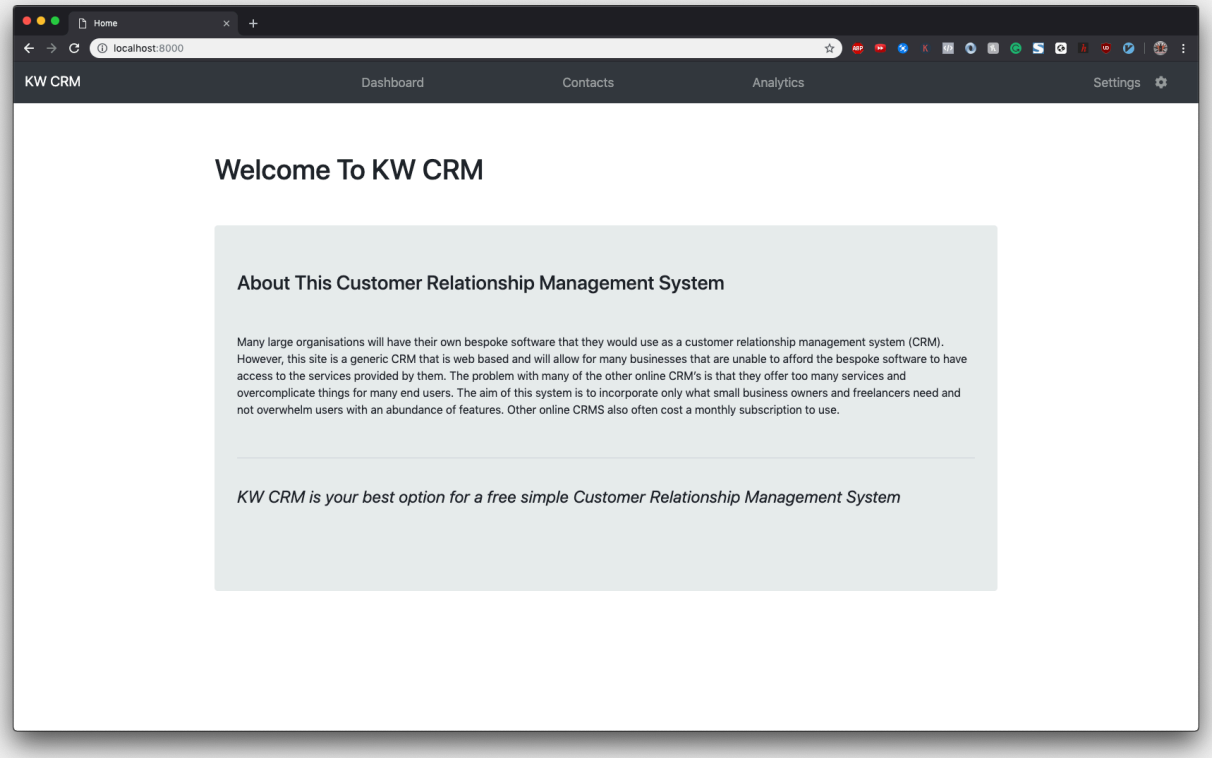
# System Maintenance Documentation

## Customer Relationship Management System (CRM)

**Kieran Williams**

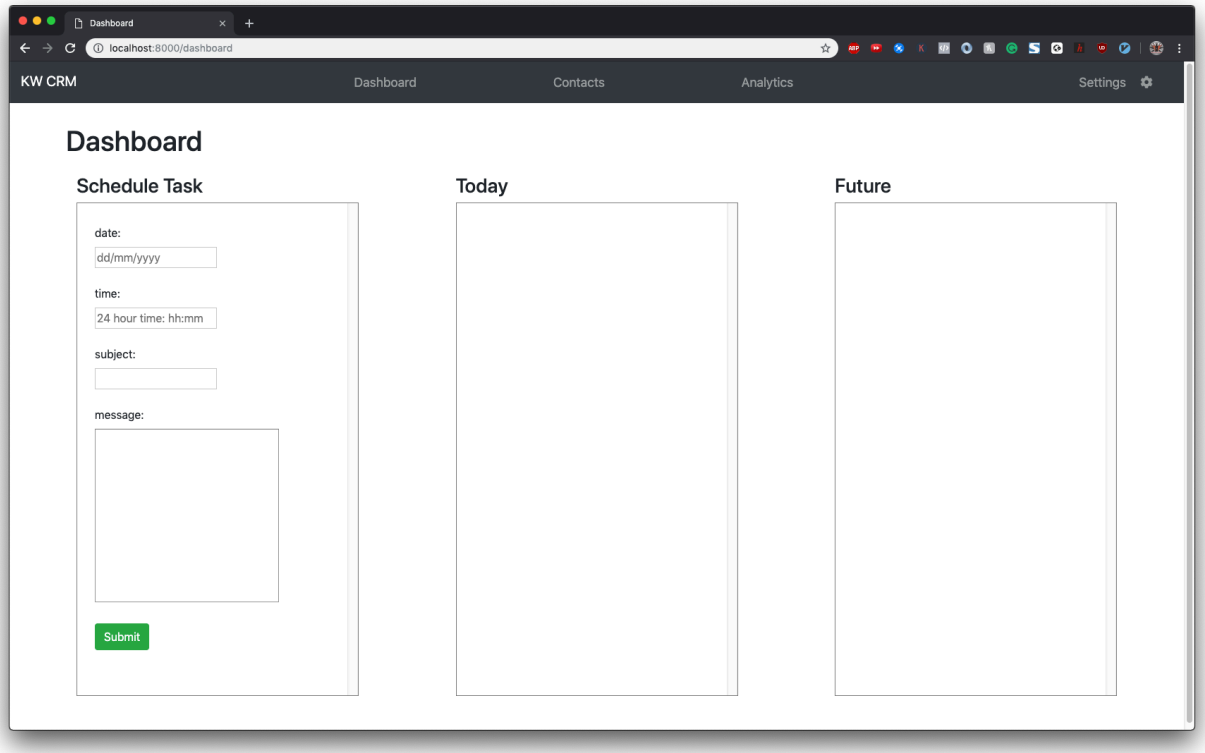
UI And All Validation Messages

## Home Page

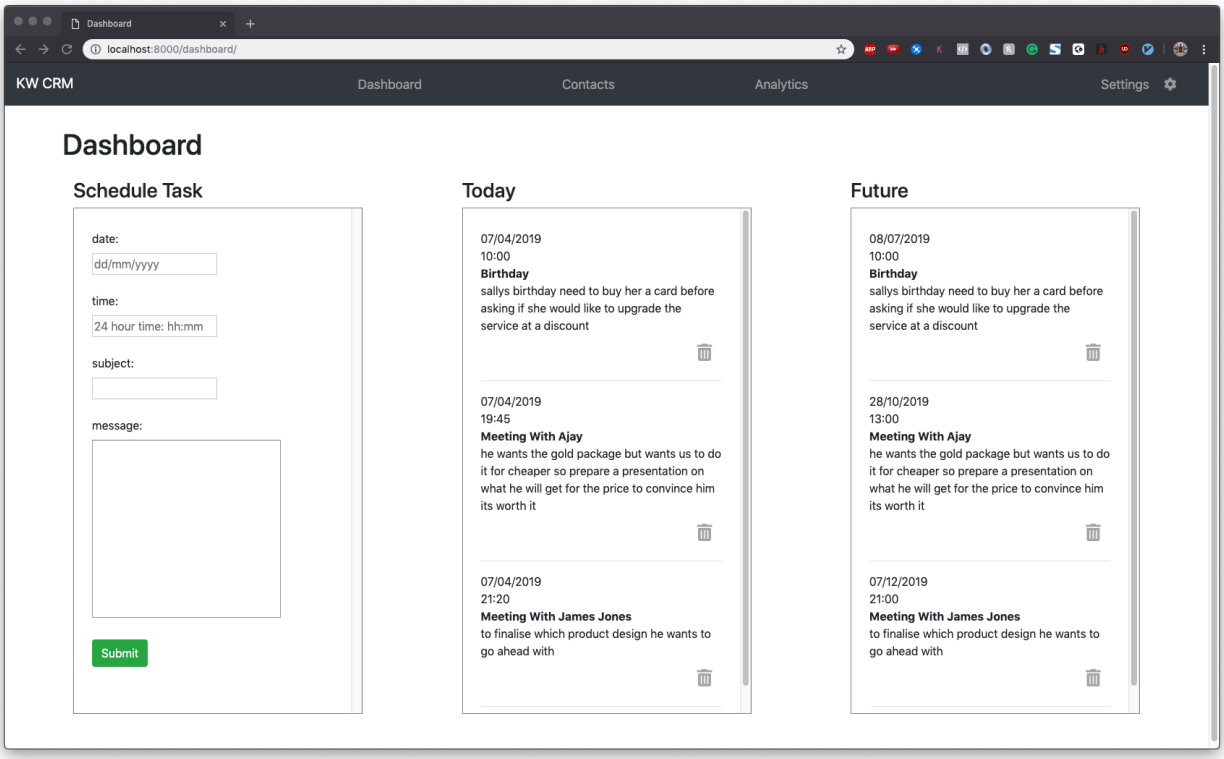



The home page has no features other than a greeting and description of the system and who it is aimed towards

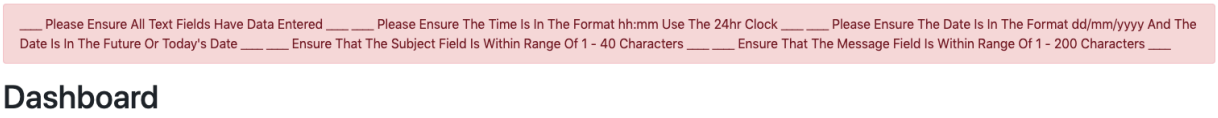
## The Dashboard




The Dashboard has the ability to add, view and delete scheduled tasks



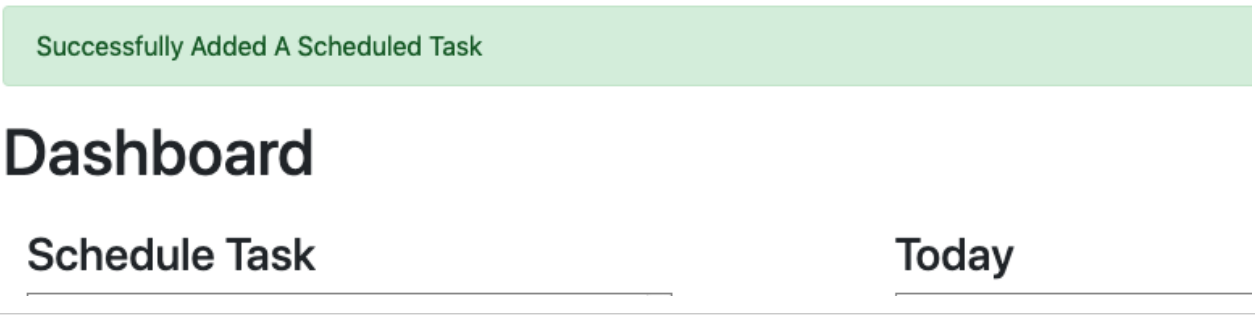

This is a populated dashboard where all the scheduled tasks for both the day and the future are displayed



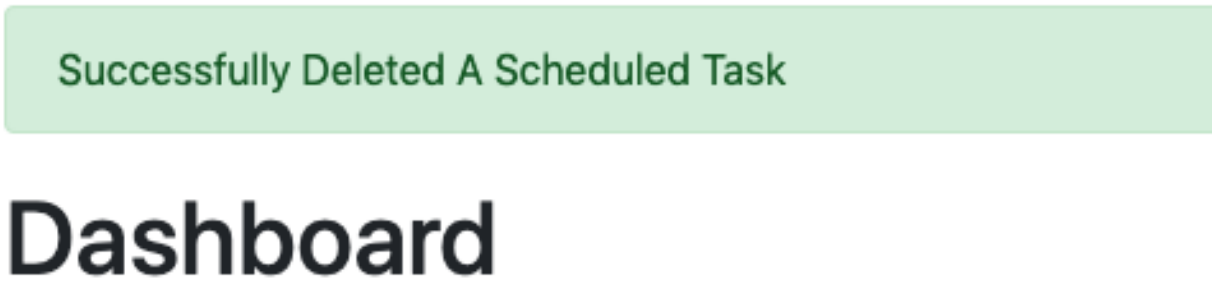




These are all the possible validation messages that can get displayed due to invalid data entry in the schedule task form

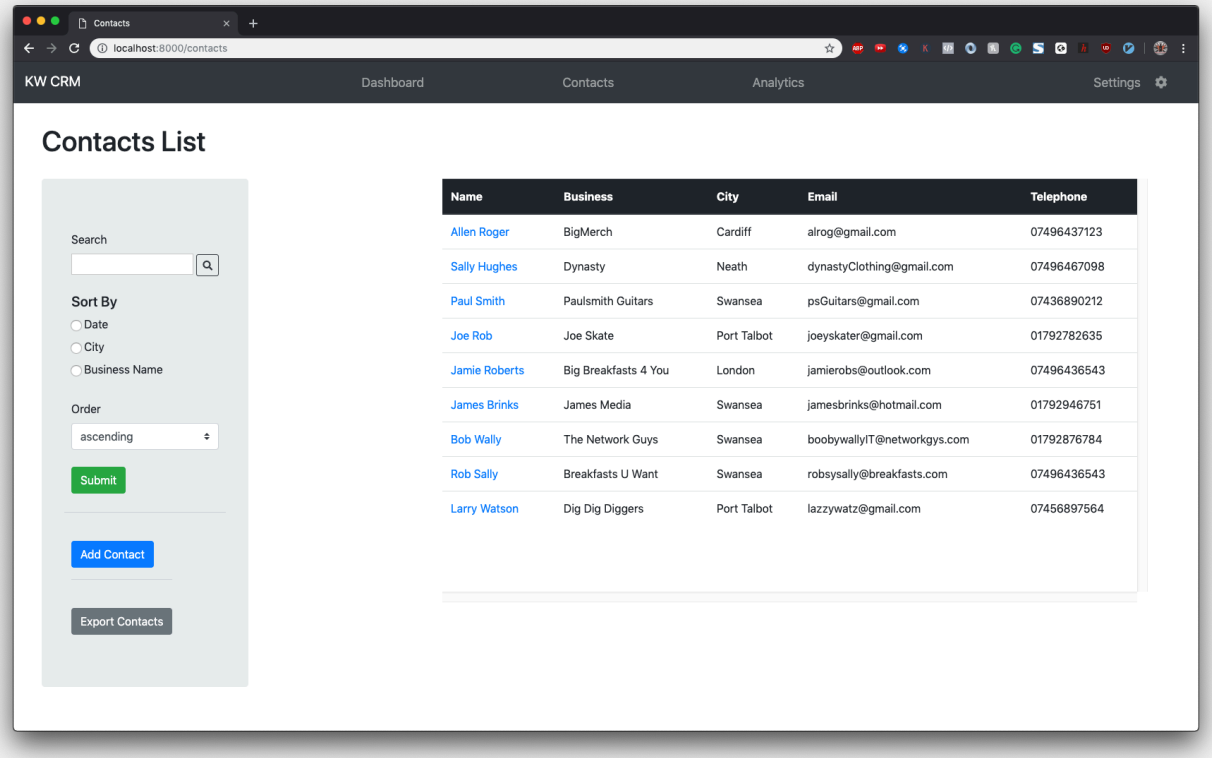



This message get's displayed when the user successfully adds a task



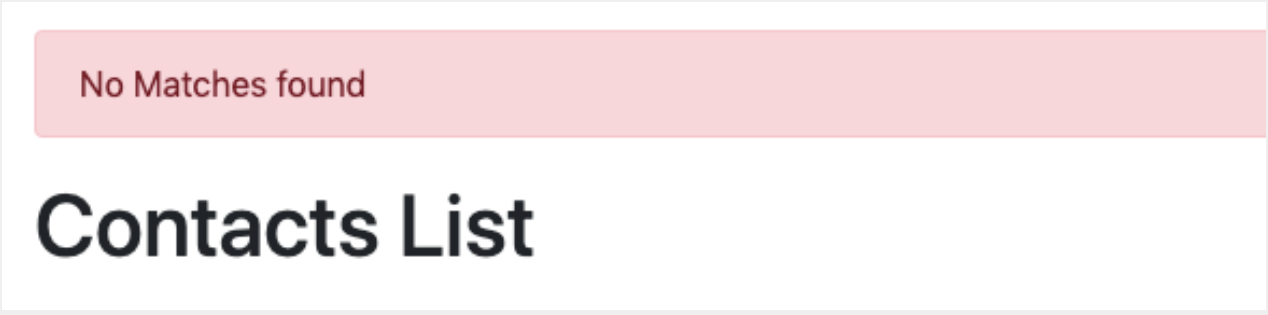
This message get's displayed when the user successfully deletes a task

**Contacts List Page**

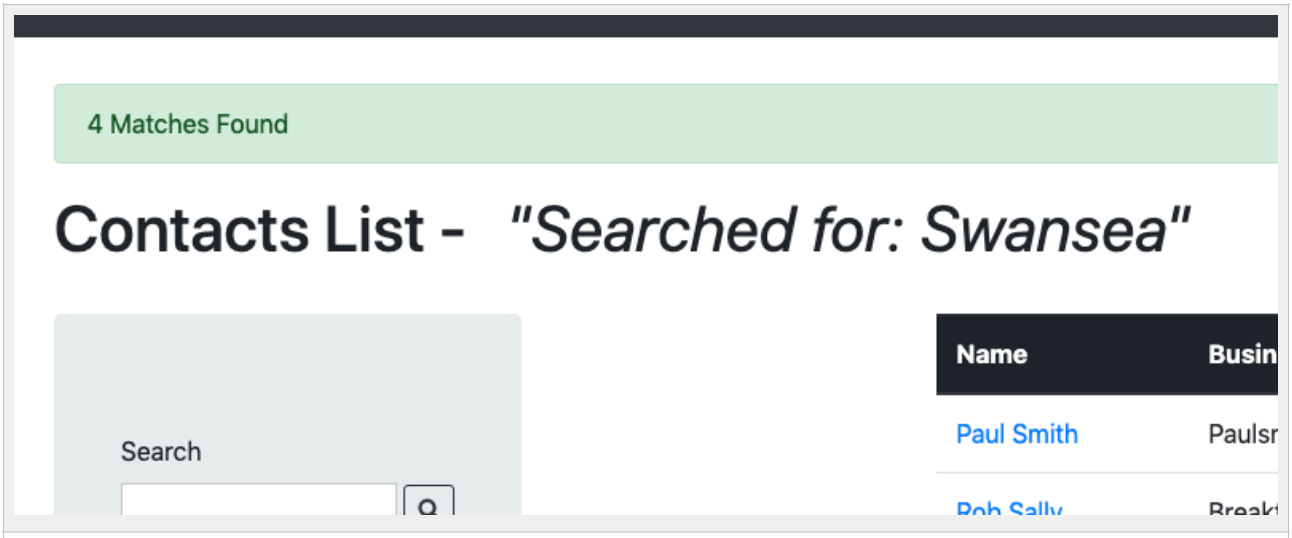



This page displays all the contacts in the data store (to view the contacts individual page, the name of the contact is to be clicked as it is a link to their page). You can search the list with the search bar. You can sort the list in both ascending and descending order. You can go to the add contact page from clicking the add contact button and download all the contact's data by clicking the export contacts button.

The purpose of the page is to display the contacts that are stored to the user and provide ways in which to find or narrow down contacts within that list.

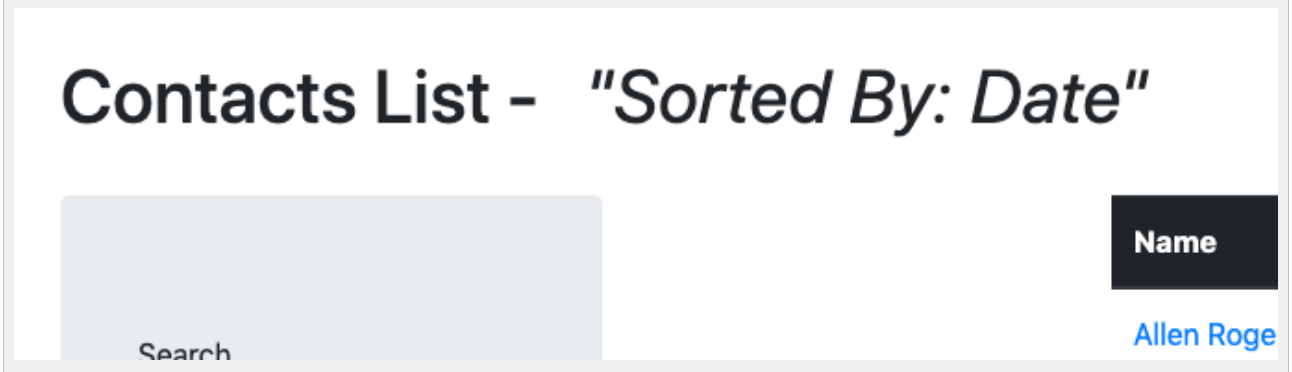


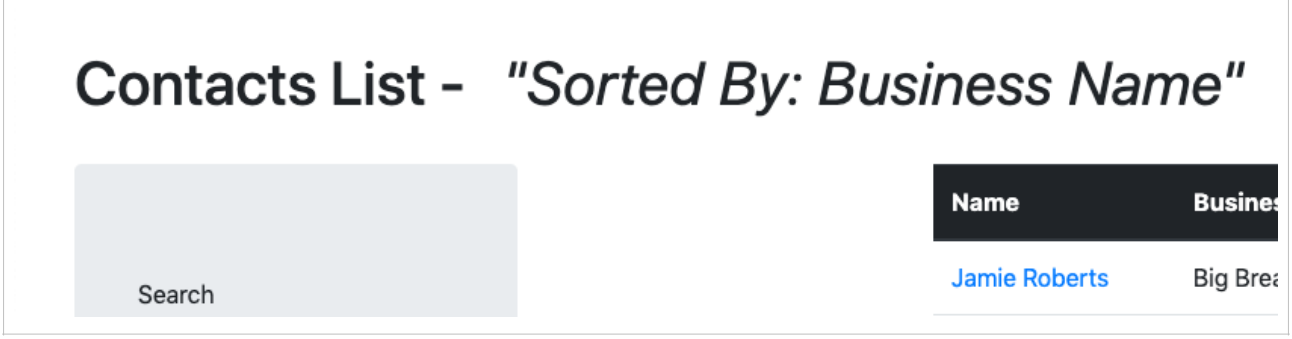

When a search value that has no matches in the contacts datastore is entered, this message is displayed



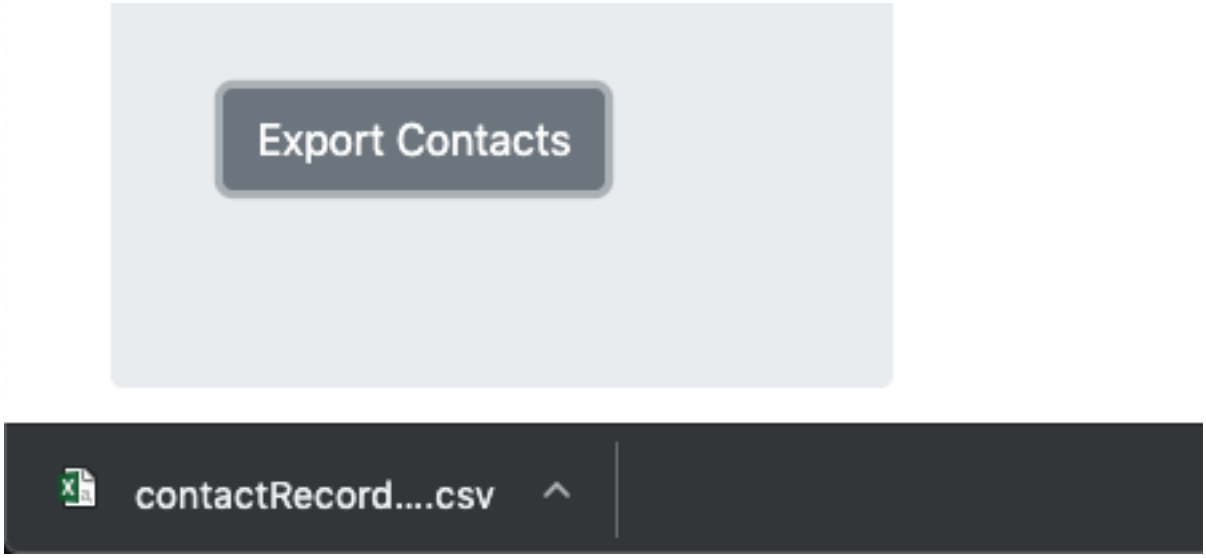

When a search value is entered where there are matches, these messages gets displayed and the contacts list only shows the records that contain the matching search value

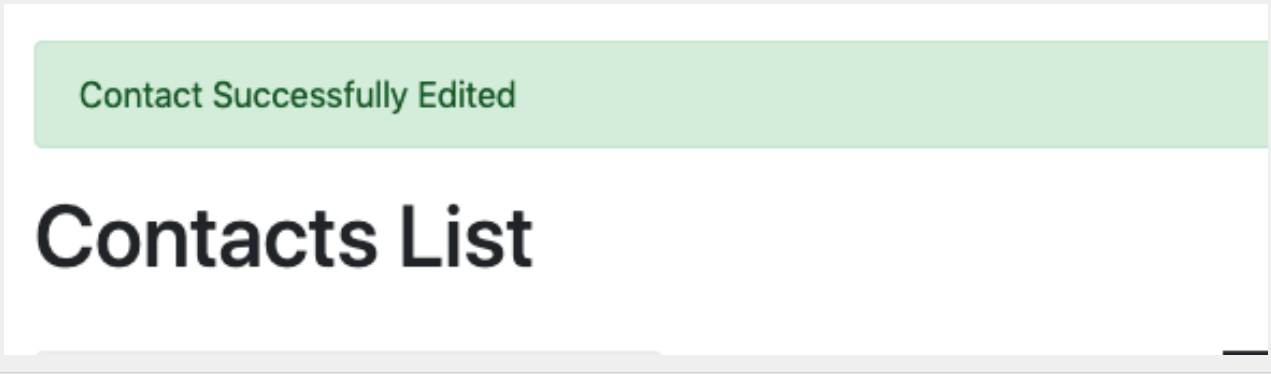
These messages get displayed when the user sorts the contacts list and then the contacts list gets sorted in either ascending or descending order in the column that the list is being sorted by

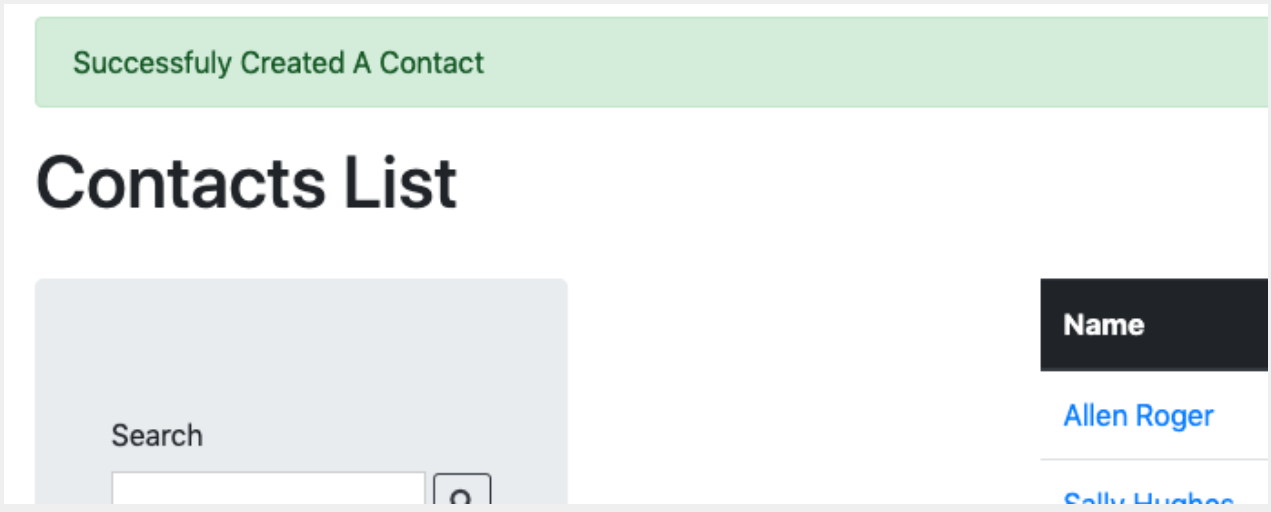



Pressing the export contacts button downloads the contacts data onto the clients computer



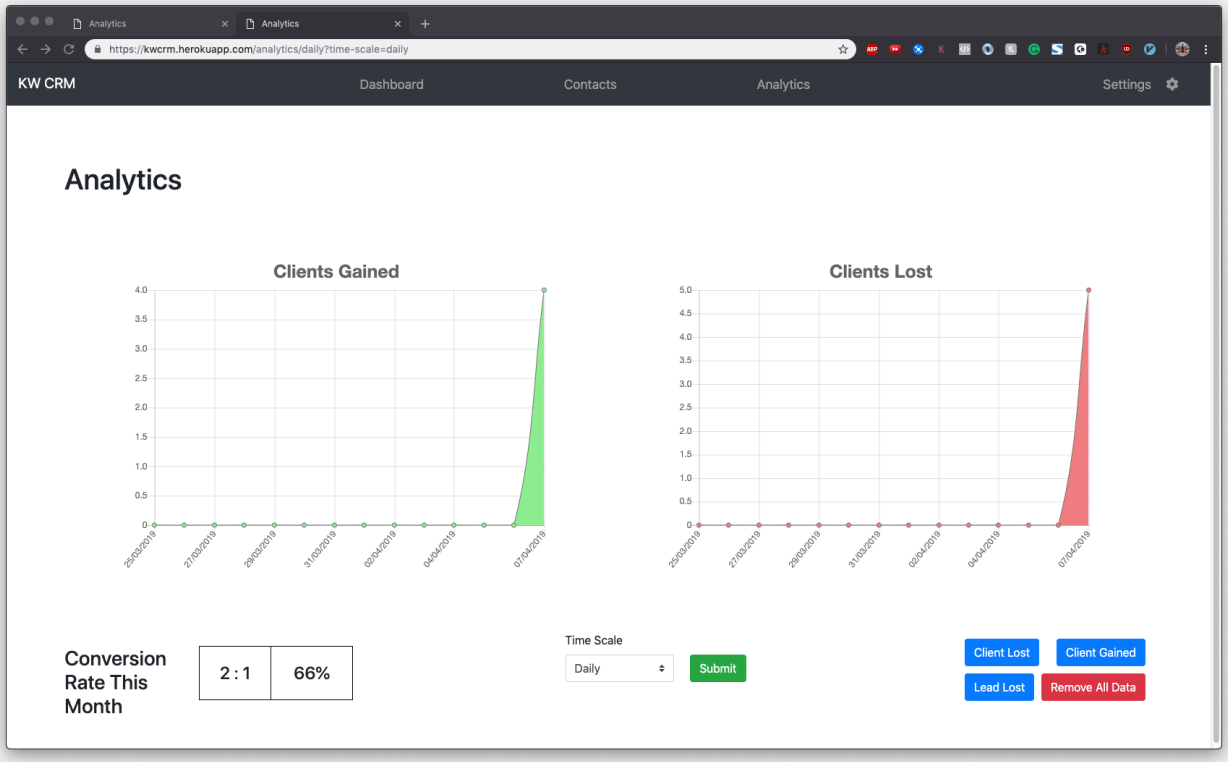

When a contact's data get's edited, this message gets displayed after they get redirected back to the contacts list page



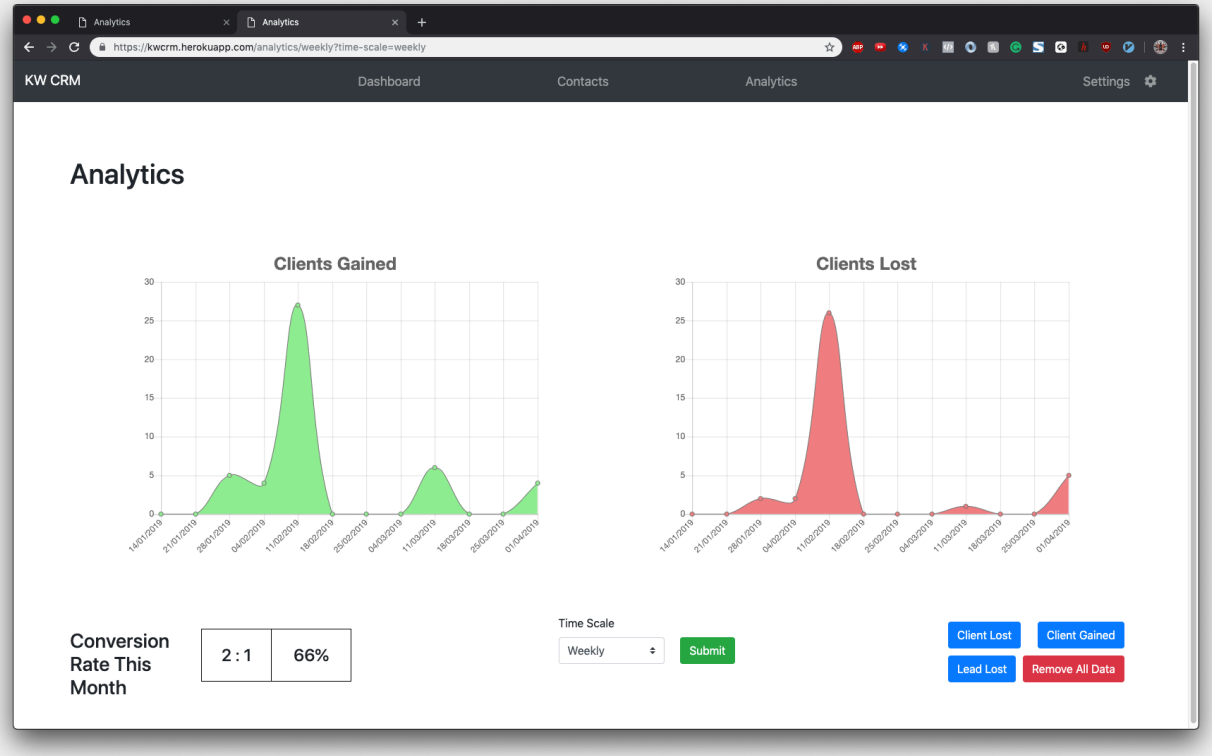

When a contact gets successfully created, this message gets displayed after they get redirected back to the contacts list page

**The Analytics Page**

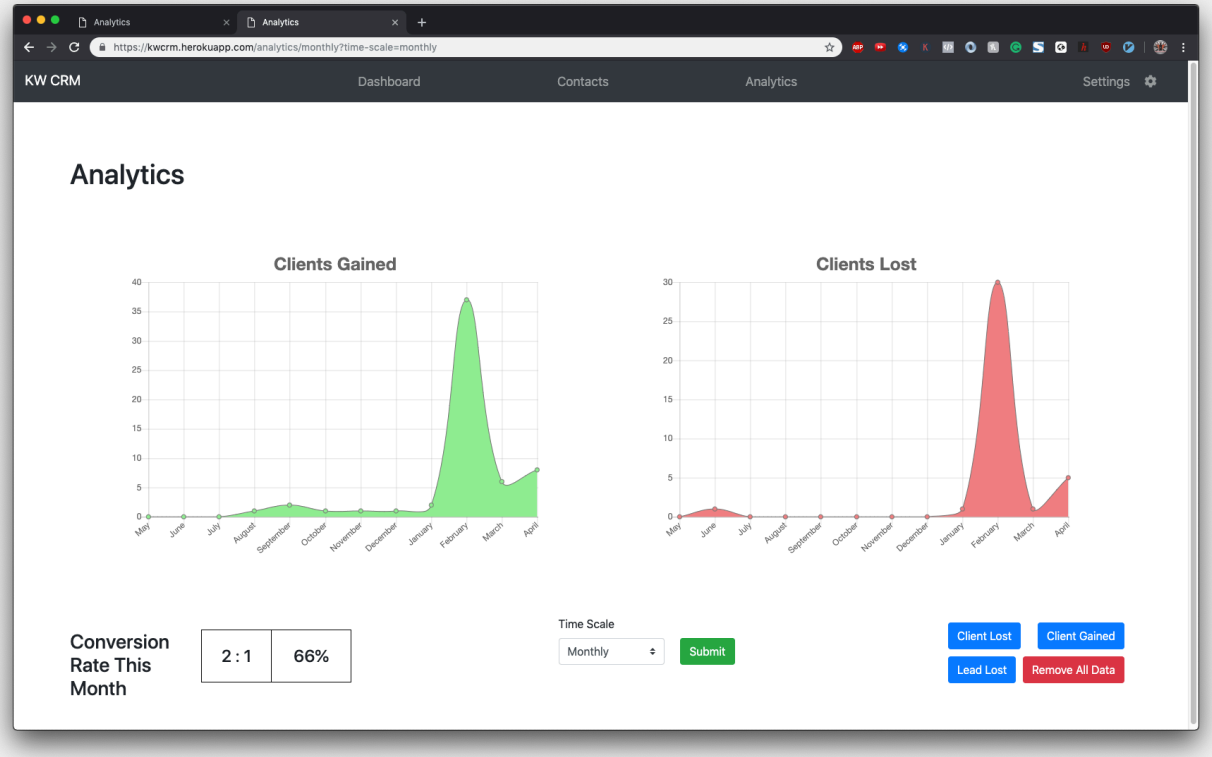
The purpose of this page is to see how the business is doing. It has the input buttons for when a lead is lost, a client is lost and for when a client is gained. When these buttons are pressed, their relevant data stores get a new record appended to the end of the file with the current date. The graphs then display the clients gained and clients lost over a period of time. The period of time that the graphs display data for is determined by the drop down menu that has options for daily (the past 14 days), weekly (the past 12 weeks), and monthly (the past 12 months). There is also the conversion rate for the the month which is the the clients gained against the leads lost in both a ratio that is in its lowest form and as a percentage.




this is the page showing the daily graphs



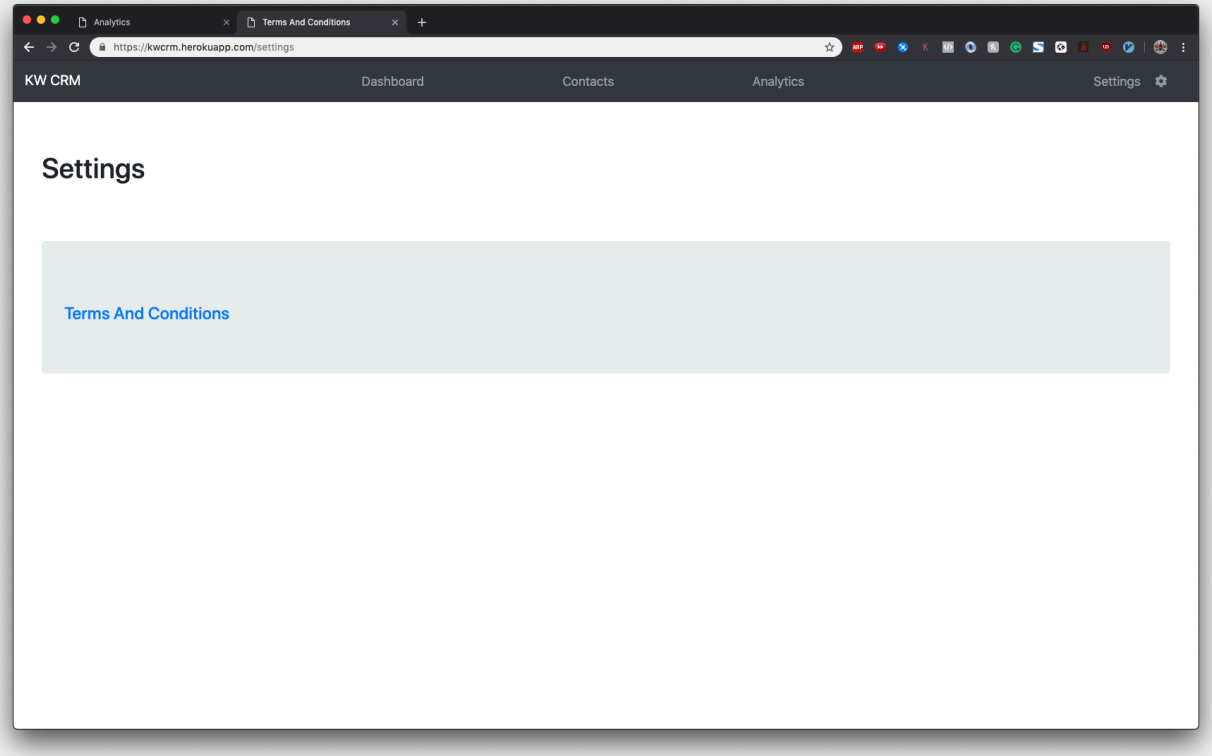

This is showing the weekly graphs



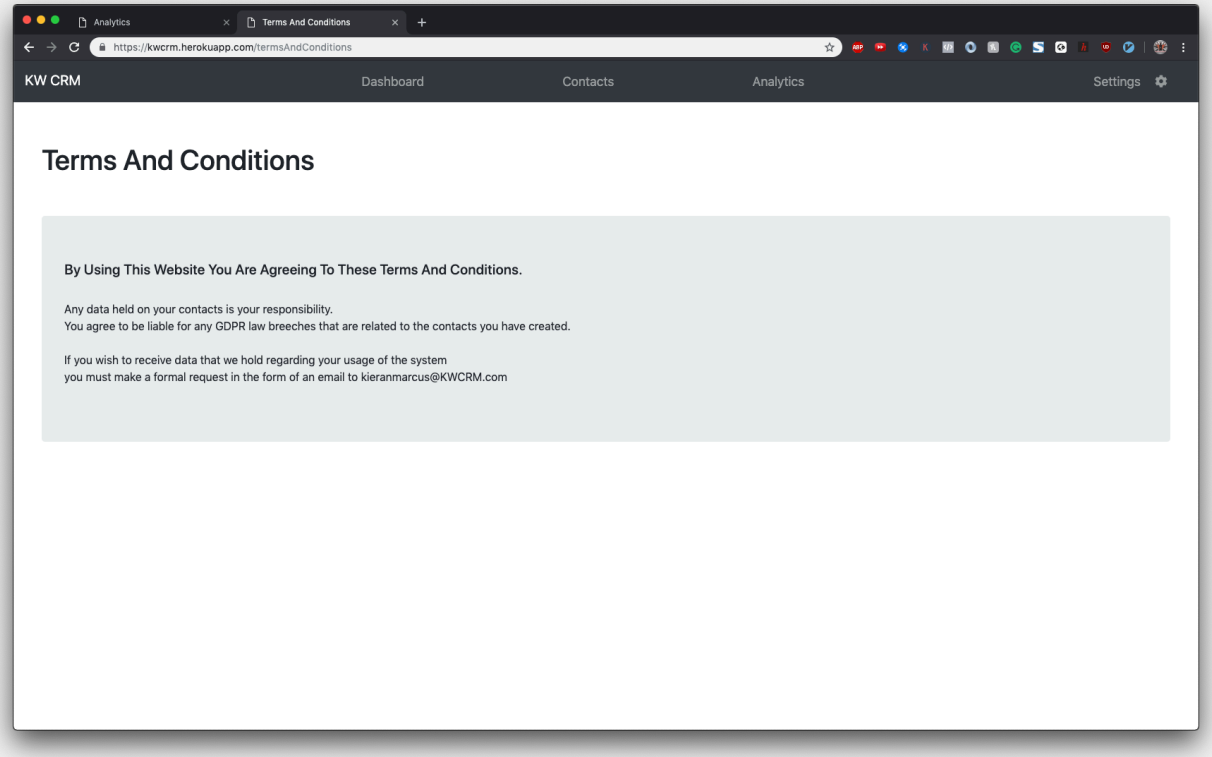

This is showing the monthly graphs

**The Settings Page**

The purpose of this page is to show any additional links and to alter any settings however there are no settings currently and so there is just one link to the terms and conditions page

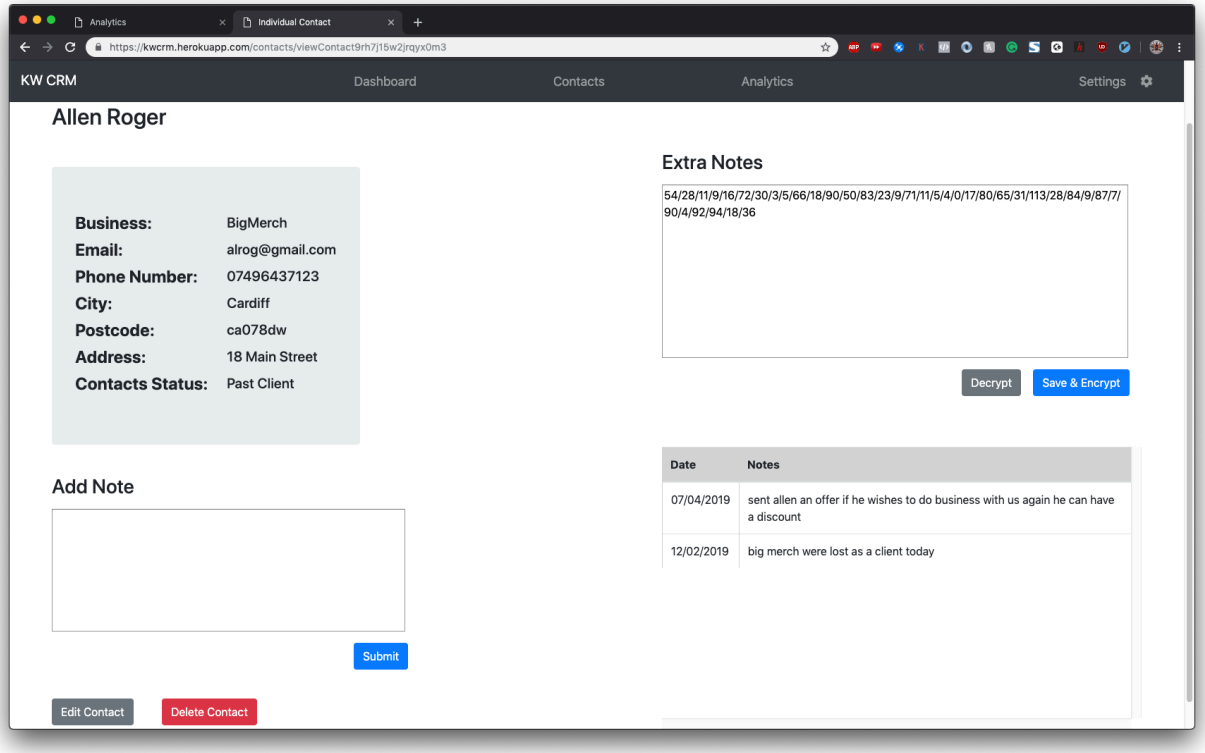



**The Terms And Conditions Page**




This page outlines the terms and conditions the user is agreeing to by using the system in order to be in line with the GDPR legislation

**The Individual Contact's Page**

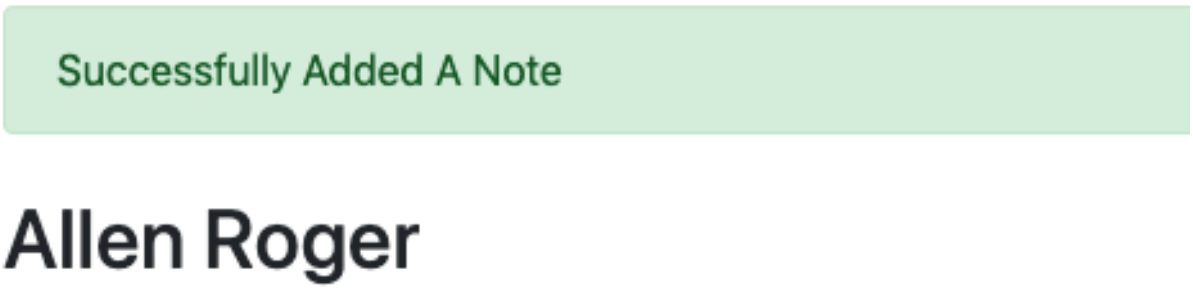



The purpose of this page is to show all the details held on the contact, to input and view all the notes that are related to that contact and to display/ input/ encrypt/ decrypt the extra notes section

When a note gets added the current date also gets added to the record and is displayed along with the note in the table

When there is no data present in the extra notes section there is place holder piece of text that describes how to use the section

You can also edit the contact that is currently selected by pressing the edit contact button that redirects to the edit contact page. Pressing the delete contact button will delete the contact that is currently selected and redirect the user to the contact's list page

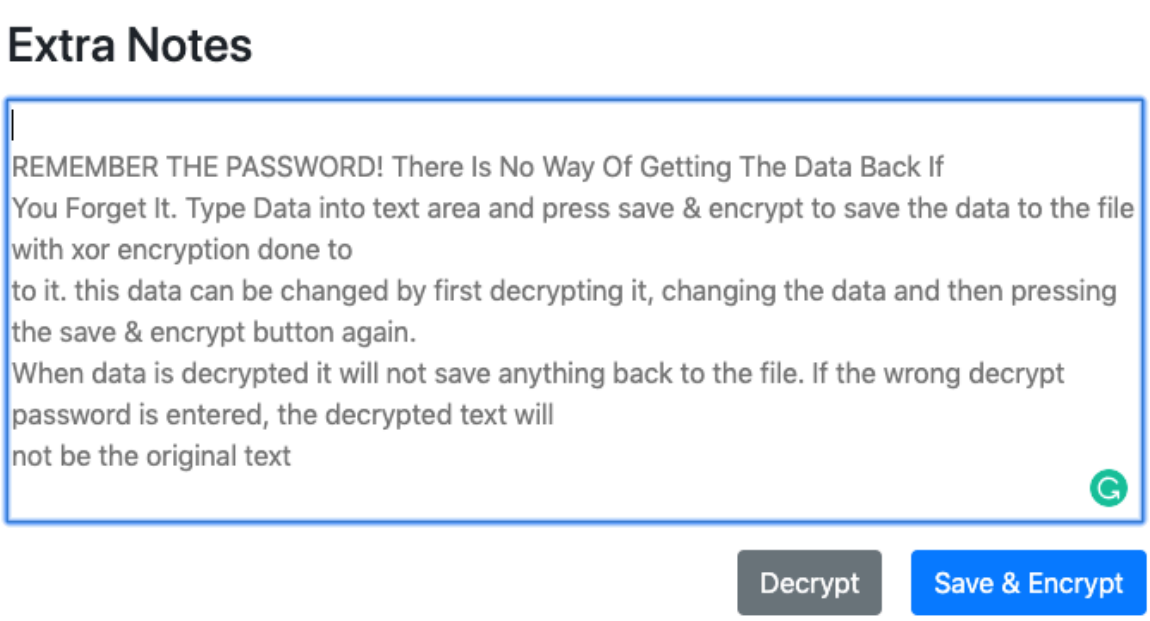



When a note is added this message gets displayed

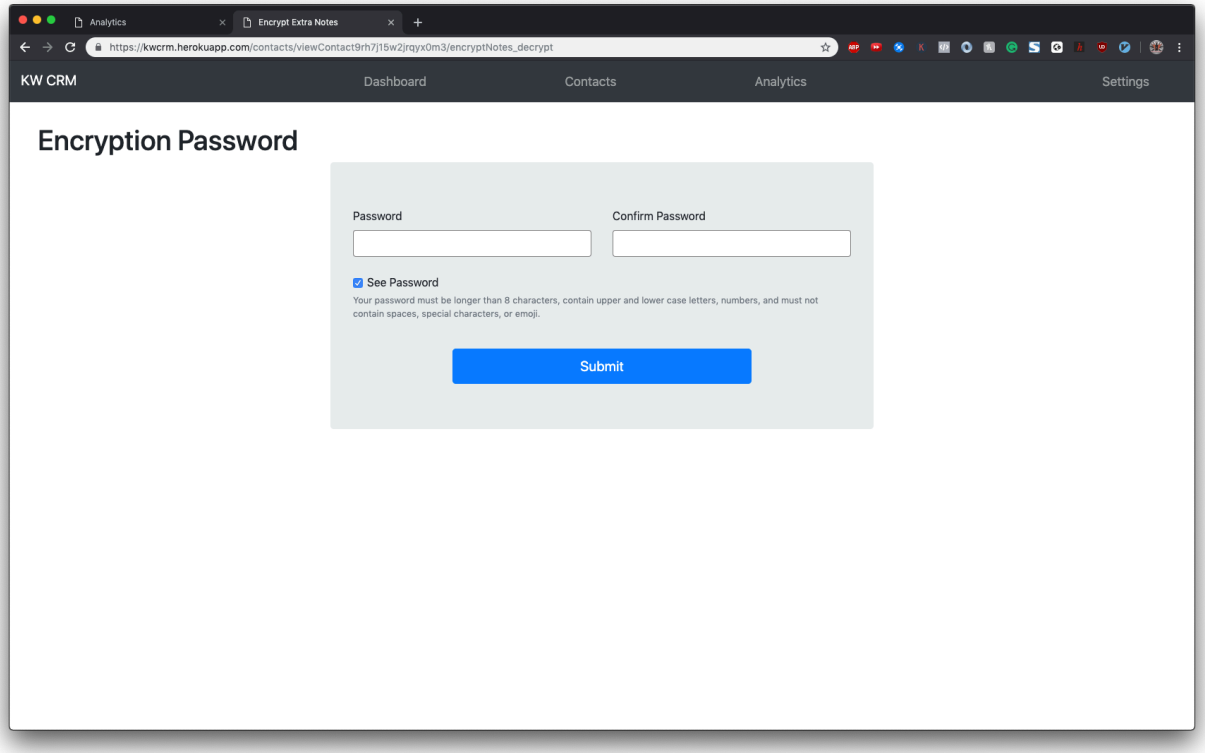
___ Please Ensure All Text Fields Have Data Entered ___ ___ Ensure That The Subject Field Is Within Range Of 1 - 250 Characters ___

## Allen Roger

These are all the possible validation messages there are for the notes input form

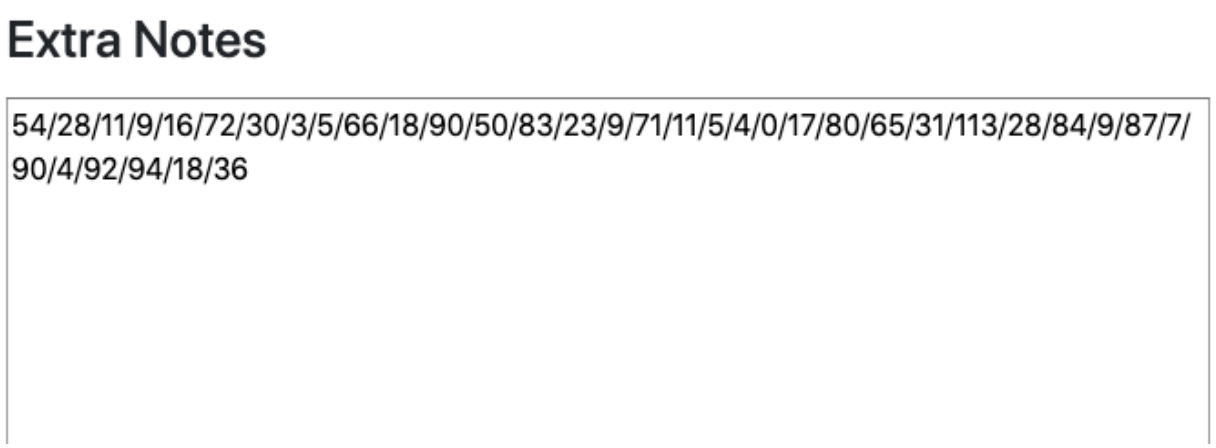



This is the place holder text that defines how to use the extra notes section to make it easier for the user to understand how to use it for the first time.

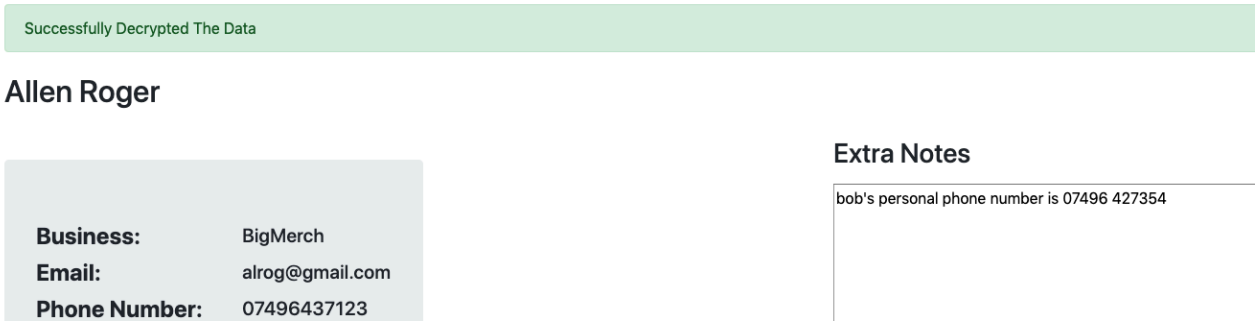


When the user types text into the extra notes section and goes to save it they get redirected to this page where they have to enter there password to encrypt or decrypt their data

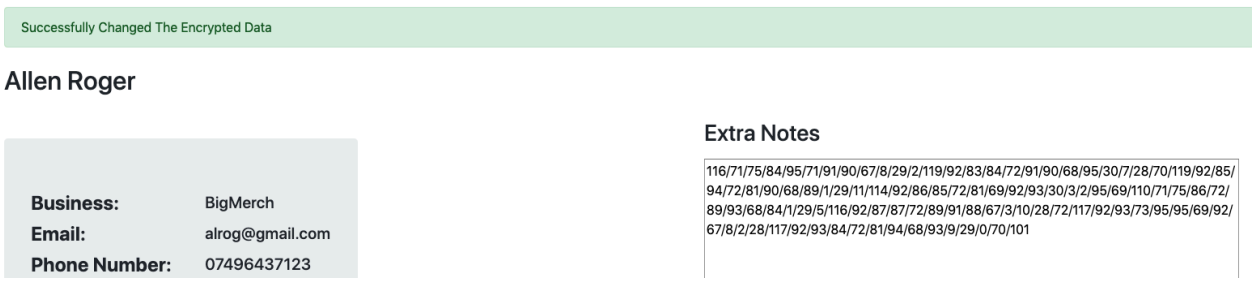

These are all the possible validation messages that can get displayed to the user



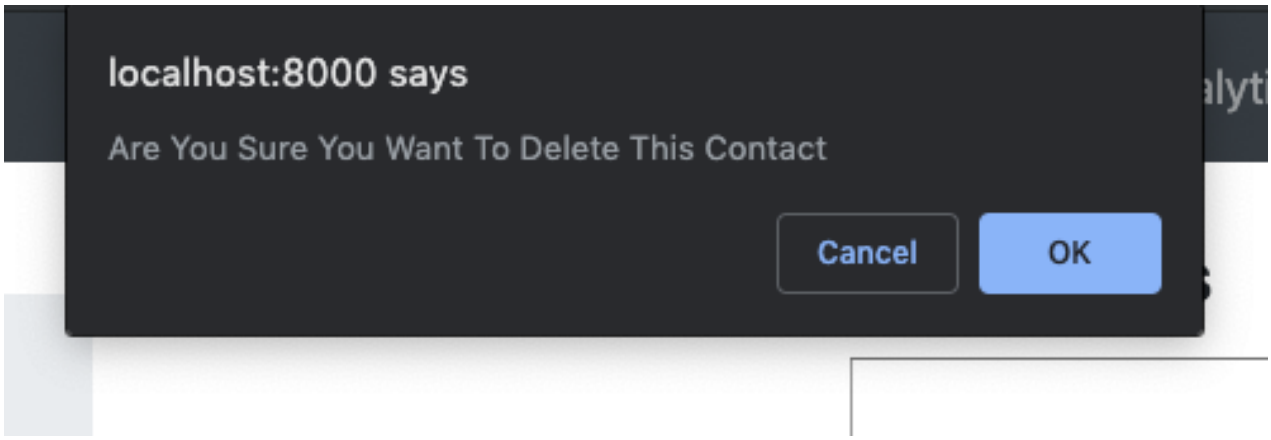

this is what the encrypted text will look like




When decrypted this is what the page will look like with a message that states the data has been decrypted and the decrypted data in the text area which will then go back to being encrypted if the page get's refreshed

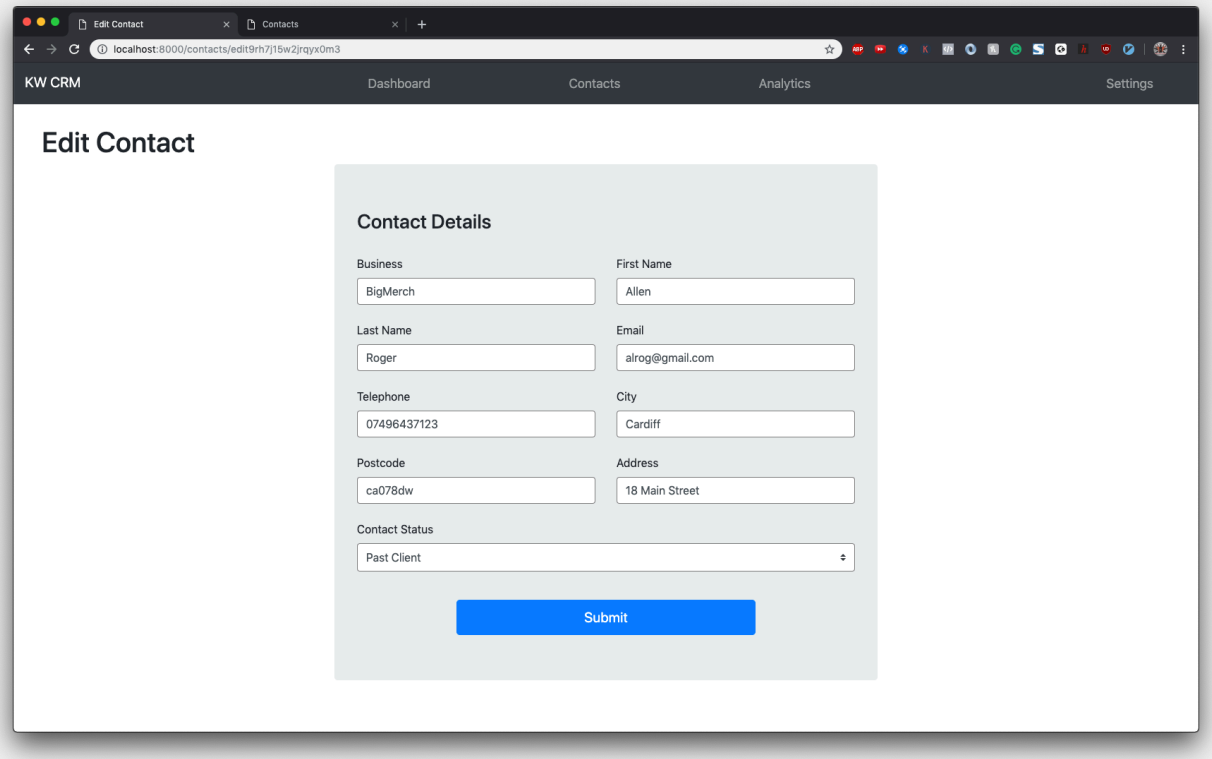



This is the message that gets displayed to the user when they encrypt the data in the extra notes section

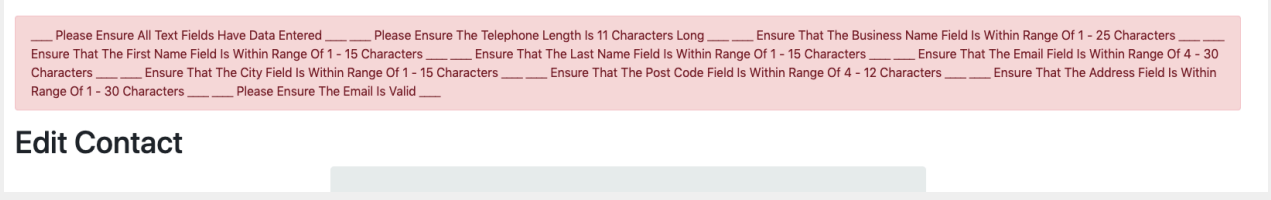



When the user goes to delete a contact, this confirmation dialog will get displayed to ensure that the user wants to delete the contact, upon pressing ok the contact will get deleted and the user will get redirected to the contacts list page, upon pressing cancel the delete request will get cancelled and the user will remain on the same page

**The Edit Contact Page**

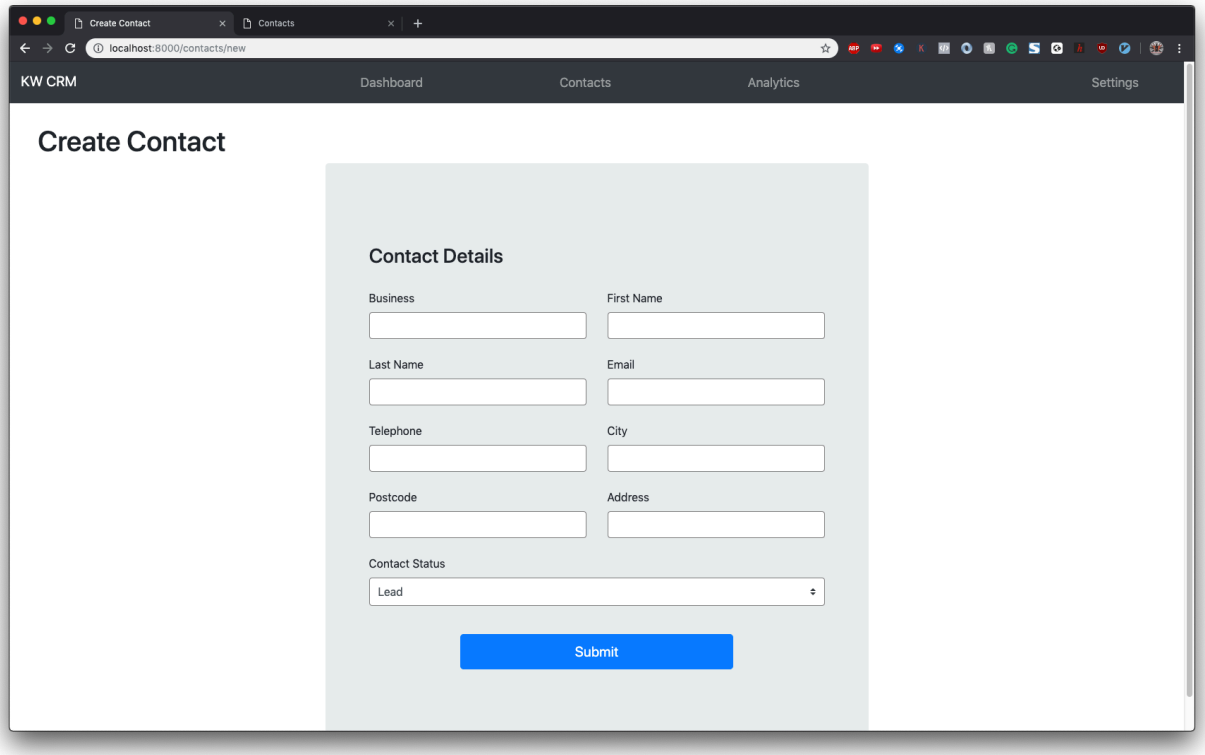



when the edit contact button on the individual contact page gets clicked, the user will get redirected to this page where they can edit the contacts data and then press submit when finished. The contact's data will get updated if the input is all valid and then the user will get r redirected to the contacts list page




These are all the possible validation messages that can get displayed due to invalid data entry in the edit contact form

**The Create Contact Page**




The purpose of this page is to allow the user to create a new contact and add it to the contacts file store

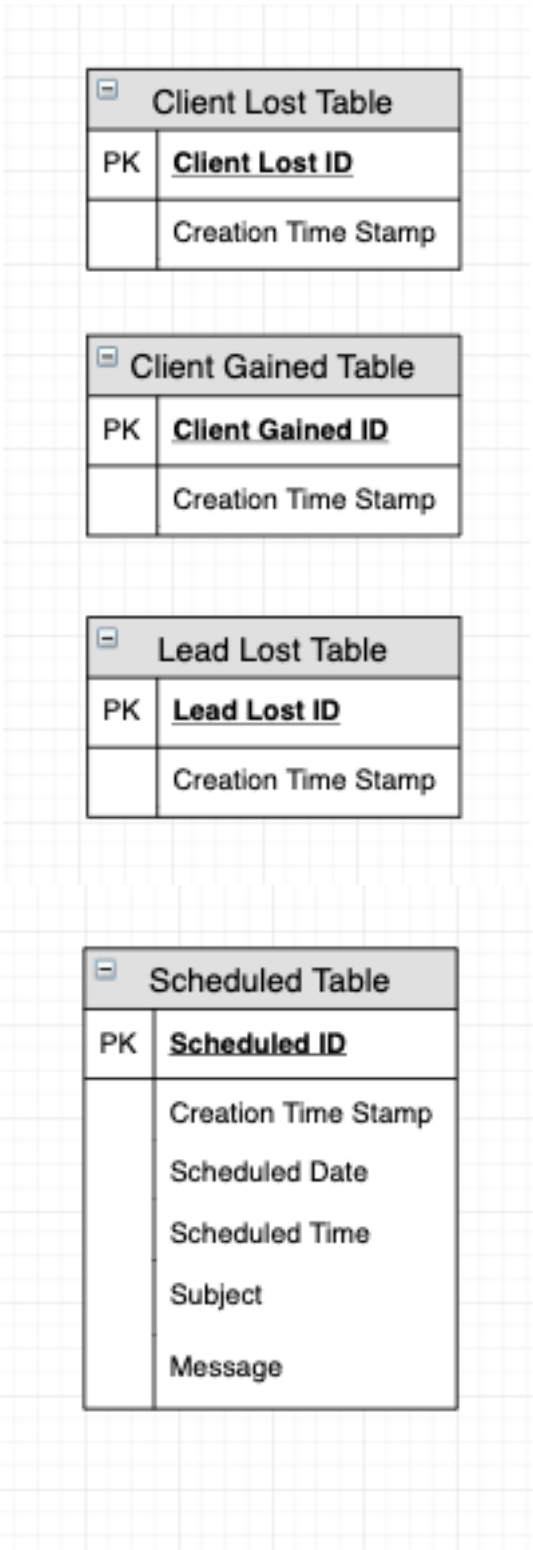
___ Please Ensure All Text Fields Have Data Entered ___ ___ Please Ensure The Telephone Length Is 11 Characters Long ___ ___ Ensure That The Business Name Field Is Within Range Of 1 - 25 Characters ___ ___ Ensure That The First Name Field Is Within Range Of 1 - 15 Characters ___ ___ Ensure That The Last Name Field Is Within Range Of 1 - 15 Characters ___ ___ Ensure That The Email Field Is Within Range Of 4 - 30 Characters ___ ___ Ensure That The City Field Is Within Range Of 1 - 15 Characters ___ ___ Ensure That The Post Code Field Is Within Range Of 4 - 12 Characters ___ ___ Ensure That The Address Field Is Within Range Of 1 - 30 Characters ___ ___ Please Ensure The Email Is Valid ___

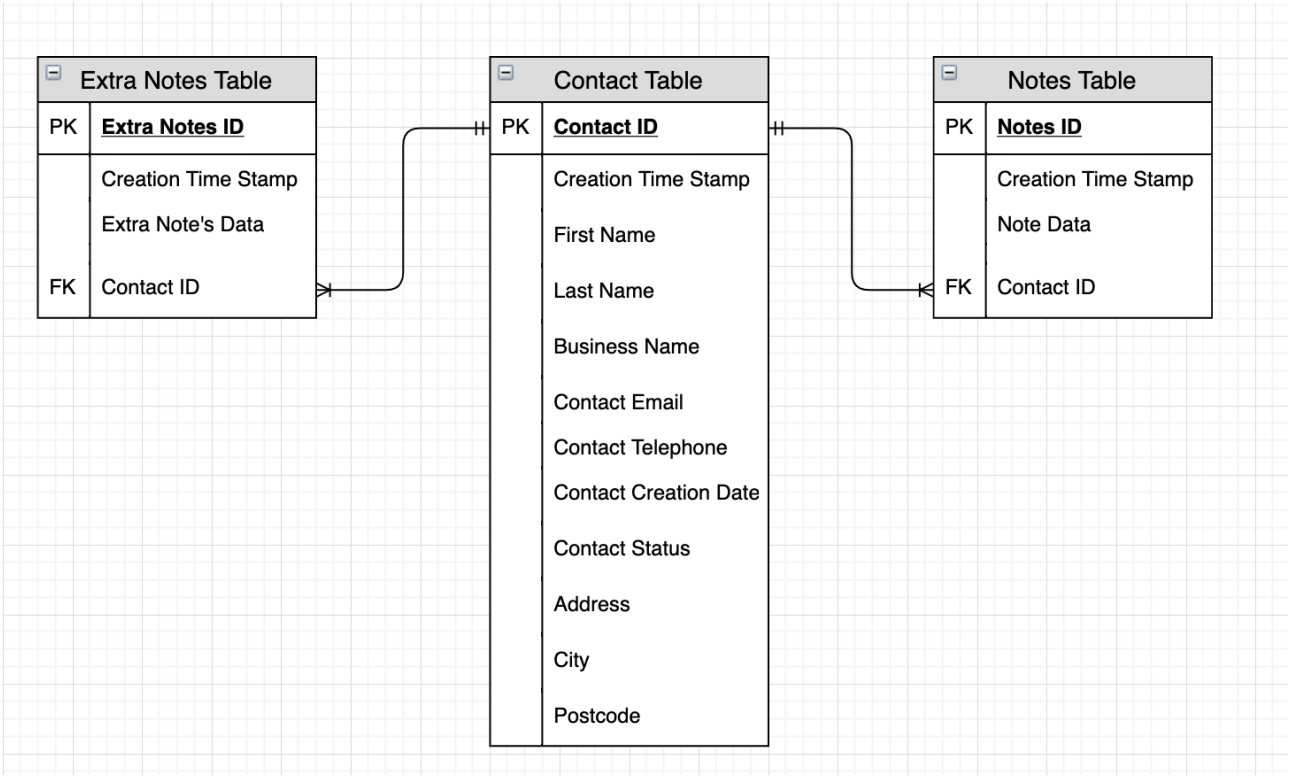
Create Contact

These are all the possible validation messages that can get displayed due to invalid data entry in the create contact form

# Data Storage

**Entity Relationship Diagrams**




Extra Notes Table
PK
Extra Notes ID
Creation Time Stamp
Extra Note's Data
FK
Contact ID
Contact Table
PK
Contact ID
Creation Time Stamp
First Name
Last Name
Business Name
Contact Email
Contact Telephone
Contact Creation Date
Contact Status
Address
City
Postcode
Notes Table
PK
Notes ID
Creation Time Stamp
Note Data
FK
Contact ID

## Data Dictionaries

*** = Primary key**

**// = Foreign Key**

## Contacts Table

| Name Of Field | Description Of Field | Type Of Field | Size Of Field | Example Data |
|---|---|---|---|---|
| **Contact's ID *** | The unique identifying field. Will be used to identify one contact from another | String | 10 | M123hjfdk343dfd |
| **Business Name** | Will store the contact's business name | String | 30 | Jimmy's Marketing |
| **First Name** | Will store the contact's first name | String | 30 | Jimmy |
| **Last Name** | Will store the contact's last name | String | 30 | Drool |
| **Email** | Will store the contact's email address. | String | 30 | kieran@kmwcrm.com |
| **Telephone** | Will store the contact's telephone number. | String | 11 | 01792947726 |
| **Contact Creation Date** | Will store the date that the contact was created. | ISO 8601 date and time format | 70 | 2014-09-08T08:02:17-05:00 |
| **Contact Status** | Will store the contact's status. | String | 30 | Recontact-lead |
| **Address** | Will store the contact's Address details. | String | 60 | 32 James Street, Pontardawe |
| **City** | Will store the contact's City. | String | 50 | Swansea |
| **Postcode** | Will store the contact's Postcode. | String | 15 | SA8 5AW |

## Extra Notes Table

| Name Of Field | Description Of Field | Type Of Field | Size Of Field | Example Data |
|---|---|---|---|---|
| **Extra Notes ID *** | The unique identifying field. Will be used to identify one record in the extra notes file from another | String | 10 | M123hjfdk343dfd |
| **Extra Notes** | Will store the contact's Extra notes can be in plain text or as an encrypted string. | String | 10,000 | 14/19/56/26/00/57/03/07/12/05/04 |
| **Extra Note Creation Date** | Will store the date of when that extra note was created. | ISO 8601 date and time format | 70 | 2014-09-08T08:02:17-05:00 |
| **Contact's ID //** | The unique identifying field. Will be used to identify a contact | String | 10 | M123hjfdk343dfd |

## Notes Table

| Name Of Field | Description Of Field | Type Of Field | Size Of Field | Example Data |
|---|---|---|---|---|
| **Notes ID *** | The unique identifying field for a given note. | String | 10 | M123hjfdk343dfd |
| **Note Creation Date** | Will store the date of when that the note was created. | ISO 8601 date and time format | 70 | 2014-09-08T08:02:17-05:00 |
| **Contact's ID //** | The unique identifying field. Will be used to identify a contact | String | 10 | M123hjfdk343dfd |

## Scheduled Tasks Table

| Name Of Field | Description Of Field | Type Of Field | Size Of Field | Example Data |
|---|---|---|---|---|
| **Scheduled ID *** | The unique identifying field. Will be used to identify one scheduling from another | String | 10 | M123hjfdk343dfd |
| **Scheduled Task Creation Date** | Will store the date of when that the scheduled task was created. | ISO 8601 date and time format | 70 | 2014-09-08T08:02:17-05:00 |
| **Scheduled Date** | Will hold the scheduling's date | String | 10 | 21/10/2019 |

| Name Of Field | Description Of Field | Type Of Field | Size Of Field | Example Data |
|---|---|---|---|---|
| **Scheduled Time** | Will hold the scheduling's time | String | 5 | 18:30 |
| **Subject** | Will hold the scheduling's Subject Line | String | 40 | Contact John to get decision |
| **Message** | Will hold the scheduling's message/ further details about the scheduling | String | 500 | Find out if he wants to go ahead with the services. He was apprehensive if he need it. Go with the slow sale, go over case studies again. |

## Client Lost Table

| Name Of Field | Description Of Field | Type Of Field | Size Of Field | Example Data |
|---|---|---|---|---|
| **Client Lost ID *** | The unique identifying field. Will be used to identify one client loss from another | Integer | 10 | 245522 |
| **Client Lost Date** | Will store the date of when that client was lost (the date when the client lost button got pressed) | ISO 8601 date and time format | 70 | 2014-09-08T08:02:17-05:00 |

## Client Gained Table

| Name Of Field | Description Of Field | Type Of Field | Size Of Field | Example Data |
|---|---|---|---|---|
| **Client Gained ID *** | The unique identifying field. Will be used to identify one client gain from another | Integer | 10 | 245522 |
| **Client Gained Date** | Will store the date of when that client was gained (the date when the client gained button got pressed) | ISO 8601 date and time format | 70 | 2014-09-08T08:02:17-05:00 |

## Lead Lost Table

| Name Of Field | Description Of Field | Type Of Field | Size Of Field | Example Data |
|---|---|---|---|---|
| **Lead Lost ID *** | The unique identifying field. Will be used to identify one lead loss from another | Integer | 10 | 245522 |
| **Lead Lost Date** | Will store the date of when that lead was lost (the date when the lead lost button got pressed) | ISO 8601 date and time format | 70 | 2014-09-08T08:02:17-05:00 |

## Populated Data Store Files

**Contacts Extra Notes File (.CSV)**

```
17gdv3216js262g0b,2019-02-12T19:35:47+00:00,mlwmc41eznjrxy3t15,41/22/8/10/8/72/30/3/9/67/87/19/54/28/22/10/3/27/18,
mlwmc41xpnjs0k9qlf,2019-02-11T00:00:00+00:00,9rh7j15w2jrqyx0m3,54/28/11/9/16/72/30/3/5/66/18/90/50/83/23/9/71/11/5/4/0/17/80/65/31/113/28/84/9/87/7/90/4/92/94/18/36,
mlwmc41y9ajs0m3o0b,2019-02-11T00:00:00+00:00,mlwmc4139ojrwkirmu,9/22/8/10/8/72/29/4/62/93/86/11,
mlwmc42118js0s8jqa,2019-02-11T20:20:51+00:00,9rh7j15xxjrqz103c,41/22/22/70/9/9/30/2/3/95/83/95/97/26/10/21/18/26/11/5/15/84/18/93/5/44/17/1/20/71/1/25/75/66/31/28/29/111/93/74/72/73/70/74/3/9/93/94/92/80/4/41/22/22/3/55,

```

**Contacts Notes (.CSV)**

```
17gdv3216js260h5v,2019-02-12T19:34:15+00:00,12/02/2019,mlwmc41eznjrxy3t15,he needed time to think over what we discussed call next week,
17gdv33s8js2aejqi,2019-02-12T21:37:10+00:00,12/02/2019,9rh7j15w2jrqyx0m3,big merch were lost as a client today,
```

**Clients Lost (.CSV)**

```
mlwmc4222wfs0dtn81,2019-02-02T00:00:00+00:00,2019-02,
mlwmc4222wfs0dtn82,2019-02-02T00:00:00+00:00,2019-02,
mlwmc4222wfs0xtn83,2019-02-04T00:00:00+00:00,2019-02,
mlwmc4222wfs0xan84,2019-02-09T00:00:00+00:00,2019-02,
slwmc4222wgs0xsyxb,2019-03-11T00:00:00+00:00,2019-03,
flwmc4222dgs0xan8u,2019-05-11T00:00:00+00:00,2019-05,
hlwmc4222wjs0xtd8u,2019-07-11T00:00:00+00:00,2019-07,
llwmc4222wddtyfgxb,2019-08-11T00:00:00+00:00,2019-08,
dlwmc4222sjdfstn8u,2019-08-11T00:00:00+00:00,2019-08,
qlwmc4222fjs0xtyfg,2019-09-11T00:00:00+00:00,2019-09,
elwmc4222wjggsdxxb,2019-10-11T00:00:00+00:00,2019-10,
rlwmc4222wjshxtn8u,2019-11-11T00:00:00+00:00,2019-11,
htlwmc4222gwjkdtxb,2019-12-11T00:00:00+00:00,2019-12,
ulwmc4222ssxtnfsgu,2019-01-11T00:00:00+00:00,2019-01,
olwmc4222wjgaxtyxb,2018-06-11T00:00:00+00:00,2018-06,
17gdv3l4fjs5ebpfy,2019-02-15T00:00:00+00:00,2019-02,
17gdv3lbujs5edify,2019-02-15T00:00:00+00:00,2019-02,
17gdv3lbujs5edkxs,2019-02-15T00:00:00+00:00,2019-02,
17gdv3lbujs5eelkd,2019-02-15T00:00:00+00:00,2019-02,
17gdv3llfjs5f25ss,2019-02-15T00:00:00+00:00,2019-02,
17gdv3llfjs5f275t,2019-02-15T00:00:00+00:00,2019-02,
17gdv31aojjs7x45n0,2019-02-16T00:00:00+00:00,2019-02,
17gdv31aojjs7x45rb,2019-02-16T00:00:00+00:00,2019-02,
17gdv31aojjs7x46aw,2019-02-16T00:00:00+00:00,2019-02,
17gdv31aojjs7x46ff,2019-02-16T00:00:00+00:00,2019-02,
17gdv31aojjs7x46ka,2019-02-16T00:00:00+00:00,2019-02,
17gdv31aojjs7x46os,2019-02-16T00:00:00+00:00,2019-02,
17gdv31aojjs7x46sy,2019-02-16T00:00:00+00:00,2019-02,
17gdv31aojjs7x46xf,2019-02-16T00:00:00+00:00,2019-02,
17gdv31aojjs7x4peg,2019-02-16T00:00:00+00:00,2019-02,
17gdv31aojjs7x4sgy,2019-02-16T00:00:00+00:00,2019-02,
17gdv31aojjs7x4slk,2019-02-16T00:00:00+00:00,2019-02,
17gdv31aojjs7x4t61,2019-02-16T00:00:00+00:00,2019-02,
17gdv31aojjs7x4tav,2019-02-16T00:00:00+00:00,2019-02,
17gdv31aojjs7x4tff,2019-02-16T00:00:00+00:00,2019-02,
17gdv31aojjs7x4tjl,2019-02-16T00:00:00+00:00,2019-02,
17gdv31aojjs7x4tok,2019-02-16T00:00:00+00:00,2019-02,
17gdv31aojjs7x4tt7,2019-02-16T00:00:00+00:00,2019-02,
17gdv31aojjs7x4txv,2019-02-16T00:00:00+00:00,2019-02,
17gdv31aojjs7x4u2h,2019-02-16T00:00:00+00:00,2019-02,
17gdv31aojjs7x4u79,2019-02-16T00:00:00+00:00,2019-02,
```

**Contacts (.CSV)**

```
9rh7j15w2jrqyx0m3,2019-02-04T00:00:00+00:00,BigMerch,Allen,Roger,alrog@gmail.com,07496437123,Cardiff,ca078dw,18 Main Street,past-client,
9rh7j15xxjrqz103c,2019-02-04T00:00:00+00:00,Dynasty,Sally,Hughes,dynastyClothing@gmail.com,07496467098,Neath,sa95nb,Abbey Road,dead-lead,
9rh7j16i2jrqzl25e,2019-02-04T00:00:00+00:00,Paulsmith Guitars,Paul,Smith,psGuitars@gmail.com,07436890212,Swansea,sa89op,78 Green Street,lead,
mlwmc41eznjrxy2mfj,2019-02-09T00:00:00+00:00,Joe Skate,Joe,Rob,joeyskater@gmail.com,01792782635,Port Talbot,sa94bw,32 Blah Blah Road,recontact-lead,
mlwmc41eznjrxy3t15,2019-02-09T00:00:00+00:00,Big Breakfasts 4 You,Jamie,Roberts,jamierobs@outlook.com,07496436543,London,Ln09eb,Queens Street,recontact-lead,
mlwmc41eznjrxy48n0,2019-02-09T00:00:00+00:00,James Media,James,Brinks,jamesbrinks@hotmail.com,01792946751,Swansea,sa14na,22 Abc Castle Road,recontact-lead,
mlwmc41eznjrxy6sm3,2019-02-09T00:00:00+00:00,The Network Guys,Bob,Wally,boobywallyIT@networkgys.com,01792876784,Swansea,sa14na,22 Abc Castle Road,lead,
mlwmc41z37js0ote8c,2019-02-11T00:00:00+00:00,Breakfasts U Want,Rob,Sally,robsysally@breakfasts.com,07496436543,Swansea,Sa19op,Kingsway,past-client,
mlwmc4a6ojrt8dqpg,2019-02-06T00:00:00+00:00,Dig Dig Diggers,Larry,Watson,lazzywatz@gmail.com,07456897564,Port Talbot,sa131dp,Station Road,current-client,

```

**Leads Lost (.CSV)**

```
mlwmc4222wfs0xtn81,2019-02-02T00:00:00+00:00,2019-02,
mlwmc4222wfs0xtn82,2019-02-02T00:00:00+00:00,2019-02,
mlwmc4222was0xtn82,2019-02-05T00:00:00+00:00,2019-02,
mlwmc4222wfs0xtn83,2019-02-04T00:00:00+00:00,2019-02,
mlwmc4222wfs0xtn84,2019-02-09T00:00:00+00:00,2019-02,
mlwmc4222wfs0xtn85,2019-02-11T00:00:00+00:00,2019-02,
mlwmc4222afs0xtn85,2019-02-11T00:00:00+00:00,2019-02,
slwmc4222wgs0xtyxb,2019-03-11T00:00:00+00:00,2019-03,
flwmc4222dgs0xtn8u,2019-05-11T00:00:00+00:00,2019-05,
glwmc4222whg0xtyxb,2019-05-11T00:00:00+00:00,2019-05,
hlwmc4222wjs0xtn8u,2019-07-11T00:00:00+00:00,2019-07,
jlwmc4222wls0xtyxb,2019-08-11T00:00:00+00:00,2019-08,
klwmc4222dfs0xtn8u,2019-08-11T00:00:00+00:00,2019-08,
llwmc4222wdg0xtyxb,2019-08-11T00:00:00+00:00,2019-08,
dlwmc4222sjs0xtn8u,2019-08-11T00:00:00+00:00,2019-08,
qlwmc4222fjs0xtyxb,2019-09-11T00:00:00+00:00,2019-09,
wlwmc4222dfs0xtn8u,2019-09-11T00:00:00+00:00,2019-09,
elwmc4222wjggxtyxb,2019-10-11T00:00:00+00:00,2019-10,
rlwmc4222wjshxtn8u,2019-11-11T00:00:00+00:00,2019-11,
htlwmc4222wjk0xtxb,2019-12-11T00:00:00+00:00,2019-12,
ulwmc4222djssxtn8u,2019-01-11T00:00:00+00:00,2019-01,
olwmc4222wjgaxtyxb,2019-01-11T00:00:00+00:00,2019-01,
17gdv3lbujs5edlz9,2019-02-15T00:00:00+00:00,2019-02,
17gdv3lbujs5edmvy,2019-02-15T00:00:00+00:00,2019-02,
17gdv3lbujs5edo4k,2019-02-15T00:00:00+00:00,2019-02,
17gdv3lbujs5edp5b,2019-02-15T00:00:00+00:00,2019-02,
17gdv3lbujs5edqgz,2019-02-15T00:00:00+00:00,2019-02,
17gdv3lbujs5edr4o,2019-02-15T00:00:00+00:00,2019-02,
17gdv3lbujs5edrhz,2019-02-15T00:00:00+00:00,2019-02,
17gdv3lbujs5edrk2,2019-02-15T00:00:00+00:00,2019-02,
17gdv3lbujs5edrsu,2019-02-15T00:00:00+00:00,2019-02,
17gdv3lbujs5edrxd,2019-02-15T00:00:00+00:00,2019-02,
17gdv3lbujs5eds7z,2019-02-15T00:00:00+00:00,2019-02,
17gdv3lbujs5edsl5,2019-02-15T00:00:00+00:00,2019-02,
17gdv3lbujs5edtia,2019-02-15T00:00:00+00:00,2019-02,
17gdv3lbujs5educ3,2019-02-15T00:00:00+00:00,2019-02,
17gdv3lbujs5edv96,2019-02-15T00:00:00+00:00,2019-02,
17gdv3llfjs5f1cas,2019-02-15T00:00:00+00:00,2019-02,
17gdv3llfjs5f1m0u,2019-02-15T00:00:00+00:00,2019-02,
17gdv3llfjs5f287o,2019-02-15T00:00:00+00:00,2019-02,
17gdv3llfjs5f29wj,2019-02-15T00:00:00+00:00,2019-02,
17gdv3llfjs5f2awb,2019-02-15T00:00:00+00:00,2019-02,
17gdv3llfjs5f2c1c,2019-02-15T00:00:00+00:00,2019-02,
17gdv3llfjs5f2d4o,2019-02-15T00:00:00+00:00,2019-02,
17gdv3marjs62zbkq,2019-02-15T00:00:00+00:00,2019-02,
```

**Scheduled (.CSV)**

```
1  mlwmc41hs6jry3i2za,2019-02-09T00:00:00+00:00,2019-02-29T00:00:00+00:00,29/02/2019,00:0
   0,Call Sally,hell yeah,
2
```

**Clients Gained (.CSV)**

```
mlwmc4222wfs0xtn81,2019-00-02T00:00:00+00:00,2019-02,
mlwmc4222wfs0xtn82,2019-02-02T00:00:00+00:00,2019-02,
mlwmc4222was0xtn82,2019-02-05T00:00:00+00:00,2019-02,
mlwmc4222wfs0xtn83,2019-02-04T00:00:00+00:00,2019-02,
mlwmc4222wfs0xtn84,2019-02-09T00:00:00+00:00,2019-02,
mlwmc4222wfs0xtn85,2019-02-11T00:00:00+00:00,2019-02,
mlwmc4222afs0xtn85,2019-02-11T00:00:00+00:00,2019-02,
slwmc4222wgs0xtyxb,2019-03-11T00:00:00+00:00,2019-03,
flwmc4222dgs0xtn8u,2019-05-11T00:00:00+00:00,2019-05,
glwmc4222whg0xtyxb,2019-05-11T00:00:00+00:00,2019-05,
hlwmc4222wjs0xtn8u,2019-06-11T00:00:00+00:00,2019-06,
jlwmc4222wls0xtyxb,2019-04-11T00:00:00+00:00,2019-04,
klwmc4222dfs0xtn8u,2019-04-11T00:00:00+00:00,2019-04,
llwmc4222wdg0xtyxb,2019-04-11T00:00:00+00:00,2019-04,
dlwmc4222sjs0xtn8u,2019-04-11T00:00:00+00:00,2019-04,
qlwmc4222fjs0xtyxb,2019-03-11T00:00:00+00:00,2019-03,
wlwmc4222dfs0xtn8u,2019-03-11T00:00:00+00:00,2019-03,
elwmc4222wjggxtyxb,2019-03-11T00:00:00+00:00,2019-03,
rlwmc4222wjshxtn8u,2019-03-11T00:00:00+00:00,2019-03,
htlwmc4222wjk0xtxb,2019-03-11T00:00:00+00:00,2019-03,
ulwmc4222djssxtn8u,2019-01-11T00:00:00+00:00,2019-01,
olwmc4222wjgaxtyxb,2019-01-11T00:00:00+00:00,2019-01,
17gdv3l4fjs5eb2hu,2019-02-15T00:00:00+00:00,2019-02,
17gdv3l4fjs5ed3bp,2019-02-15T00:00:00+00:00,2019-02,
17gdv3lbujs5edcue,2019-02-15T00:00:00+00:00,2019-02,
17gdv3lbujs5ede7z,2019-02-15T00:00:00+00:00,2019-02,
17gdv3lbujs5edfuf,2019-02-15T00:00:00+00:00,2019-02,
17gdv3lbujs5eeqyh,2019-02-13T00:00:00+00:00,2019-02,
17gdv3lbujs5eervx,2019-02-11T00:00:00+00:00,2019-02,
17gdv3llfjs5f118r,2019-02-09T00:00:00+00:00,2019-02,
17gdv3llfjs5f19ew,2019-02-03T00:00:00+00:00,2019-02,
17gdv3llfjs5f19ew,2019-02-03T00:00:00+00:00,2019-02,
17gdv3llfjs5f19ew,2019-02-03T00:00:00+00:00,2019-02,
17gdv3llfjs5f19ew,2019-02-03T00:00:00+00:00,2019-02,
17gdv3llfjs5f1skm,2019-02-15T00:00:00+00:00,2019-02,
17gdv3llfjs5f1ve1,2019-02-15T00:00:00+00:00,2019-02,
17gdv3llfjs5f1wiz,2019-02-15T00:00:00+00:00,2019-02,
17gdv3llfjs5f1xxr,2019-02-15T00:00:00+00:00,2019-02,
17gdv3llfjs5f1z61,2019-02-15T00:00:00+00:00,2019-02,
17gdv3marjs62z8fx,2019-02-15T00:00:00+00:00,2019-02,
wlwmc4222dfs0xtn8u,2018-09-11T00:00:00+00:00,2018-09,
elwmc4222wjggxtyxb,2018-10-11T00:00:00+00:00,2018-10,
rlwmc4222wjshxtn8u,2018-11-11T00:00:00+00:00,2018-11,
htlwmc4222wjk0xtxb,2018-12-11T00:00:00+00:00,2018-12,
```

1

1

# Validation Methods By Type Of Validation

| Validation Type | Validation Code Snippet |
| --- | --- |
| **Look Up Check** | |

```
// preforms a look up check on the string comparing every character in the argument string
// to an array of acceptable characters. true is returned if there is a match found in the string to the list of
// acceptable characters, false returned if there is no match.
Validation[_stringLookupCheckForLowerCase] = (stringValue) => {
    let lowerCaseFlag = false;
    // all possible lower case letters put into an array of characters
    let lowerCaseArray = "abcdefghijklmnopqrstuvwxyz".split("");
    // The stringValue argument gets split up into an array of characters
    let stringArray = stringValue.split("");
    let currentCharacter = '';

        // check for a lower case character in the stringValue
        // by iterating through the stringArray and checking to see if a value matches any value in the lowerCaseArray
        // this is achieved through using a NESTED FOR LOOP
        for (let i = 0; i < stringArray.length; i++) {
            currentCharacter = stringArray[i];
            for (let j = 0; j < lowerCaseArray.length; j++) {
                // if a value in the stringArray matches the value in the lowerCaseArray, set the lowerCaseFlag to true
                if ( currentCharacter === lowerCaseArray[j] ) {
                    lowerCaseFlag = true;
                }
            }
        }
        // if the lowerCaseFlag is still set to false after the loop, then a lowerCase character doesn't exist, false is returned
    return lowerCaseFlag;
};
```

```

// preforms a look up check on the string comparing every character in the argument string
// to an array of acceptable characters. true is returned if there is a match found in the string to the list of
// acceptable characters, false returned if there is no match.
Validation[_stringLookupCheckForUpperCase] = (stringValue) => {
    let upperCaseFlag = false;
    // all possible upper case letters put into an array of characters
    let upperCaseArray = "ABCDEFGHIJKLMNOPQRSTUVWXYZ".split("");
    // The stringValue argument gets split up into an array of characters
    let stringArray = stringValue.split("");
    let currentCharacter = '';

    // check for a upper case character in the stringValue
    // by iterating through the stringArray and checking to see if a value matches any value in the upperCaseArray
    // this is achieved through using a NESTED FOR LOOP
    for (let i = 0; i < stringArray.length; i++) {
        currentCharacter = stringArray[i];
        for (let j = 0; j < upperCaseArray.length; j++) {
            // if a value in the stringArray matches the value in the upperCaseArray, set the upperCaseFlag to true
            if ( currentCharacter === upperCaseArray[j] ) {
                upperCaseFlag = true;
            }
        }
    }
    // if the upperCaseFlag is still set to false after the loop, then an upperCase character doesn't exist, false is returned
    return upperCaseFlag;
};

```

| Validation Type | Validation Code Snippet |
| --- | --- |
| | |

```
// preforms a look up check on the string comparing every character in the argument string
// to an array of acceptable characters. true is returned if there is a match found in the string to the list of
// acceptable characters, false returned if there is no match.
Validation[_stringLookupCheckForNumbers] = (stringValue) => {
    let numbersFlag = false;
    // all possible number characters  put into an array of characters
    let numbersArray = "1234567890".split("");
    // The stringValue argument gets split up into an array of characters
    let stringArray = stringValue.split("");
    let currentCharacter = '';

    // check for a number character in the stringValue
    // by iterating through the stringArray and checking to see if a value matches any value in the numbersArray
    // this is achieved through using a NESTED FOR LOOP
    for (let i = 0; i < stringArray.length; i++) {
        currentCharacter = stringArray[i];
        for (let j = 0; j < numbersArray.length; j++) {
            // if a value in the stringArray matches the value in the numbersArray, set the numbersFlag to true
            if ( currentCharacter === numbersArray[j] ) {
                numbersFlag = true;
            }
        }
    }
    // if the numbersFlag is still set to false after the loop, then a number character doesn't exist, false is returned
    return numbersFlag;
};

```

```
// preforms a look up check on the string comparing every character in the argument string
// to a space character. true is returned if there is a match found in the string, false returned if there is no match.
Validation[_stringLookupCheckForSpaces] = (stringValue) => {
    let spaceFlag = false;
    // the comparison space character
    let spaceCharacter = " ";
    // The stringValue argument gets split up into an array of characters
    let stringArray = stringValue.split("");
    let currentCharacter = '';

    // check for a space character in the stringValue
    // by iterating through the stringArray and checking to see if a value matches any value in the spaceCharacter
    for (let i = 0; i < stringArray.length; i++) {
        currentCharacter = stringArray[i];
        if (currentCharacter === spaceCharacter) {
            spaceFlag = true;
        }
    }
    // if the spaceFlag is still set to false after the loop, then a space character doesn't exist, false is returned
    return spaceFlag;
};

```

| Validation Type | Validation Code Snippet |
| --- | --- |
| | |

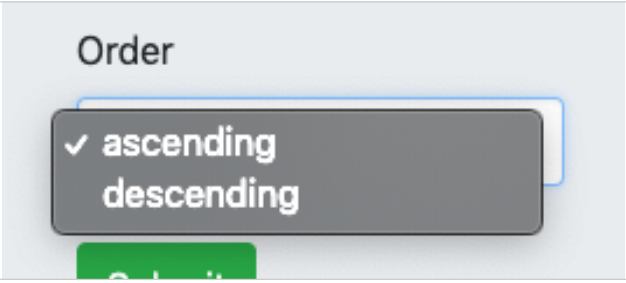
```
510
511 // preforms a look up check on the string comparing every character in the argument string
512 // to a comma character. true is returned if there is a match found in the string, false returned if there is no match.
513 Validation.stringLookupCheckForCommas = (stringValue) => {
514     let commaFlag = false;
515     // the comparison comma character
516     let commaCharacter = ",";
517     // The stringValue argument gets split up into an array of characters
518     let stringArray = stringValue.split("");
519     let currentCharacter = '';
520
521     // check for a number character in the stringValue
522     // by iterating through the stringArray and checking to see if a value matches any value in the commaCharacter
523     for (let i = 0; i < stringArray.length; i++) {
524         currentCharacter = stringArray[i];
525         if (currentCharacter === commaCharacter) {
526             commaFlag = true;
527         }
528     }
529     // if the commaFlag is still set to false after the loop, then a comma character doesn't exist, false is returned
530     return commaFlag;
531 };
532
```

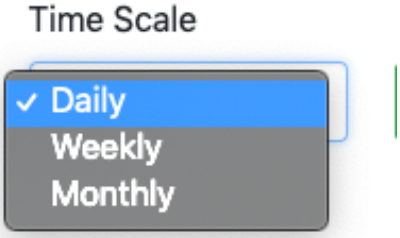
| Validation Type | Validation Code Snippet |
| --- | --- |
| | |

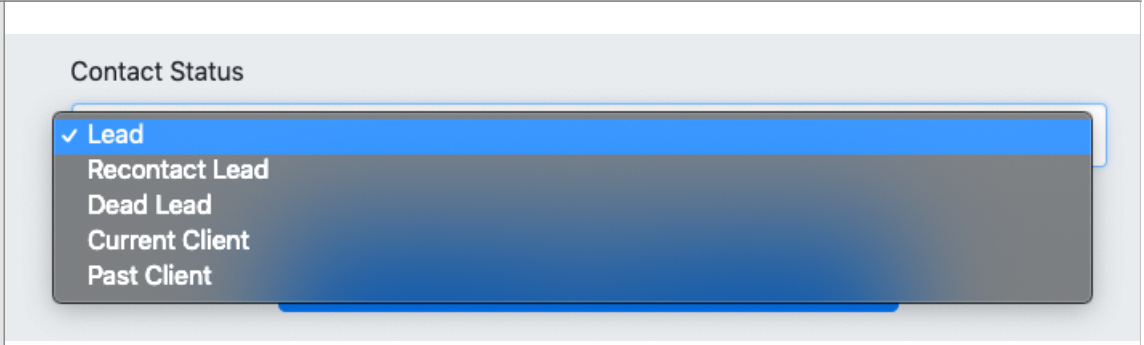
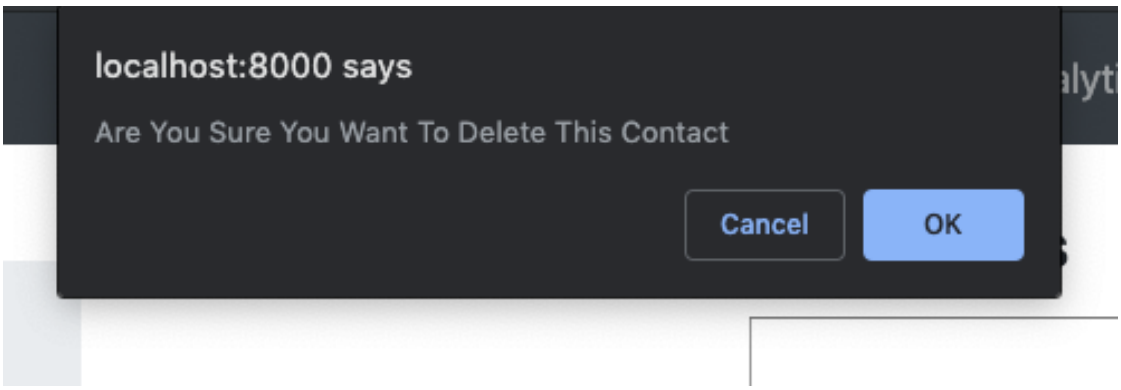
```
Validation.stringLookupCheck = (stringValue, objectOfWhatNeedsToBeInTheString) => {

    // if there is no object parameter an error message is thrown to
    // notify that an object parameter is needed
    if (objectOfWhatNeedsToBeInTheString === undefined) {
        throw "ERROR NEED TO HAVE AN OBJECT PARAMETER: "
        + "as the second parameter, set options of:\nlowerCase, upperCase, numbers: " +
        " to true \nif you want to check the string contains one of them types of characters ";
    }



    // checks to see if there is a property in the object passed as argument into the method
    // called lowerCase and checks to see if it's value is set to the boolean value true
    if (objectOfWhatNeedsToBeInTheString.lowerCase === true) {
        // preforms lookup check to see if there is any lowerCase characters in the string
        // if there is not then false gets returned
        if(!Validation[_stringLookupCheckForLowerCase](stringValue)){ return false }
    }

    // checks to see if there is a property in the object passed as argument into the method
    // called upperCase and checks to see if it's value is set to the boolean value true
    if (objectOfWhatNeedsToBeInTheString.upperCase === true) {
        // preforms lookup check to see if there is any upperCase characters in the string
        // if there is not then false gets returned
        if(!Validation[_stringLookupCheckForUpperCase](stringValue)){ return false }
    }

    // checks to see if there is a property in the object passed as argument into the method
    // called numbers and checks to see if it's value is set to the boolean value true
    if (objectOfWhatNeedsToBeInTheString.numbers === true) {
        // preforms lookup check to see if there is any number characters in the string
        // if there is not then false gets returned
        if(!Validation[_stringLookupCheckForNumbers](stringValue)){ return false }
    }

    // if false hasn't been returned yet then the string must contain all criteria
    return true;

};
```

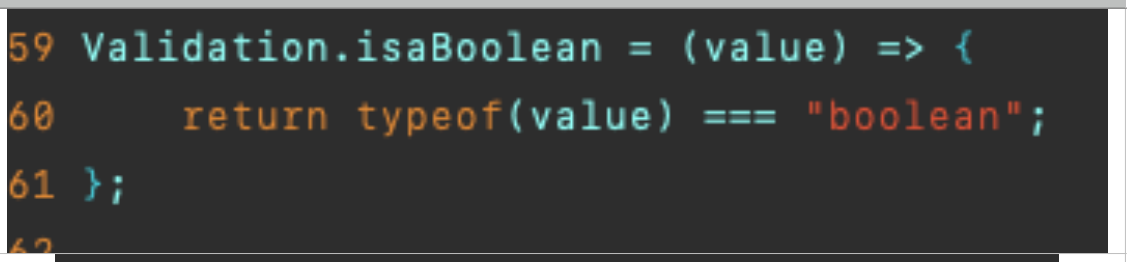
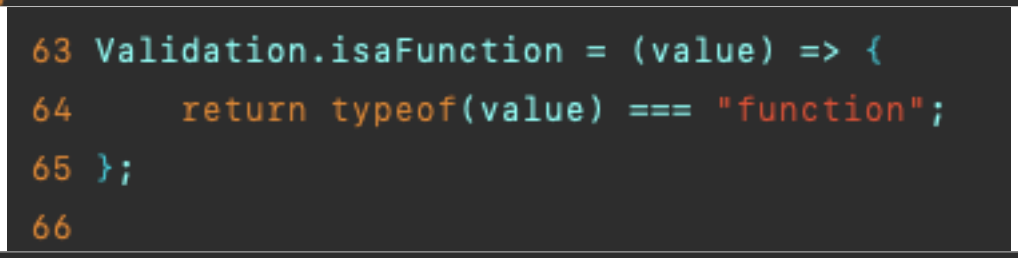
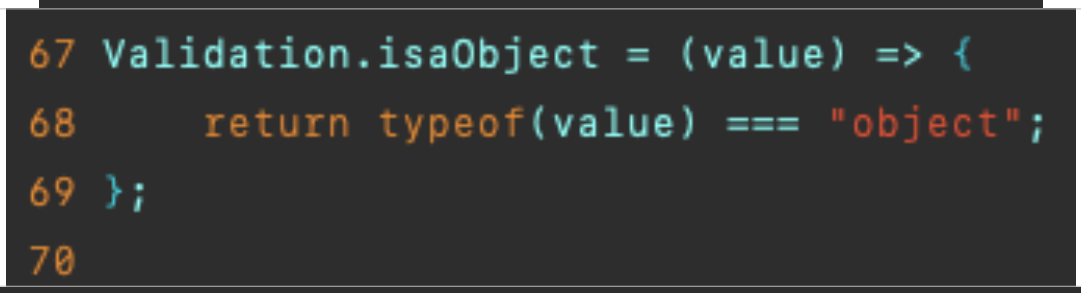
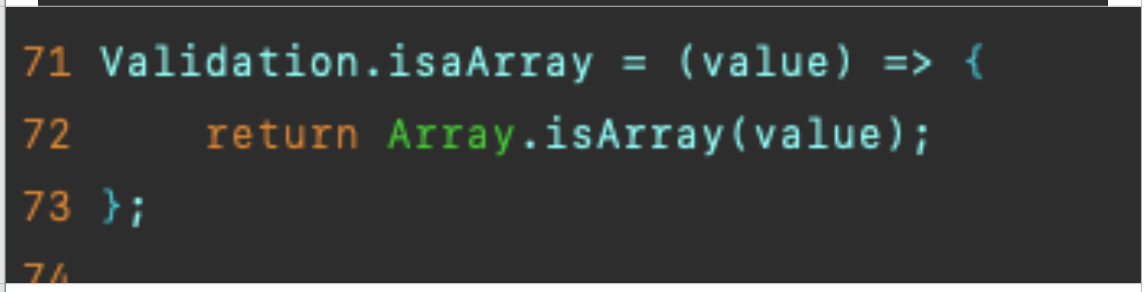
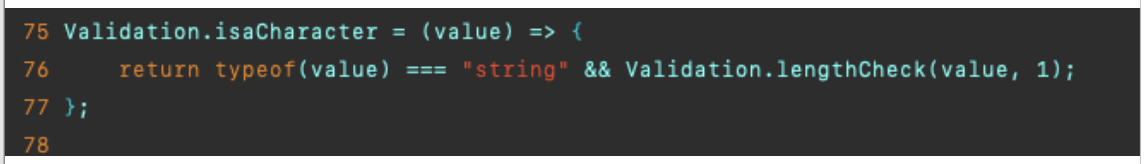
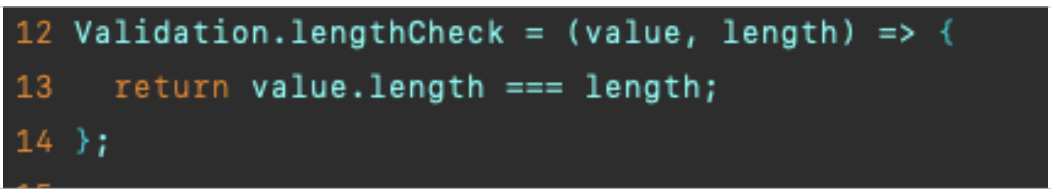
| Validation Type | Validation Code Snippet |
| --- | --- |
| | |

```
573
574 // method checks that the string passed in as an argument only contains lower case characters, upper case characters
575 // or number characters. if a character is found in the string that is not one of them, then the method returns false
576 // if all characters in the string that is passed as argument are either upper, lower or numbers then true is returned
577 Validation.checkStringIsMadeOfOnlyLowerUpperAndNumberChars = (stringValue, allowSpaces) => {
578     // iterate through each character in the string checking if it is a number, lowercase char or an uppercase char
579     let curChar = '';
580     let charArray = stringValue.split(""); // puts the string into an array of its characters
581     for (let i = 0; i < charArray.length; i++) {
582
583         curChar = charArray[i];
584
585         if (allowSpaces === true) {
586             // if all of the checks come back false, false is returned
587             if ((!Validation[_stringLookupCheckForLowerCase](curChar))
588                 && (!Validation[_stringLookupCheckForUpperCase](curChar))
589                 && (!Validation[_stringLookupCheckForNumbers](curChar))
590                 && (!Validation[_stringLookupCheckForSpaces](curChar)))
591             {
592                 return false;
593             }
594         }
595
596         else {
597             // if all of the checks come back false, false is returned
598             if ((!Validation[_stringLookupCheckForLowerCase](curChar))
599                 && (!Validation[_stringLookupCheckForUpperCase](curChar))
600                 && (!Validation[_stringLookupCheckForNumbers](curChar)))
601             {
602                 return false;
603             }
604         }
605
606     }
607
608     // loop has finished, if false hasn't been returned it means that all the characters
609     // in the string consist of either a number, lowercase char or an uppercase char
610     return true;
611 };
612
```



| Validation Type | Validation Code Snippet |
| --- | --- |
| |  |
| |  |
| Type Check | `Validation.isaNumber = (value, stringToNumberConversionNeeded) => {`<br><br>`    if (stringToNumberConversionNeeded === true) {`<br>`        value = Number(value);`<br>`    }`<br><br>`    return typeof(value) === "number" && !isNaN(value);`<br>`};` |
| | `Validation.isaInt = (value) => {`<br>`    return typeof(value) === "number" && !isNaN(value) && value % 1 === 0;`<br>`};` |
| | `Validation.isaReal = (value) => {`<br>`    return typeof(value) === "number" && !isNaN(value) && value % 1 !== 0;`<br>`};` |
| | `Validation.isaString = (value) => {`<br>`    return typeof(value) === "string";`<br>`};` |

| Validation Type | Validation Code Snippet |
| --- | --- |
| |  |
| |  |
| |  |
| |  |
| |  |
| **Length Check** |  |

| Validation Type | Validation Code Snippet |
|---|---|

**Range Check**

```
16 Validation.rangeCheck = (value, lower, upper, stringLengthInRange) => {
17     if (typeof value === "string"){
18         if (stringLengthInRange) {
19             value = value.length;
20         }
21         else {
22             value = Number(value);
23         }
24     }
25
26     if (upper === undefined || upper === null) {
27         upper = lower;
28     }
29
30     return (value >= lower && value <= upper );
31
32 };
```

**Format Check**

```
285 Validation.timeFormat_HHMM = (value) => {
286
287     let valueArray = [];
288     let val = 0;
289     let hour, minute;
290
291     // checks that the input is a string and that it has a length of 5 characters
292     if(Validation.isaString(value) && Validation.lengthCheck(value, 5)) {
293
294         valueArray = value.split(""); // puts the string into an array
295
296         // loops through every element in the array so that checks can be done on the hour and minute parts of the time
297         for (let i = 0; i < valueArray.length; i++) {
298
299             // will be using this a lot so to make it more readable, the current array element is set to the variable val
300             val = valueArray[i];
301
302             // the 2 index in the array as the element should be a ':' if it is not, false is returned
303             if (i === 2) {
304                 if (val !== ":") {
305                     return false;
306                 }
307             }
308             else {
309
310                 // checks that any other character is a number value, if it isn't false is returned
311                 if (!Validation.isaNumber(Number(val))) {
312                     return false;
313                 }
314             }
315         }
316
317         hour = valueArray[0] + valueArray[1];
318         minute = valueArray[3] + valueArray[4];
319
320         return ((Number(hour) >= 0 && Number(hour) <= 23 && Number(minute) >= 0  && Number(minute) <= 59));
321     }
322     else {
323         return false;
324     }
325 };
326
```

| Validation Type | Validation Code Snippet |
|---|---|
| | 84 Validation.fileIsTXT = (file) => {<br>85 let regex = /^([a-zA-Z0-9\s_\\.\-:])+(.txt)$/;<br>86 return (regex.test(file.value));<br>87 };<br>88 |
| | 79 Validation.fileIsCSV = (file) => {<br>80 let regex = /^([a-zA-Z0-9\s_\\.\-:])+(.csv)$/;<br>81 return (regex.test(file));<br>82 };<br>83 |
| | 327 Validation.emailFormat = (value) => {<br>328 let emailRegex = /^(([^<>()\[\]\\.,;:\s@"]+(\.[^<>()\[\]\\.,;:\s@"]+)*)\|(".+"))@((\[[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}])\|(([a-zA-Z\-0-9]+\.)+[a-zA-Z]{2,}))$/ ;<br>329 return emailRegex.test(value);<br>330 };<br>331 |
| **Presence Check** | 8 Validation.presenceCheck = (value) => {<br>9 return !(value === undefined \|\| value === null \|\| /^\s*$/.test(value));<br>10 }; |
| **Comparison Check** | 382 // EXAMPLE OF A POLYMORPHIC METHOD<br>383 // value two can be an array to check valueOne against multiple values in<br>384 // the valueTwo argument. although valueTwo can be a single value to check equality<br>385 Validation.comparisonCheck = (valueOne, valueTwo) => {<br>386 if (typeof valueTwo === "object") {<br>387 let foundMatchFlag = false;<br>388 // iterate through the array and compare valueOne to all the values in the array, if one matches then return true<br>389 // if there's no matches return false<br>390 for (let i = 0; i < valueTwo.length; i ++) {<br>391 if (valueOne === valueTwo[i]) {<br>392 foundMatchFlag = true;<br>393 }<br>394 }<br>395 return foundMatchFlag;<br>396<br>397 } else { // if valueTwo is not an array then check for equality on the single value<br>398 return valueOne === valueTwo;<br>399 }<br>400<br>401 }; |

# Source Code For Final System

## Customer Relationship Management System (CRM)

**Kieran Williams**

# Root Directory Contents

Kieran Marcus Williams

# Index.js

Kieran Marcus Williams

```
const express = require("express");
const app = express();
const methodOverride = require("method-override");
const bodyParser = require("body-parser");
const session = require('express-session');

//requiring routes
const
    indexRoutes = require("./routes/index"),
    analyticsRoutes = require("./routes/analyticsRoutes"),
    dashboardRoutes = require("./routes/dashboardRoutes"),
    contactsRoutes = require("./routes/contactsRoutes");




// so that I can pass data between the backend and front end without just xhr requests
// .urlencoded({extended: true}) allows me to pass nested objects to the front end and from the front end to the server
app.use(bodyParser.urlencoded({extended: true}));
app.set("view engine", "ejs"); //sets my view engine to the ejs templating language which will allow me to use js in the mark up
app.use('/public', express.static(__dirname + '/public'));
app.use(express.static(__dirname + '/public'));  // express middleware to serve the static files such as css and js to the client using the express HTTP framework
app.use(methodOverride("_method")); // methodOverride will allow me to use more http verbs than html5 forms support
app.use(session({secret: 'mySecret', resave: false, saveUninitialized: false})); // will allow me to store data to the session property

// setting my express http framework to use these url routes as the route paths for each
// of the different types of routes to aid readability with  RESTFUL ROUTES for each of the routes
// the routes that extend from their base
app.use("/", indexRoutes);
app.use("/contacts", contactsRoutes);
app.use("/analytics", analyticsRoutes);
app.use("/dashboard", dashboardRoutes);

app.listen(process.env.PORT || 8000, (err) => {
    if (err) {
        console.log(err);
    }
    else {
        console.log("server started on port 8000");
    }
});
```

# Package-lock.json

Kieran Marcus Williams

```
{
  "name": "kwcrmCourseworkProject",
  "version": "1.0.0",
  "lockfileVersion": 1,
  "requires": true,
  "dependencies": {
    "accepts": {
      "version": "1.3.5",
      "resolved": "https://registry.npmjs.org/accepts/-/accepts-1.3.5.tgz",
      "integrity": "sha1-63d99gEXI6OxTopywIBcjoZ0a9I=",
      "requires": {
        "mime-types": "2.1.20",
        "negotiator": "0.6.1"
      }
    },
    "array-flatten": {
      "version": "1.1.1",
      "resolved":
      "https://registry.npmjs.org/array-flatten/-/array-flatten-1.1.1.tgz",
      "integrity": "sha1-ml9pkFGx5wczKPKgCJaLZOopVdI="
    },
    "body-parser": {
      "version": "1.18.3",
      "resolved": "https://registry.npmjs.org/body-parser/-/body-parser-1.18.3.tgz",
      "integrity": "sha1-WykhmP/dVTs6DyDe0FkrlWlVyLQ=",
      "requires": {
        "bytes": "3.0.0",
        "content-type": "1.0.4",
        "debug": "2.6.9",
        "depd": "1.1.2",
        "http-errors": "1.6.3",
        "iconv-lite": "0.4.23",
        "on-finished": "2.3.0",
        "qs": "6.5.2",
        "raw-body": "2.3.3",
        "type-is": "1.6.16"
      }
    },
    "bytes": {
      "version": "3.0.0",
      "resolved": "https://registry.npmjs.org/bytes/-/bytes-3.0.0.tgz",
      "integrity": "sha1-0ygVQE1olpn4Wk6k+odV3ROpYEg="
    },
    "content-disposition": {
      "version": "0.5.2",
      "resolved":
      "https://registry.npmjs.org/content-disposition/-/content-disposition-0.5.2.tgz"
      ,
      "integrity": "sha1-DPaLud318r55YcOoUXjLhdunjLQ="
    },
    "content-type": {
      "version": "1.0.4",
      "resolved": "https://registry.npmjs.org/content-type/-/content-type-1.0.4.tgz",
      "integrity":
      "sha512-hIP3EEPs8tB9AT1L+NUqtwOAps4mk2Zob89MWXMHjHWg9milF/j4osnnQLXBCBFBk/tvIG/t
      Uc9mOUJiPBhPXA=="
    },
    "cookie": {
      "version": "0.3.1",
      "resolved": "https://registry.npmjs.org/cookie/-/cookie-0.3.1.tgz",
      "integrity": "sha1-5+Ch+e9DtMi6klxcWpboBtFoc7s="
    },
    "cookie-signature": {
      "version": "1.0.6",
      "resolved":
      "https://registry.npmjs.org/cookie-signature/-/cookie-signature-1.0.6.tgz",
      "integrity": "sha1-4wOogrNCzD7oylE6eZmXNNqzriw="
    },
    "crc": {
      "version": "3.4.4",
      "resolved": "https://registry.npmjs.org/crc/-/crc-3.4.4.tgz",
      "integrity": "sha1-naHpgOO9RPxck79as9ozeNheRms="
    },
```

```
      "debug": {
        "version": "2.6.9",
        "resolved": "https://registry.npmjs.org/debug/-/debug-2.6.9.tgz",
        "integrity":
        "sha512-bC7ElrdJaJnPbAP+1EotYvqZsb3ecl5wi6Bfi6BJTUcNowp6cvspg0jXznRTKDjm/E7AdgFB
        VeAPVMNcKGsHMA==",
        "requires": {
          "ms": "2.0.0"
        }
      },
      "depd": {
        "version": "1.1.2",
        "resolved": "https://registry.npmjs.org/depd/-/depd-1.1.2.tgz",
        "integrity": "sha1-m81S4UwJd2PnSbJ0xDRu0uVgtak="
      },
      "destroy": {
        "version": "1.0.4",
        "resolved": "https://registry.npmjs.org/destroy/-/destroy-1.0.4.tgz",
        "integrity": "sha1-l4hXRCxEdJ5CBmE+N5RiBYJqvYA="
      },
      "ee-first": {
        "version": "1.1.1",
        "resolved": "https://registry.npmjs.org/ee-first/-/ee-first-1.1.1.tgz",
        "integrity": "sha1-WQxhFWsK4vTwJVcyoViyZrxWsh0="
      },
      "ejs": {
        "version": "2.6.1",
        "resolved": "https://registry.npmjs.org/ejs/-/ejs-2.6.1.tgz",
        "integrity":
        "sha512-0xy4A/twfrRCnkhfk8ErDi5DqdAsAqeGxht4xkCUrsvhhbQNs7E+4jV0CN7+NKIY0aHE72+X
        vqtBIXzD31ZbXQ=="
      },
      "encodeurl": {
        "version": "1.0.2",
        "resolved": "https://registry.npmjs.org/encodeurl/-/encodeurl-1.0.2.tgz",
        "integrity": "sha1-rT/0yG7C0CkyL1oCw6mmBslbP1k="
      },
      "escape-html": {
        "version": "1.0.3",
        "resolved": "https://registry.npmjs.org/escape-html/-/escape-html-1.0.3.tgz",
        "integrity": "sha1-Aljq5NPQwJdN4cFpGI7wBR0dGYg="
      },
      "etag": {
        "version": "1.8.1",
        "resolved": "https://registry.npmjs.org/etag/-/etag-1.8.1.tgz",
        "integrity": "sha1-Qa4u62XvpiJorr/qg6x9eSmbCIc="
      },
      "express": {
        "version": "4.16.3",
        "resolved": "http://registry.npmjs.org/express/-/express-4.16.3.tgz",
        "integrity": "sha1-avilAjUNsyRuzEvs9rWjTSL37VM=",
        "requires": {
          "accepts": "1.3.5",
          "array-flatten": "1.1.1",
          "body-parser": "1.18.2",
          "content-disposition": "0.5.2",
          "content-type": "1.0.4",
          "cookie": "0.3.1",
          "cookie-signature": "1.0.6",
          "debug": "2.6.9",
          "depd": "1.1.2",
          "encodeurl": "1.0.2",
          "escape-html": "1.0.3",
          "etag": "1.8.1",
          "finalhandler": "1.1.1",
          "fresh": "0.5.2",
          "merge-descriptors": "1.0.1",
          "methods": "1.1.2",
          "on-finished": "2.3.0",
          "parseurl": "1.3.2",
          "path-to-regexp": "0.1.7",
          "proxy-addr": "2.0.4",
          "qs": "6.5.1",
```

```
      "range-parser": "1.2.0",
      "safe-buffer": "5.1.1",
      "send": "0.16.2",
      "serve-static": "1.13.2",
      "setprototypeof": "1.1.0",
      "statuses": "1.4.0",
      "type-is": "1.6.16",
      "utils-merge": "1.0.1",
      "vary": "1.1.2"
    },
    "dependencies": {
      "body-parser": {
        "version": "1.18.2",
        "resolved":
        "https://registry.npmjs.org/body-parser/-/body-parser-1.18.2.tgz",
        "integrity": "sha1-h2eKGdhLR9hZuDGZvVm84iKxBFQ=",
        "requires": {
          "bytes": "3.0.0",
          "content-type": "1.0.4",
          "debug": "2.6.9",
          "depd": "1.1.2",
          "http-errors": "1.6.3",
          "iconv-lite": "0.4.19",
          "on-finished": "2.3.0",
          "qs": "6.5.1",
          "raw-body": "2.3.2",
          "type-is": "1.6.16"
        }
      },
      "iconv-lite": {
        "version": "0.4.19",
        "resolved": "https://registry.npmjs.org/iconv-lite/-/iconv-lite-0.4.19.tgz",
        "integrity":
        "sha512-oTZqweIP51xaGPI4uPa56/Pri/480R+mo7SeU+YETByQNhDG55ycFyNLIgta9vXhILrx
        XDmF7ZGhqZIcuN0gJQ=="
      },
      "qs": {
        "version": "6.5.1",
        "resolved": "https://registry.npmjs.org/qs/-/qs-6.5.1.tgz",
        "integrity":
        "sha512-eRzhrN1WSINYCDCbrz796z37LOe3m5tmW7RQf6oBntukAG1nmovJvhnwHHRMAfeoItc1
        m2Hk02WER2aQ/iqs+A=="
      },
      "raw-body": {
        "version": "2.3.2",
        "resolved": "https://registry.npmjs.org/raw-body/-/raw-body-2.3.2.tgz",
        "integrity": "sha1-vNYMd9Prk83gBQKVw/N5OJvIj4k=",
        "requires": {
          "bytes": "3.0.0",
          "http-errors": "1.6.2",
          "iconv-lite": "0.4.19",
          "unpipe": "1.0.0"
        },
        "dependencies": {
          "depd": {
            "version": "1.1.1",
            "resolved": "https://registry.npmjs.org/depd/-/depd-1.1.1.tgz",
            "integrity": "sha1-V4O04cRZ8G+lyif5kfPQbnoxA1k="
          },
          "http-errors": {
            "version": "1.6.2",
            "resolved":
            "https://registry.npmjs.org/http-errors/-/http-errors-1.6.2.tgz",
            "integrity": "sha1-CgAsyFcHGSp+eUbO7cERVfYOxzY=",
            "requires": {
              "depd": "1.1.1",
              "inherits": "2.0.3",
              "setprototypeof": "1.0.3",
              "statuses": "1.4.0"
            }
          },
          "setprototypeof": {
            "version": "1.0.3",
```

```
                "resolved":
                "https://registry.npmjs.org/setprototypeof/-/setprototypeof-1.0.3.tgz",
                "integrity": "sha1-ZlZ+NwQ+608E2RvWWMDL77VbjgQ="
              }
            }
          },
          "statuses": {
            "version": "1.4.0",
            "resolved": "https://registry.npmjs.org/statuses/-/statuses-1.4.0.tgz",
            "integrity":
            "sha512-zhSCtt8v2NDrRlPQpCNtw/heZLtfUDqxBM1udqikb/Hbk52LK4nQSwr10u77iopCW5Ls
            yHpuXS0GnEc48mLeew=="
          }
        }
      },
      "express-session": {
        "version": "1.15.6",
        "resolved":
        "https://registry.npmjs.org/express-session/-/express-session-1.15.6.tgz",
        "integrity":
        "sha512-r0nrHTCYtAMrFwZ0kBzZEXa1vtPVrw0dKvGSrKP4dahwBQ1BJpF2/y1Pp4sCD/0kvxV4zZec
        lyvfmw0B4RMJQA==",
        "requires": {
          "cookie": "0.3.1",
          "cookie-signature": "1.0.6",
          "crc": "3.4.4",
          "debug": "2.6.9",
          "depd": "1.1.2",
          "on-headers": "1.0.1",
          "parseurl": "1.3.2",
          "uid-safe": "2.1.5",
          "utils-merge": "1.0.1"
        }
      },
      "filereader": {
        "version": "0.10.3",
        "resolved": "https://registry.npmjs.org/filereader/-/filereader-0.10.3.tgz",
        "integrity": "sha1-x0fUos2PYeVBinwH/hJXpD8KzbE="
      },
      "finalhandler": {
        "version": "1.1.1",
        "resolved": "https://registry.npmjs.org/finalhandler/-/finalhandler-1.1.1.tgz",
        "integrity":
        "sha512-Y1GUDo39ez4aHAw7MysnUD5JzYX+WaIj8I57kO3aEPT1fFRL4sr7mjei97FgnwhAyyzRYmQZ
        aTHb2+9uZ1dPtg==",
        "requires": {
          "debug": "2.6.9",
          "encodeurl": "1.0.2",
          "escape-html": "1.0.3",
          "on-finished": "2.3.0",
          "parseurl": "1.3.2",
          "statuses": "1.4.0",
          "unpipe": "1.0.0"
        },
        "dependencies": {
          "statuses": {
            "version": "1.4.0",
            "resolved": "https://registry.npmjs.org/statuses/-/statuses-1.4.0.tgz",
            "integrity":
            "sha512-zhSCtt8v2NDrRlPQpCNtw/heZLtfUDqxBM1udqikb/Hbk52LK4nQSwr10u77iopCW5Ls
            yHpuXS0GnEc48mLeew=="
          }
        }
      },
      "forwarded": {
        "version": "0.1.2",
        "resolved": "https://registry.npmjs.org/forwarded/-/forwarded-0.1.2.tgz",
        "integrity": "sha1-mMI9qxF1ZXuMBXPozszZGw/xjIQ="
      },
      "fresh": {
        "version": "0.5.2",
        "resolved": "https://registry.npmjs.org/fresh/-/fresh-0.5.2.tgz",
        "integrity": "sha1-PYyt2Q2XZWn6g1qx+OSyOhBWBac="
```

```
    },
    "http-errors": {
      "version": "1.6.3",
      "resolved": "http://registry.npmjs.org/http-errors/-/http-errors-1.6.3.tgz",
      "integrity": "sha1-i1VoC7S+KDoLW/TqLjhYC+HZMg0=",
      "requires": {
        "depd": "1.1.2",
        "inherits": "2.0.3",
        "setprototypeof": "1.1.0",
        "statuses": "1.5.0"
      }
    },
    "iconv-lite": {
      "version": "0.4.23",
      "resolved": "https://registry.npmjs.org/iconv-lite/-/iconv-lite-0.4.23.tgz",
      "integrity":
      "sha512-neyTUVFtahjf0mB3dZT77u+8O0QB89jFdnBkd5P1JgYPbPaia3gXXOVL2fq8VyU2gMMD7SaN
      7QukTB/pmXYvDA==",
      "requires": {
        "safer-buffer": "2.1.2"
      }
    },
    "inherits": {
      "version": "2.0.3",
      "resolved": "https://registry.npmjs.org/inherits/-/inherits-2.0.3.tgz",
      "integrity": "sha1-Yzwsg+PaQqUC9SRmAiSA9CCCYd4="
    },
    "ipaddr.js": {
      "version": "1.8.0",
      "resolved": "https://registry.npmjs.org/ipaddr.js/-/ipaddr.js-1.8.0.tgz",
      "integrity": "sha1-6qM9bd16zo9/b+DJygRA5wZzix4="
    },
    "media-typer": {
      "version": "0.3.0",
      "resolved": "https://registry.npmjs.org/media-typer/-/media-typer-0.3.0.tgz",
      "integrity": "sha1-hxDXrwqmJvj/+hzgAWhUUmMlV0g="
    },
    "merge-descriptors": {
      "version": "1.0.1",
      "resolved":
      "https://registry.npmjs.org/merge-descriptors/-/merge-descriptors-1.0.1.tgz",
      "integrity": "sha1-sAqqVW3YtEVoFQ7J0blT8/kMu2E="
    },
    "method-override": {
      "version": "3.0.0",
      "resolved":
      "https://registry.npmjs.org/method-override/-/method-override-3.0.0.tgz",
      "integrity":
      "sha512-IJ2NNN/mSl9w3kzWB92rcdHpz+HjkxhDJWNDBqSlas+zQdP8wBiJzITPg08M/k2uVvMow7Sk
      41atndNtt/PHSA==",
      "requires": {
        "debug": "3.1.0",
        "methods": "1.1.2",
        "parseurl": "1.3.2",
        "vary": "1.1.2"
      },
      "dependencies": {
        "debug": {
          "version": "3.1.0",
          "resolved": "https://registry.npmjs.org/debug/-/debug-3.1.0.tgz",
          "integrity":
          "sha512-OX8XqP7/1a9cqkxYw2yXss15f26NKWBpDXQd0/uK/KPqdQhxbPa994hnzjcE2VqQpDsl
          f55723cKPUOGSmMY3g==",
          "requires": {
            "ms": "2.0.0"
          }
        }
      }
    },
    "methods": {
      "version": "1.1.2",
      "resolved": "https://registry.npmjs.org/methods/-/methods-1.1.2.tgz",
      "integrity": "sha1-VSmk1nZUE07cxSZmVoNbD4Ua/O4="
```

```
    },
    "mime": {
      "version": "1.4.1",
      "resolved": "https://registry.npmjs.org/mime/-/mime-1.4.1.tgz",
      "integrity":
      "sha512-KI1+qOZu5DcW6wayYHSzR/tXKCDC5Om4s1z2QJjDULzLcmf3DvzS7oluY4HCTrc+9FiKmWUg
      eNLg7W3uIQvxtQ=="
    },
    "mime-db": {
      "version": "1.36.0",
      "resolved": "https://registry.npmjs.org/mime-db/-/mime-db-1.36.0.tgz",
      "integrity":
      "sha512-L+xvyD9MkoYMXb1jAmzI/lWYAxAMCPvIBSWur0PZ5nOf5euahRLVqH//FKW9mWp2lkqUgYiX
      PgkzfMUFi4zVDw=="
    },
    "mime-types": {
      "version": "2.1.20",
      "resolved": "https://registry.npmjs.org/mime-types/-/mime-types-2.1.20.tgz",
      "integrity":
      "sha512-HrkrPaP9vGuWbLK1B1FfgAkbqNjIuy4eHlIYnFi7kamZyLLrGlo2mpcx0bBmNpKqBtYtAfGb
      odDddIgddSJC2A==",
      "requires": {
        "mime-db": "1.36.0"
      }
    },
    "moment": {
      "version": "2.24.0",
      "resolved": "https://registry.npmjs.org/moment/-/moment-2.24.0.tgz",
      "integrity":
      "sha512-bV7f+6l2QigeBBZSM/6yTNq4P2fNpSWj/0e7jQcy87A8e7o2nAfP/34/2ky5Vw4B9S446EtI
      hodAzkFCcR4dQg=="
    },
    "ms": {
      "version": "2.0.0",
      "resolved": "https://registry.npmjs.org/ms/-/ms-2.0.0.tgz",
      "integrity": "sha1-VgiurfwAvmwpAd9fmGF4jeDVl8g="
    },
    "negotiator": {
      "version": "0.6.1",
      "resolved": "https://registry.npmjs.org/negotiator/-/negotiator-0.6.1.tgz",
      "integrity": "sha1-KzJxhOiZIQEXeyhWP7XnECrNDKk="
    },
    "on-finished": {
      "version": "2.3.0",
      "resolved": "https://registry.npmjs.org/on-finished/-/on-finished-2.3.0.tgz",
      "integrity": "sha1-IPEzZIGwg811M3mSoWlxqi2QaUc=",
      "requires": {
        "ee-first": "1.1.1"
      }
    },
    "on-headers": {
      "version": "1.0.1",
      "resolved": "https://registry.npmjs.org/on-headers/-/on-headers-1.0.1.tgz",
      "integrity": "sha1-ko9dD0cNSTQmUepnlLCFfBAGk/c="
    },
    "parseurl": {
      "version": "1.3.2",
      "resolved": "https://registry.npmjs.org/parseurl/-/parseurl-1.3.2.tgz",
      "integrity": "sha1-/CidTtiZMRlGDBViUyYs3I3mW/M="
    },
    "path-to-regexp": {
      "version": "0.1.7",
      "resolved":
      "https://registry.npmjs.org/path-to-regexp/-/path-to-regexp-0.1.7.tgz",
      "integrity": "sha1-32BBeABfUi8V60SQ5yR6G/qmf4w="
    },
    "proxy-addr": {
      "version": "2.0.4",
      "resolved": "https://registry.npmjs.org/proxy-addr/-/proxy-addr-2.0.4.tgz",
      "integrity":
      "sha512-5erio2h9jp5CHGwcybmxmVqHmnCBZeewlfJ0pex+UW7Qny7OOZXTtH56TGNyBizkgiOwhJtM
      KrVzDTeKcySZwA==",
      "requires": {
```

```
        "forwarded": "0.1.2",
        "ipaddr.js": "1.8.0"
      }
    },
    "qs": {
      "version": "6.5.2",
      "resolved": "https://registry.npmjs.org/qs/-/qs-6.5.2.tgz",
      "integrity":
      "sha512-N5ZAX4/LxJmF+7wN74pUD6qAh9/wnvdQcjq9TZjevvXzSUo7bfmw91saqMjzGS2xq91/odN2
      dW/WOl7qQHNDGA=="
    },
    "random-bytes": {
      "version": "1.0.0",
      "resolved": "https://registry.npmjs.org/random-bytes/-/random-bytes-1.0.0.tgz",
      "integrity": "sha1-T2ih3Arli9P7lYSMMDJNt11kNgs="
    },
    "range-parser": {
      "version": "1.2.0",
      "resolved": "https://registry.npmjs.org/range-parser/-/range-parser-1.2.0.tgz",
      "integrity": "sha1-9JvmtIeJTdxA3MlKMi9hEJLgDV4="
    },
    "raw-body": {
      "version": "2.3.3",
      "resolved": "https://registry.npmjs.org/raw-body/-/raw-body-2.3.3.tgz",
      "integrity":
      "sha512-9esiElv1BrZoI3rCDuOuKCBRbuApGGaDPQfjSflGxdy4oyzqghxu6klEkkVIvBje+FF0BX9c
      oEv8KqW6X/7njw==",
      "requires": {
        "bytes": "3.0.0",
        "http-errors": "1.6.3",
        "iconv-lite": "0.4.23",
        "unpipe": "1.0.0"
      }
    },
    "safe-buffer": {
      "version": "5.1.1",
      "resolved": "https://registry.npmjs.org/safe-buffer/-/safe-buffer-5.1.1.tgz",
      "integrity":
      "sha512-kKvNJn6Mm93gAczWVJg7wH+wGYWNrDHdWvpUmHyEsgCtIwwo3bqPtV4tR5tuPaUhTOo/kvhV
      wd8XwwOllGYkbg=="
    },
    "safer-buffer": {
      "version": "2.1.2",
      "resolved": "https://registry.npmjs.org/safer-buffer/-/safer-buffer-2.1.2.tgz",
      "integrity":
      "sha512-YZo3K82SD7Riyi0E1EQPojLz7kpepnSQI9IyPbHHg1XXXevb5dJI7tpyN2ADxGcQbHG7vcyR
      Hk0cbwqcQriUtg=="
    },
    "send": {
      "version": "0.16.2",
      "resolved": "https://registry.npmjs.org/send/-/send-0.16.2.tgz",
      "integrity":
      "sha512-E64YFPUssFHEFBvpbbjr44NCLtI1AohxQ8ZSiJjQLskAdKuriYEP6VyGEsRDH8ScozGpkaX1
      BGvhanqCwkcEZw==",
      "requires": {
        "debug": "2.6.9",
        "depd": "1.1.2",
        "destroy": "1.0.4",
        "encodeurl": "1.0.2",
        "escape-html": "1.0.3",
        "etag": "1.8.1",
        "fresh": "0.5.2",
        "http-errors": "1.6.3",
        "mime": "1.4.1",
        "ms": "2.0.0",
        "on-finished": "2.3.0",
        "range-parser": "1.2.0",
        "statuses": "1.4.0"
      },
      "dependencies": {
        "statuses": {
          "version": "1.4.0",
          "resolved": "https://registry.npmjs.org/statuses/-/statuses-1.4.0.tgz",
```

```
      "integrity":
      "sha512-zhSCtt8v2NDrRlPQpCNtw/heZLtfUDqxBM1udqikb/Hbk52LK4nQSwr10u77iopCW5LsyHpuXS0GnEc48mLeew=="
    }
  }
},
"serve-static": {
  "version": "1.13.2",
  "resolved": "https://registry.npmjs.org/serve-static/-/serve-static-1.13.2.tgz",
  "integrity":
  "sha512-p/tdJrO4U387R9oMjb1oj7qSMaMfmOyd4j9hOFoxZe2baQszgHcSWjuya/CiT5kgZZKRudHNOA0pYXOl8rQ5nw==",
  "requires": {
    "encodeurl": "1.0.2",
    "escape-html": "1.0.3",
    "parseurl": "1.3.2",
    "send": "0.16.2"
  }
},
"setprototypeof": {
  "version": "1.1.0",
  "resolved":
  "https://registry.npmjs.org/setprototypeof/-/setprototypeof-1.1.0.tgz",
  "integrity":
  "sha512-BvE/TwpZX4FXExxOxZyRGQQv651MSwmWKZGqvmPcRIjDqWub67kTKuIMx43cZZrS/cBBzwBcNDWoFxt2XEFIpQ=="
},
"statuses": {
  "version": "1.5.0",
  "resolved": "https://registry.npmjs.org/statuses/-/statuses-1.5.0.tgz",
  "integrity": "sha1-Fhx9rBd2Wf2YEfQ3cfqZOBR4Yow="
},
"type-is": {
  "version": "1.6.16",
  "resolved": "https://registry.npmjs.org/type-is/-/type-is-1.6.16.tgz",
  "integrity":
  "sha512-HRkVv/5qY2G6I8iab9cI7v1bOIdhm94dVjQCPFElW9W+3GeDOSHmy2EBYe4VTApuzolPcmgFTN3ftVJRKR2J9Q==",
  "requires": {
    "media-typer": "0.3.0",
    "mime-types": "2.1.20"
  }
},
"uid-safe": {
  "version": "2.1.5",
  "resolved": "https://registry.npmjs.org/uid-safe/-/uid-safe-2.1.5.tgz",
  "integrity":
  "sha512-KPHm4VL5dDXKz01UuEd88Df+KzynaohSL9fBh096KWAxSKZQDI2uBrVqtvRM4rwrIrRRKsdLNML/lnaaVSRioA==",
  "requires": {
    "random-bytes": "1.0.0"
  }
},
"uniqid": {
  "version": "5.0.3",
  "resolved": "https://registry.npmjs.org/uniqid/-/uniqid-5.0.3.tgz",
  "integrity":
  "sha512-R2qx3X/LYWSdGRaluio4dYrPXAJACTqyUjuyXHoJLBUOIfmMcnYOyY2d6Y4clZcIz5lK6ZaI0Zzmm0cPfsIqzQ=="
},
"unpipe": {
  "version": "1.0.0",
  "resolved": "https://registry.npmjs.org/unpipe/-/unpipe-1.0.0.tgz",
  "integrity": "sha1-sr9O6FFKrmFltIF4KdIbLvSZBOw="
},
"utils-merge": {
  "version": "1.0.1",
  "resolved": "https://registry.npmjs.org/utils-merge/-/utils-merge-1.0.1.tgz",
  "integrity": "sha1-n5VxD1CiZ5R7LMwSR0HBAoQn5xM="
},
"vary": {
  "version": "1.1.2",
  "resolved": "https://registry.npmjs.org/vary/-/vary-1.1.2.tgz",
```

```
        "integrity": "sha1-IpnwLG3tMNSllhsLn3RSShj2NPw="
      }
    }
  }
```

# Package.json

Kieran Marcus Williams

```
{
  "name": "kwcrmCourseworkProject",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
    "start": "node index.js"
  },
  "author": "kieran williams",
  "license": "ISC",
  "dependencies": {
    "body-parser": "^1.18.3",
    "ejs": "^2.6.1",
    "express": "^4.16.3",
    "express-session": "^1.15.6",
    "filereader": "^0.10.3",
    "method-override": "^3.0.0",
    "moment": "^2.24.0",
    "uniqid": "^5.0.3"
  }
}
```

# Classes Directory Contents

Kieran Marcus Williams

# Analyticals.js

Kieran Marcus Williams

```
const DateHandler = require('./DateHandler');
const ConversionRate = require('./ConversionRate');
const GenerateDataPoints = require('./GenerateDataPoints');


const _clientsGainedFilePath = Symbol("clientsGainedFilePath");
const _clientsLostFilePath = Symbol("clientsLostFilePath");
const _leadsLostFilePath = Symbol("leadsLostFilePath");

// The class that will be used to handle all back-end tasks of the Analyticals page.
class Analyticals {

    constructor(clientsGainedCSVHandlerObject, clientsLostCSVHandlerObject,
    leadsLostCSVHandlerObject) {

        this[_clientsGainedFilePath] = clientsGainedCSVHandlerObject;
        this[_clientsLostFilePath] = clientsLostCSVHandlerObject;
        this[_leadsLostFilePath] = leadsLostCSVHandlerObject;
    }

    // this static method will return an object for the conversion rate of
    // the month as both a percentage and as a ratio in it's lowest form
     GetConversionRateForTheMonth () {

        let clientsGainedFile = this[_clientsGainedFilePath];
        let leadsLostFile = this[_leadsLostFilePath];

        let currentDate = DateHandler.getCurrentDate();
        let numberOfClientsGainedThisMonth;
        let numberOfLeadsLostThisMonth;


        // to find data for clients gained and leads lost this month I will have to
        find all the matches
        // in the second index against the current date in standard time with the
        last 18 characters truncated
        // as in the 2nd index of all the analytics files there is the month and
        year stored as yyyy-mm

        currentDate = currentDate.split(""); // turns the date into an array of
        characters
        currentDate.splice(7,18); // truncates the last 18 characters from the array
        currentDate = currentDate.join(""); // joins the array together into a
        string and stores the value into the variable

        numberOfClientsGainedThisMonth = clientsGainedFile.countMatches([2],
        currentDate);
        numberOfLeadsLostThisMonth = leadsLostFile.countMatches([2], currentDate);

        return ConversionRate.conversionRate(numberOfClientsGainedThisMonth,
        numberOfLeadsLostThisMonth);
    }


    // will return the data points and labels for both clients gained and lost
    // done for the past 14 days

     getDataPointsAndLabelsForThePast14Days () {

        let clientsLost = new GenerateDataPoints("clientsLost",
            this[_clientsGainedFilePath],
            this[_clientsLostFilePath],
            this[_leadsLostFilePath]);

        let clientsGained = new GenerateDataPoints("clientsGained",
            this[_clientsGainedFilePath],
            this[_clientsLostFilePath],
            this[_leadsLostFilePath]);

        // returns an object containing data for the clients gained and lost with
        their relative
        // data points and labels to go along with both

```

```
65          return {
66
67              clientsGained: {
68                  dataPoints: clientsGained.getPast14DaysOfDataPoints().dataPoints,
69                  labels: clientsGained.getPast14DaysOfDataPoints().labels
70              },
71
72              clientsLost: {
73                  dataPoints: clientsLost.getPast14DaysOfDataPoints().dataPoints,
74                  labels: clientsLost.getPast14DaysOfDataPoints().labels
75              }
76          }
77
78      }
79
80
81      // will return the data points and labels for both clients gained and lost
82      // done for the past 12 weeks
83
84       getDataPointsAndLabelsForThePast12Weeks () {
85
86          let clientsLost = new GenerateDataPoints("clientsLost",
87              this[_clientsGainedFilePath],
88              this[_clientsLostFilePath],
89              this[_leadsLostFilePath]);
90
91          let clientsGained = new GenerateDataPoints("clientsGained",
92              this[_clientsGainedFilePath],
93              this[_clientsLostFilePath],
94              this[_leadsLostFilePath]);
95
96          // returns an object containing data for the clients gained and lost with
            their relative
97          // data points and labels to go along with both
98
99          return {
100
101             clientsGained: {
102                 dataPoints: clientsGained.getDataPointsForLast12Weeks().dataPoints,
103                 labels: clientsGained.getDataPointsForLast12Weeks().labels
104             },
105
106             clientsLost: {
107                 dataPoints: clientsLost.getDataPointsForLast12Weeks().dataPoints,
108                 labels: clientsLost.getDataPointsForLast12Weeks().labels
109             }
110         }
111
112     }
113
114     // will return the data points and labels for both clients gained and lost
115     // done for the past 12 months
116
117      getDataPointsAndLabelsForThePast12Months () {
118
119         let clientsLost = new GenerateDataPoints("clientsLost",
120             this[_clientsGainedFilePath],
121             this[_clientsLostFilePath],
122             this[_leadsLostFilePath]);
123
124         let clientsGained = new GenerateDataPoints("clientsGained",
125             this[_clientsGainedFilePath],
126             this[_clientsLostFilePath],
127             this[_leadsLostFilePath]);
128
129         // returns an object containing data for the clients gained and lost with
            their relative
130         // data points and labels to go along with both
131
132         return {
133
134             clientsGained: {
135                 dataPoints: clientsGained.getDataPointsForLast12Months().dataPoints,
```

```
                labels: clientsGained.getDataPointsForLast12Months().labels
            },

            clientsLost: {
                dataPoints: clientsLost.getDataPointsForLast12Months().dataPoints,
                labels: clientsLost.getDataPointsForLast12Months().labels
            }
        }

    }

}


module.exports = Analyticals;

```

# ConversionRate.js

Kieran Marcus Williams

```
1
2   // declaring the symbols so that I can imitate private variables and methods
3   // will prevent the methods and variables appearing as suggestions when I type
    'Analytics.'
4   // will only show when I type 'Analytics[_' this will make the class easier to use
    else where when
5   // an instance of this class is made and help ensure that everything works as intended
6   const _greatestCommonDivisor = Symbol('greatestCommonDivisor');
7   const _getConversionRateRatioAsString = Symbol('getConversionRateRatioAsString');
8   const _getConversionRatePercentageAsString =
    Symbol('getConversionRatePercentageAsString');
9
10  // The class that will be used to create a conversion rate
11  class ConversionRate {
12
13      // uses A RECURSIVE ALGORITHM to find the greatest common divisor
14      // uses the basis of the Euclidean algorithm
15      static [_greatestCommonDivisor] (valueOne, valueTwo)  {
16          let remainder;
17          if (!valueTwo) { // conditional gets met when valueTwo ('remainder' after
            first call) == 0 (terminating condition)
18              return valueOne; // valueOne is now our greatest common divisor
19          }
20          remainder = valueOne % valueTwo; // calculates the remainder
21          // calls its self again with the valueOne parameter passing in the valueTwo
            as an argument
22          // and in the place of the valueTwo parameter the remainder is passed as an
            argument
23          return this[_greatestCommonDivisor](valueTwo, remainder);
24      };
25
26
27      // method outputs a ratio as type string of the conversion rate
28      static [_getConversionRateRatioAsString] (numberOfClientsGained,
        numberOfLeadsLost) {
29
30          // if both the numberOfClientsGained and the numberOfLeadsLost  is == 0,
31          // output a dash so that the output is not 'NaN : NaN'
32          if (numberOfClientsGained === 0 && numberOfLeadsLost === 0) {
33              return '-'
34          }
35          // stores the greatest common divisor in the gcd variable so that the ratio
            can be
36          // shown in it's lowest form
37          let gcd = this[_greatestCommonDivisor](numberOfClientsGained,
            numberOfLeadsLost);
38          // the ratio values are calculated by dividing the inputs by the greatest
            common divisor
39          // the output is a string such as "2:3" if the inputs for example were 20
            and 30
40          return `${numberOfClientsGained/gcd} : ${numberOfLeadsLost/gcd}`;
41      }
42
43      // method outputs a percentage value as type string of the conversion rate
44      static [_getConversionRatePercentageAsString](numberOfClientsGained,
        numberOfLeadsLost){
45          // if both the numberOfClientsGained and the numberOfLeadsLost  is == 0,
46          // output a dash so that the output is not 'NaN%'
47          if (numberOfClientsGained === 0 && numberOfLeadsLost === 0) {
48              return '-'
49          }
50          // total is calculated to get a divisor
51          let total  = numberOfClientsGained + numberOfLeadsLost;
52          // percentage is calculated by dividing the numberOfClients gained by the
            total variable
53          // then multiplying by 100 to get the number as a value between 0 and 100
54          // Math.floor() method is used to truncate the value or any decimal values
            it may have
55          let percentage = Math.floor((numberOfClientsGained / total) * 100);
56          // the return is the percentage
57          return `${percentage}%`
58      }
59
```

```
60
61
62
63
64      // public static method that can be called anywhere without instantiating the
        class
65      // used to get an object containing the conversion rate as a ratio and as a
66      // percentage. arguments passed should be of type number
67      static conversionRate (numberOfClientsGained, numberOfLeadsLost) {
68
69          let conversionRateRatio =
            this[_getConversionRateRatioAsString](numberOfClientsGained,
            numberOfLeadsLost);
70          let conversionRatePercentage =
            this[_getConversionRatePercentageAsString](numberOfClientsGained,
            numberOfLeadsLost);
71
72
73          return {
74              ratio: conversionRateRatio,
75              percentage: conversionRatePercentage
76          }
77      }
78  }
79
80  module.exports = ConversionRate;
81
82
83
```

# CSVHandler.js

Kieran Marcus Williams

```
"use strict";

const
    fs = require("fs"),
    path = require('path'),
    uniqueID = require('uniqid'),
    Sort = require("./Sort"),
    Search = require("./Search"),
    DateHandler = require("./DateHandler");


// way of imitating private methods and variables as the JS does not have a
// a way to define them yet. Done by differentiating the way
// they are accessed. It isn't, accessible through the object model
// of the class like you would expect the method or variable to be.
// This is done so that it is clear what is meant to be accessible and
// what is not, from outside the class. Even tho the IDE won't
// suggest the methods or variables, it is still possible to
// access them. And so, this is not perfect but better than nothing.

const _csvToTxt = Symbol('csvToTxt');
const _txtToCsv = Symbol('txtToCsv');
const _csvToArray = Symbol('txtInCsvFormatToArray');
const _FILE_PATH = Symbol('filePath');
const _rewriteDataToFileWithChanges = Symbol('rewriteDataToFileWithChanges');
const _generateID = Symbol('generateID');
const _makeChangesToRecord = Symbol('makeChangesToRecord');
const _getLastRecordItem = Symbol('_getLastRecordItem');


class CSVHandler {

    constructor(filePath) {
        // private property of the class
        this[_FILE_PATH] = filePath;
    }

    // these functions are used to change the
    // file from a csv file to a txt file or vice versa
    // these are private methods
    [_csvToTxt]() {
        return path.basename(this[_FILE_PATH], '.txt');
    }

    [_txtToCsv]() {
        return path.basename(this[_FILE_PATH], '.csv');
    }

    // will put a txt file that is in a csv format into a 2d array and return it
    [_csvToArray]() {
        let file = fs.readFileSync(this[_FILE_PATH], 'utf8');
        let recordsArray = file.split("\n");
        let csvInArray = [];

        // there is an empty array element at the end of each record
        // in this loop every record has the empty element removed and the rest of the
        // array gets put back into the csvInArray array.
        for (let i = 0; i < recordsArray.length; i++) {
            let singleRecordArray = recordsArray[i].split(",");
            singleRecordArray.pop(); // gets rid of the empty last element in the
            array
            csvInArray.push(singleRecordArray);
        }
        return csvInArray;
    }


    // returns all records stored in the file
    getAllRecords () {
        let arrayOfRecords = this[_csvToArray]();
        arrayOfRecords.pop();
        return arrayOfRecords;
    }
```

```
73
74
75          // used to find a record by it's id
76          // requires only the id to retrieve the record
77          selectRecordByID(id) {
78              let array = this[_csvToArray]();
79              array.pop();
80
81              let recordFoundByID = Search.binarySearch2D(array, [0], id); // Stores the
                record found.
82
83              // function done on recordFoundByID in the return is done to flatten the array
84              // Output would be a multi dimensional array otherwise containing just one
                record.
85              return [].concat.apply([], recordFoundByID)
86          }
87
88          // searches for a record to return based off of a search value
89          // will check all columns entered to see if there is a match in a any of them
90          // columnsToSearch parameter should be an array of numbers that represent the
            index's of the columns to be searched
91          // the search value should be a sting
92          selectRecordByField(columnsToSearch, searchValue) {
93              let arrayOfRecords = this[_csvToArray]();
94
95              arrayOfRecords.splice(arrayOfRecords.length - 1, 1);
96              return Search.binarySearch2D(arrayOfRecords, columnsToSearch, searchValue)
97          }
98
99          // will return a number for how many matches were found in that column for the
            searchValue
100         // columnsToSearch parameter should be an array of numbers that represent the
            index's of the columns
101         // to be searched. The search value should be a sting.
102         countMatches (columnsToSearched, searchValue) {
103             return this.selectRecordByField(columnsToSearched, searchValue).length;
104         }
105
106         // returns only the records found that match the searchValue
107         // unlike the selectRecordByField which returns an array of objects
108         getRecordsOnlyByField (columnsToSearch, searchValue) {
109             let arrayOfRecordsAndIndexsObj = this.selectRecordByField(columnsToSearch,
                searchValue);
110             let arrayORecordsOnly = [];
111             for (let i = 0; i < arrayOfRecordsAndIndexsObj.length; i++) {
112                 arrayORecordsOnly.push(arrayOfRecordsAndIndexsObj[i].record)
113             }
114             return arrayORecordsOnly;
115         }
116
117
118         // The generateUniqueID variable will make this a polymorphic method as it will
            run differently dependent on
119         // the argument that gets passed for it.
120         // the data argument should be an array of all the data contained in a record
121         addRecord(data, generateUniqueID, noTimeInTimeStamp) { // no time in time stamp
            allows the user to make a time stamp with no time elements in it
122
123             let arrayOfRecords = this[_csvToArray]();
124             let arrayHolder = [];
125             let currentDateTime;
126
127
128             // if there is no argument passed in the generateUniqueID parameter
129             // the it's default value will be set to undefined which would be
                interpreted as
130             // false by the if statements. Because this method is more likely to be used
131             // to create a unique id when a record is being added, it makes sense for it
                to be true when
132             // the user is creating a record and leaves out that argument. Also done to
                aid readability as
133             // I could have named it doNotGenerateId but parameters aren't typically
                named like that
```

```
134
135             if (generateUniqueID === undefined) {
136                 generateUniqueID = true;
137             }
138
139             if (generateUniqueID) {
140                 // generate the unique id, has to be declared down here so it does not
141                 // interfere with the UniqueIdGenerator class when it accesses the
142                 // addRecord method
143                 const ID = this[_generateID](); // generates a unique id value
144
145                 if (noTimeInTimeStamp === true) { // sets appropriate date time stamp
                    dependent on argument passed in
146                     currentDateTime = DateHandler.getCurrentDate();
147                 } else {
148                     currentDateTime = DateHandler.getCurrentDateTime();
149                 }
150
151
152                 let recordWithDateStampAndID = [ID, currentDateTime];
153                 // add record
154                 data.forEach((field) => {
155                     recordWithDateStampAndID.push(field);
156                 });
157                 // console.log(recordWithDateStampAndID);
158                 // removes the empty last element in the array caused by the empty new
                    line
159                 // .pop() resulted in the entire array got cleared for some reason so
                    this is the
160                 // work around
161                 arrayOfRecords.splice(arrayOfRecords.length - 1);
162                 arrayOfRecords.push(recordWithDateStampAndID);
163                 // console.log( arrayOfRecords);
164                 this[_rewriteDataToFileWithChanges](arrayOfRecords);
165
166             }
167
168             // used by the unique id generator private method
169             else {
170                 arrayOfRecords.pop();
171                 arrayHolder.push(data);
172                 // takes the 2D array 'arrayOfRecords' and appends the new record to the
                    end of it
173                 arrayOfRecords.push(arrayHolder);
174                 // rewrite the array of records with the appended record to the file
175                 this[_rewriteDataToFileWithChanges](arrayOfRecords);
176             }
177         }
178
179
180         deleteRecord(id) {
181             let arrayOfRecords = Sort.bubbleSort2D(this[_csvToArray](), 0);
182             arrayOfRecords.splice(arrayOfRecords.length - 1);
183             //exception handling so program doesn't crash if the delete was not possible
184             try{
185                 arrayOfRecords.splice(Search.binarySearch2D(arrayOfRecords, [0],
                    id)[0].index, 1); // deletes
186             }
187             catch(e) {
188                 console.error(`record was not able to be deleted: check record with that
                    id (${id}) exists\n`);
189                 console.error(e)
190             }
191             // arrayOfRecords is parsed back into the file without
192             // the record with the id passed as argument
193             // unless the record was not able to be deleted, then all
194             // records will get parsed back into the original csv file
195             this[_rewriteDataToFileWithChanges](arrayOfRecords);
196         }
197
198
199
200         // arrayOfEdits format: [["col that needs to be edited", "new data"], ... ,
```

```
        ["col that needs to be edited", "new data"]]
201     // arrayOfEdits = [["1", "james"],["2", "doug"]];
202     // example of method calling code: users.editRecord("32", [["1","james"], ["2",
        "jones"]]);
203     // id should be of string type
204     editRecord(id, arrayOfEdits) {
205         // file gets turned to 2d array ad then sorted by the id column
206         let arrayOfRecords = Sort.bubbleSort2D(this[_csvToArray](), 0);
207         arrayOfRecords.splice(arrayOfRecords.length - 1); // removes last element in
            array because it is empty
208         // find record by id and returns a javascript object: {record: [record
            data], index: index record is in 2d array}
209         // the [0] is at the end because it is returned contained in an array
210         let recordObject = Search.binarySearch2D(arrayOfRecords, [0], id)[0];
211         // puts the object values into variables
212         let index = recordObject.index;
213         let record = recordObject.record;
214
215         // calls the makeChangesToRecord private method and then
216         // stores that new record in the place of the old record
217         arrayOfRecords[index] = this[_makeChangesToRecord](record, arrayOfEdits);
218         // rewrites the 2D array to a the csv file.
219         this[_rewriteDataToFileWithChanges](arrayOfRecords);
220     }
221
222
223     // method that will return the last record in the file
224     [_getLastRecordItem]() {
225         let arrayOfRecords = this[_csvToArray]();
226         let arrLength = arrayOfRecords.length;
227         // to access last record, the element has to be length - 2 because
228         // arrays start from 0 and then the other -1 comes form the fact that it
            counts the new line.
229         return arrayOfRecords[arrLength - 2];
230     }
231
232
233     // takes a 2d array as a argument, will write the 2D Array to the file.
234     // works by deleting all data contained in the file and then writing all the
        data in the
235     // 2D array to the text file.
236     [_rewriteDataToFileWithChanges](array2D) {
237
238         // remove all data stored in the file
239         // call back to carry out further tasks after the data has been erased
240         fs.writeFile(this[_FILE_PATH], '', (err) => {
241             if (err) {
242                 console.log(err);
243
244             }
245             else {
246                 // rewrite data to file
247                 let record = [];
248                 //iterates through the length of the array (number of records in the
                    array)
249                 for (let i = 0; i < array2D.length; i++) {
250                     record = array2D[i].join(","); // joins the record  into a
                        string separated with a comma
251                     // writes a record to the file in format of:
252                     // 'comma separated record data' + "," + new line
253                     // call back to output that the data has been re-written
                        successfully or an error
254                     // message outlining the error
255                     fs.appendFileSync(this[_FILE_PATH], `${record},\n`);
256                 }
257             }
258         });
259
260     }
261
262     deleteAllRecords () {
263         fs.writeFile(this[_FILE_PATH], '', (err) => {
264             console.log(err);
```

```
265             });
266         }
267 
268         // column to sort by should be a number of the index of the field that will get
            sorted
269         // ascending should be set to true or false depending you want the 2d array to
            be in
270         // ascending order: true or descending order: false
271         // if the ascending parameter has no value passed as argument, it will sort it
            in ascending order
272         // as that is how I defined the method in the sort class.
273         sortBy (columnToSortBy, ascending) {
274              let sortedArray = Sort.bubbleSort2D(this[_csvToArray](), columnToSortBy,
                 ascending);
275              sortedArray.pop(); // gets rid of the empty element at the end of the array
276              return sortedArray;
277         }
278 
279 
280 
281 
282 
283         [_generateID]() {
284             // could use this but there is no point of taking up extra server space so I
                am using a trusted unique Id generator package
285             // let idFile = new CSVHandler(__dirname + "/dataStoreFiles/IDs.csv"); //
                create an instance of the CSVHandler class
286             //         // let lastID = idFile.getLastRecordItem()[0]; // get last id
                number stored in the csv file
287             //         // let newID = parseInt(lastID) + 1; // set id to that value + 1
288             //         // // if the new id comes back as not a number due to there being
                no number returned from the file
289             //         // // then make new id = 1;
290             //         // if (isNaN(newID)) {
291             //         //     newID = 1;
292             //         // } // only time this module wont work is when the IDs.csv file
                has no lines in it at all
293             //         // // to fix - press the return key in the csv file to add a
                blank line
294             //         // idFile.addRecord(newID, false); // add id just generated to
                the csv file
295             //         // return newID.toString();
296             return uniqueID();
297         };
298 
299 
300         // this private method makes changes to a single record
301         // record should be an array of a single record
302         // arrayOfEdits should be a 2D array containing arrays of changes
303         // arrayOfEdits format: [["col that needs to be edited", "new data"], ... ,
            ["col that needs to be edited", "new data"]]
304         // example data: record = ["101","kieran","williams"]; edits = [["1",
            "james"],["2", "doug"]];
305 
306         [_makeChangesToRecord](record, arrayOfEdits) {
307             //initialize the variables
308             let col = "";
309             let newData = "";
310 
311             arrayOfEdits.forEach((edit) => {
312                 col = parseInt(edit[0]);
313                 // makes sure that the column is not 0 because if it is, it will be
                    possible
314                 // to change the id field of a record which is something that should not
                    ever happen
315                 if (col !== 0) {
316                     newData = edit[1];
317                     record[col] = newData;
318                 }
319             });
320             return record;
321         }
322 
```

```
323
324   }
325
326   module.exports = CSVHandler;
327
```

# DateHandler.js

Kieran Marcus Williams

```
const moment = require('moment');

const _ddmmyyyyToStandardDateTimeCheck = Symbol('ddmmyyyyToStandardDateTimeCheck');

class DateHandler {

    // creates a time stamp from a date and a time
    // returns a time stamp that moment.js or the
    // Date class js has can work with. arguments to be type string
    // date => dd/mm/yyyy
    // time is optional, and the seconds value is also optional
    // time => hh:mm:ss || hh:mm
    static createTimeStamp (date, time) {
        let day, month, year, hour, minute, second; // declares variables
        if (time === undefined) {time = "00:00:00"} // if time is passed as an argument, time is set to "00:00:00"
        if (time.length === 5) {time = `${time}:00`} // if time is only hh:mm => the ss is set to 00
        date = date.split("/"); // splits the date string up into a string array ["dd","mm","yyyy"]
        time = time.split(":"); // splits the time string up into a string array ["hh","mm","ss"]
        day = date[0];
        month = date[1];
        year = date[2];
        hour = time[0];
        minute = time[1];
        second = time[2];
        return `${year}-${month}-${day}T${hour}:${minute}:${second}+00:00`;
    }

    // uses the moment library to get the current date and time
    // outputs a string in the standard date time stamp
    static getCurrentDateTime () {
        return moment().format();
    }

    // uses the moment library to get the current date and time
    // outputs a string in the standard date time stamp and then
    // turns the dateTimeStamp into the standard date time stamp but with
    // only the date information and the time data will all be set to 0
    static getCurrentDate() {
        // returns the result of the current time stamp turned into dd/mm/yyyy, turned into
        // the standard date time stamp but with not time information
        return
        this.createTimeStamp(this.standardDateTo_Date_DDMMYYYY(this.getCurrentDateTime()));
    }

    // pass a date time stamp as argument format: YYYY-MM-DD'T'HH:MM:SS+MS:mS
    // eg: '2019-02-02T22:52:49+00:00'
    // will output the date as dd/mm/yyyy
    static standardDateTo_Date_DDMMYYYY(dateTimeStamp){
        // could use moment(dateTimeStamp).format("DD/MM/YYYY");
        // but will use my own algorithm to extract the date instead

        // splits argument into an array containing the date at index 0 &  the time at index 1
        // [0] at the end gets us just the date element, split the date up into an array ["yyyy","mm","dd"]
        // could use .reverse and then .join("/") which would give me "dd/mm/yyyy"
        // but am writing them process myself for algorithm marks
        let date = dateTimeStamp.split("T")[0].split("-");

        // self invoking function that reverses the date and then stores the result in
        // the dateReversed variable
        let dateReversed = (() => {
            let temp = "";
            let j = date.length -1;
            for (let i = 0; i < date.length / 2; i++, j--) {
                temp = date[i];
                date[i] = date[j];
```

```
                date[j] = temp;
            }
            return date;
        })();

        return dateReversed.join("/");
    }

    // pass a date time stamp as argument format: YYYY-MM-DD'T'HH:MM:SS+MS:mS
    // eg: '2019-02-02T22:52:49+00:00'
    // will output the time as hh:mm:ss if the withSeconds argument is == true
    // if it is not then the time will be outputted as hh:mm
    static standardDateTo_Time_HHMMSS(dateTimeStamp, withSeconds){

        // splits argument into an array containing the date at index 0 &  the time at index 1
        // [1] at the end gets us just the time element. split the time element up into an array of characters,
        // the array of characters will be: ["h","h",":","m","m",":","s","s","+","m","s",":","m","s"]
        // in terms of format and length. if withSeconds option parameter is set to true, then all the characters
        // in the array after the 7th index will get removed. if withSeconds is not true, elements after 4th index will get removed
        let time = dateTimeStamp.split("T")[1].split("");
        let indexToRemoveFrom;
        if (withSeconds !== true) {indexToRemoveFrom = 5 }
        else { indexToRemoveFrom = 8 }
        //self invoking function that truncates the array of unwanted data will be returned
        return (() => {
            let outputArr = []; // declare output array
            // append elements from time to outputArray up to the indexToRemoveFrom
            for (let i = 0; i < indexToRemoveFrom; i++){ outputArr.push(time[i]) }
            return outputArr.join(""); //joins the array of characters together into a string
        })();
    }

    // checks if the dateToCheck is equal to the current date
    // dateToCheck should be type string in either dd/mm/yyyy format or standard date time format
    static isToday (dateToCheck) {
        // gets the users current date in a time stamp. Then, that time stamp
        // gets turned into a dd/mm/yyyy format
        let currentDate =
        this.standardDateTo_Date_DDMMYYYY(this.getCurrentDateTime());
        // checks to see if the date is in standard time or dd/mm/yyyy format
        // if it is in standard time format then the length of the string will not be = to 10
        // in that case the dateToCheck is converted into dd/mm/yyyy format
        if (dateToCheck.length !== 10) { dateToCheck =
        this.standardDateTo_Date_DDMMYYYY(dateToCheck) }
        // compares the 2 strings, if the dates are equal true is returned. Otherwise, false is returned
        return (dateToCheck === currentDate);
    }




    // checks if the dateToCheck's month or year is equal to the current month or year
    // dateToCheck should be type string in either dd/mm/yyyy format or standard date time format
    // monthOrYear option should be set to 'month' or 'year'
    static isThisMonthOrYear (dateToCheck, monthOrYearOption) {
        // gets the users current date in a time stamp. Then, that time stamp
        // gets turned into a dd/mm/yyyy format
        let currentDate =
        this.standardDateTo_Date_DDMMYYYY(this.getCurrentDateTime());
        // checks to see if the date is in standard time or dd/mm/yyyy format
        // if it is in standard time format then the length of the string will not be = to 10
```

```
123             // in that case the dateToCheck is converted into dd/mm/yyyy format
124             dateToCheck = this[_ddmmyyyyToStandardDateTimeCheck] (dateToCheck);
125 
126             // splits the datetime by "T" and then take index 0 of the array to get
                yyyy-mm-dd
127             // split the yyyy-mm-dd by "-" to get array of ["yyyy","mm","dd"]
128             dateToCheck = dateToCheck.split("T")[0].split("-");
129             currentDate = currentDate.split("T")[0].split("-");
130 
131             // if the the option 'year' is not passed in as an argument the
132             // months will be compared by default
133             if (monthOrYearOption !== 'year') {
134                 // returns the evaluation result of the equality between the month data
                    from both arrays
135                 return dateToCheck[1] === currentDate[1];
136             }
137             else {
138                 // returns the evaluation result of the equality between the year data
                    from both arrays
139                 return dateToCheck[2] === currentDate[2];
140             }
141 
142         }
143 
144 
145 
146 
147         // can pass in date as standard date time format or as dd/mm/yyyy as type string
148         // objectContainingTheAddingToMake should be type object can be a negative
            number to subtract days
149         // objectContainingTheAddingToMake: examples: {days : 6} or { months: 9} or
            {years : 9}
150         // can even do any combination: examples: {days : 7, months : 9} or {days : 8,
            months: 9, years : 2}
151         static addToDate (date, objectContainingTheAddingToMake) {
152             // if date is in dd/mm/yyyy format it gets changed into the standard date
                time format the moment.js library can work with
153             date = this[_ddmmyyyyToStandardDateTimeCheck] (date);
154             // uses the moment library to make relevant changes to the date. The result
                then gets formatted to the standard
155             // date format by using the .format()
156             return moment(date).add(objectContainingTheAddingToMake).format();
157         }
158 
159 
160 
161         // can pass in date as standard date time format or as dd/mm/yyyy as type string
162         // objectContainingTheAddingToMake should be type object can be a negative
            number to subtract days
163         // objectContainingTheAddingToMake: examples: {days : 6} or { months: 9} or
            {years : 9}
164         // can even do any combination: examples: {days : 7, months : 9} or {days : 8,
            months: 9, years : 2}
165         static subtractFromDate (date, objectContainingTheSubtractionsToMake) {
166             // if date is in dd/mm/yyyy format it gets changed into the standard date
                time format the moment.js library can work with
167             date = this[_ddmmyyyyToStandardDateTimeCheck] (date);
168             // uses the moment library to make relevant changes to the date. The result
                then gets formatted to the standard
169             // date format by using the .format()
170             return moment(date).subtract(objectContainingTheSubtractionsToMake).format();
171         }
172 
173 
174         // can pass in date as standard date time format or as dd/mm/yyyy as type string
175         static dateToWordedMonth (date) {
176 
177             // if date is in dd/mm/yyyy format it gets changed into the standard date
                time format the moment.js library can work with
178             date = this[_ddmmyyyyToStandardDateTimeCheck] (date);
179             return moment(date).format("MMMM");
180         }
181 
```

```
182
183
184
185         // method checks to see if the dateToBeChecked is in the past or in the future.
            dateToBeChecked should type string
186         // The argument should be in the standard date format for greater
187         // accuracy as time will also be accounted for then but dd/mm/yyyy format will
            work
188         // the method will return true if the date has passed and false if the date is
            in the future
189         static hasDatePassed (dateToBeChecked) {
190             let currentDateTime = this.getCurrentDateTime();
191             // if date is in dd/mm/yyyy format it gets changed into the standard date
                time format the moment.js library can work with
192             dateToBeChecked = this[_ddmmyyyyToStandardDateTimeCheck] (dateToBeChecked);
193
194             // returns the evaluation of the dates
195             return currentDateTime > dateToBeChecked;
196         }
197
198
199         // private method to check if a date is dd/mm/yyyy or standard date time stamp
200         // if the date is dd/mm/yyyy format then the date gets changed to the equivalent
            in
201         // the standard date time stamp
202         static [_ddmmyyyyToStandardDateTimeCheck] (date) {
203             if (date.length === 10) { date = DateHandler.createTimeStamp(date) }
204             return date;
205         }
206     }
207
208
209     module.exports = DateHandler;
210
```

# Encryption.js

Kieran Marcus Williams

```
function Encryption(){}

const _generateKey = Symbol('generateKey');

// change all characters from input and key to character codes to store in separate arrays
// xor all input characters by key characters, store xor characters in new array

Encryption[_generateKey] = (key, inputLength) => {
    let keyArray; // used to store key / password as an array of characters
    let difference;
    while (key.length <= inputLength) { // loop to make the key longer or equal to the length of the input text
        key = key + key + "p";
    }
    keyArray = key.split("");

    // stores how much longer the length of the key is to the input
    difference = keyArray.length - inputLength;

    // loop to get rid of last element in the array each time to make length of keyArray == inputArray
    for (let i = 0; i < difference; i++){
        keyArray.pop();
    }
    return keyArray;
};


Encryption.xorEncrypt = (input, key) => {
    let inputArray = input.split(""); // puts input into an array of characters
    let keyArray = []; // used to store key / password as an array of characters
    let xorValues = [];
    keyArray = Encryption[_generateKey](key, input.length);

    // replace all characters in their element with a character code for inputArray and keyArray
    // preform xor on the elements, storing the answers in the xorValues Array
    for (let i = 0; i < keyArray.length; i++) {
        keyArray[i] = keyArray[i].charCodeAt(0).toString();
        inputArray[i] = inputArray[i].charCodeAt(0).toString();
        xorValues[i] = keyArray[i] ^ inputArray[i]; // preforms the xor on both characters from the key and the input
    }

    //makes the last number of the coma separated string, the length of original message
    // and puts double quotes around the string so that it can be saved to the
    // csv file without causing disruption to the rest of the file
    return  xorValues.join("/") + "/" + (input.length).toString() ;
};


// Static method that can be called anywhere in my code
Encryption.xorDecrypt = (input, key) => { // input needs to be comma separated integers as one string.
    let inputArray = input.split("/"); // splits the input array up into an array of the numbers
    let keyArray = [];
    let xorValues = [];
    let decryptedTextArray = [];

    // gets length of input before it was encrypted and stores it in a constant
    const INPUT_LENGTH_BEFORE_ENCRYPTION = inputArray[(inputArray.length-1)];
    inputArray.pop(); // gets rid of the length of original text element
    keyArray = Encryption[_generateKey](key, INPUT_LENGTH_BEFORE_ENCRYPTION);

    for (let i = 0; i < keyArray.length; i++) {
        keyArray[i] = keyArray[i].charCodeAt(0).toString();
        xorValues[i] = keyArray[i] ^ inputArray[i]; // preforms the xor on both characters from the key and the input

        // gets the character code from the xor result, turns it into a string and
```

```
            then puts it into the decryptedTextArray
            decryptedTextArray[i] = String.fromCharCode(xorValues[i]);

        }

        return (decryptedTextArray.join("")); // decrypted text gets returned


    };

    module.exports = Encryption;


    // // Test code
    //
    // let text  = "hey";
    // let password = "wowww";
    // let fakePass = "wowx";
    //
    //
    // let encryptedText = Encryption.xorEncrypt(text, password);
    // let decryptedText = Encryption.xorDecrypt(encryptedText, fakePass);
    //
    // console.log(`text: ${text} \npassword: ${password} \nencrypted text:
    ${encryptedText} \ndecrypted text: ${decryptedText}`);
```

# GenerateDataPoints.js

Kieran Marcus Williams

```
const CSVHandler = require('./CSVHandler');
const DateHandler = require('./DateHandler');

const _getDataPointsFromLast7Days = Symbol("getDataPointsFromLast7Days");
const _fileToLookThrough = Symbol("fileToLookThrough");
const _clientsGainedCSVHandlerObject = Symbol("clientsGainedCSVHandlerObject");
const _clientsLostCSVHandlerObject = Symbol("clientsLostCSVHandlerObject");
const _leadsLostCSVHandlerObject = Symbol("leadsLostCSVHandlerObject");

class GenerateDataPoints{

    constructor (fileToLookThrough, clientsGainedCSVHandlerObject,
    clientsLostCSVHandlerObject, leadsLostCSVHandlerObject) {
        this[_fileToLookThrough] = fileToLookThrough;
        this[_clientsGainedCSVHandlerObject] = clientsGainedCSVHandlerObject;
        this[_clientsLostCSVHandlerObject] = clientsLostCSVHandlerObject;
        this[_leadsLostCSVHandlerObject] = leadsLostCSVHandlerObject;
    }

    // will get the data points and the labels for the past 7 days over however many
    weeks and return the labels as dd/mm/yyyy
    // and will also output the data points which will be the number of matches
    found on the date being checked
    [_getDataPointsFromLast7Days] (numOfWeeks) {

        let clientsGained = this[_clientsGainedCSVHandlerObject];
        let clientsLost = this[_clientsLostCSVHandlerObject];
        let leadsLost = this[_leadsLostCSVHandlerObject];

        let dataPointsArray = [];
        let datesArray = []; // will be used to store the dates so that they can be
        used as labels for the graph

        // uses the getCurrentDate() method instead of getCurrentDateTime() method,
        // as the dates stored in the database wont contain time information. This
        will allow for an easy check on the dates
        const DATE = DateHandler.getCurrentDate(); // date is a constant value as it
        will not get changed at all
        let date = DATE; // date is used as a temp variable that will get
        manipulated in the loop
        let noDays = numOfWeeks * 7;
        let numberOfDateMatches = 0;
        // variables below used to get the date into dd/mm/yyyy for the labels
        let dateDDMMYY_to_standardDateTime;
        let standardDateToDDMMYYY;

        // console.log(`current date: ${DATE}`); // will output the current date for
        debugging

        for (let i = noDays - 1; i >= 0; i-- ){
            date = DateHandler.subtractFromDate(DATE, {days: i});
            // console.log(`${i} days ago: ${date}`); // can be used to check the
            dates I am getting back


            switch (this[_fileToLookThrough]){
                case "clientsGained":
                    numberOfDateMatches = clientsGained.countMatches([1],date);
                    break;
                case "clientsLost":
                    numberOfDateMatches = clientsLost.countMatches([1],date);
                    break;
                case "leadsLost":
                    numberOfDateMatches = leadsLost.countMatches([1],date);
                    break;
                default:
                    console.error("ensure the fileToLook through constructor method
                    argument is: 'clientsGained', 'clientsLost', or 'leadsLost' \n" +
                        "You entered: " + this[_fileToLookThrough]);
                    break;
            }


```

```
64              // done to get the date into dd/mm/yyyy for the labels
65              dateDDMMYY_to_standardDateTime =
                DateHandler.standardDateTo_Date_DDMMYYYY(date);
66              standardDateToDDMMYYY =
                DateHandler.standardDateTo_Date_DDMMYYYY(DateHandler.createTimeStamp(dateD
                DMMYY_to_standardDateTime));
67
68              datesArray.push(standardDateToDDMMYYY); // appends the dd/mm/yyyy labels
                to the datesArray
69
70
                ////////////////////////////////////////////////////////////////////////
                ////////////////////////////////////////////////////
71
72              dataPointsArray.push(numberOfDateMatches); // puts the number of matches
                per date into an array
73          }
74          // method returns an object containing the labels and the corresponding
            dataPoints
75          return {labels: datesArray, dataPoints: dataPointsArray};
76      }
77
78
79
80
81      // returns the last 14 days of data points for the files: clientsGained,
        clientsLost leadsLost
82      // them above strings are the accepted parameter values in the string type
83       getPast14DaysOfDataPoints() {
84
85          return this[_getDataPointsFromLast7Days] (2, this[_fileToLookThrough]);
86      }
87
88
89
90
91      // returns data points for 12 weeks of matches from the files: clientsGained,
        clientsLost leadsLost
92      // them above strings are the accepted parameter values in the string type
93       getDataPointsForLast12Weeks () {
94          // get the past 12 weeks of data as individual days and then go through the
            array
95          // add 7 of the match numbers at a time append that value to a new array for
            data points
96          // and the first date of the 7 dates to a different array for labels
97
98          let allLabels = this[_getDataPointsFromLast7Days](12).labels;
99          let allDataPoints = this[_getDataPointsFromLast7Days](12).dataPoints;
100         let usedLabels = [];
101         let usedDataPoints = [];
102         let totalMatchesInTheWeek;
103
104         // goes through all 12 weeks
105         for (let i = 0; i < 12; i++){
106
107             totalMatchesInTheWeek = 0; // initializes the variable / resets the
                variable back to 0
108
109             usedLabels.push(allLabels[i*7]); // appends the 1st date of each week
                into an array for the labels
110
111             // goes through all 7 days in the week
112             for (let j = 0; j < 7; j++) {
113                 totalMatchesInTheWeek += allDataPoints[j]; // gets total matches for
                    the week
114             }
115
116             usedDataPoints.push(totalMatchesInTheWeek); // appends the average
                number of matches within the week to an array
117             allDataPoints.splice(0,7); // deletes the first week (7 days)/ 7
                elements of the array as they are no longer needed
118
119         }
```

```
120             return {labels: usedLabels, dataPoints: usedDataPoints}
121         }
122
123
124
125         getDataPointsForLast12Months () {
126
127             // get the dates of the past 12 months
128              // turn them dates into yyyy-mm and store in an array
129             // count matches in the 2nd index of the files for each of the 12 dates
130
131              let clientsGained = this[_clientsGainedCSVHandlerObject];
132              let clientsLost = this[_clientsLostCSVHandlerObject];
133              let leadsLost = this[_leadsLostCSVHandlerObject];
134
135              const CURRENT_DATE = DateHandler.getCurrentDateTime(); // gets the current
                 time
136              let date = CURRENT_DATE; // will be the variable getting changed
137              let numberOfDateMatches;
138              let dataPointsArray = [];
139              let labelsArray = [];
140
141
142              for (let i = 11; i >= 0; i--) {
143
144                  date = DateHandler.subtractFromDate(date, {month: i}); // subtracts i
                     months from the current date
145
146                  labelsArray.push(DateHandler.dateToWordedMonth(date)); // appends the
                     month as it's name to the csv file
147
148                  date = date.split(""); // puts the date into an array of characters
149                  date.splice(7,18); // gets rid of the last 18 characters
150                  date = date.join("");
151
152
153                  switch (this[_fileToLookThrough]){
154                      case "clientsGained":
155                          numberOfDateMatches = clientsGained.countMatches([2],date);
156                          break;
157                      case "clientsLost":
158                          numberOfDateMatches = clientsLost.countMatches([2],date);
159                          break;
160                      case "leadsLost":
161                          numberOfDateMatches = leadsLost.countMatches([2],date);
162                          break;
163                      default:
164                          console.error("ensure the fileToLook through constructor method
                             argument is: 'clientsGained', 'clientsLost', or 'leadsLost' \n" +
165                              "You entered: " + this[_fileToLookThrough]); // outputs an
                                 error message of what is most likely the cause of any
                                 problems in this case
166                          break;
167                  }
168
169                  dataPointsArray.push(numberOfDateMatches); // appends the number of
                     matches found for that month to the data points array
170
171
172                  date = CURRENT_DATE; // sets the date back to the current date so that
                     it can be
173              }
174
175              return {labels: labelsArray, dataPoints: dataPointsArray}
176
177         }
178
179     }
180
181     module.exports = GenerateDataPoints;
182
183
```

# Search.js

Kieran Marcus Williams

```
const Sort = require("./Sort"); // allows the use of sort methods in this file

function Search(){}


// creates private method
const _binarySearch2DFinder = Symbol('binarySearch2DFinder');
const _checkBoundsForMatches = Symbol('checkBoundsForMatches');



// searches for a value and returns the index the element was found at
// or returns -1 if the search value was not found in the array
// ONLY WORKS ON SORTED ARRAYS
Search.binarySearch = (array, searchValue) => {
    // makes sure that the array is sorted before the search is done
    array = Sort.bubbleSort(array);
    let first = 0;
    let last = array.length;
    let middle = Math.floor((first + last)/2);
    let found = false;
    while((!found) || (first <= last)){
        if(array[middle] === searchValue){
            found = true;
            return middle;
        }
        if(searchValue < array[middle]){
            last = middle - 1;
            middle = Math.floor((first + last)/2);
        }
        if(searchValue > array[middle]){
            first = middle + 1;
            middle = Math.floor((first + last)/2);
        }
    }
    return -1;
};

Search[_checkBoundsForMatches] = (array, middle, col, searchValue) => {
    let matchAtBoundsFoundFlag = true; // set to true so that the loop will start
    let recordsFound = [];
    let upperBounds = middle;
    let lowerBounds = middle;

// the algorithm works in the binary search as any other matches will be found either side of the initial match
// due to a binary search only working on sorted arrays which will result in the matches in conjunction in the array

// loop through upper values
    while (matchAtBoundsFoundFlag === true){

        // set to false so that if no matches are found on the upper bounds from the first match
        // the loop will end.
        matchAtBoundsFoundFlag = false;

        // checks that if the upperBounds was to get increased by one,
        // the program will not crash due to the record not being in the bounds of the array
        if ((upperBounds + 1) > (array.length - 1)) {
            break; // terminate the loop
        }

        if (array[upperBounds + 1][col] === searchValue) { // checks to see if the array element above is a match to the search value
            matchAtBoundsFoundFlag = true; // set back to true so that the loop will run again to check for more matches.
            upperBounds++; // increase value stored in upperBounds variable by 1

            // match to the search value was found so the recordsFound
            // array gets the record and index of record in a javascript object appended to the end of it
```

```
67                recordsFound.push({ record : array[upperBounds], index : upperBounds });
68            }
69        }
70
71        matchAtBoundsFoundFlag = true; // reset the flag value so the next loop can run
72
73    // loop through lower values
74        while (matchAtBoundsFoundFlag === true){
75
76            // set to false so that if no matches are found on the lower bounds of the
              first match
77            // the loop will end.
78            matchAtBoundsFoundFlag = false;
79
80            // checks that if the lowerBounds was to get decreased by one,
81            // the program will not crash due to the record not being in the bounds of
              the array
82            if ((lowerBounds - 1) < (0)) {
83                break; // terminate the loop
84            }
85
86            if (array[lowerBounds - 1][col] === searchValue) {
87                matchAtBoundsFoundFlag = true; // set back to true so that the loop will
                  run again to check for more matches.
88                lowerBounds--; // increase value stored in lowerBounds variable by 1
89
90                // match to the search value was found so the recordsFound
91                // array gets the record and index of record in a javascript object
                  appended to the end of it
92                recordsFound.push({ record : array[lowerBounds], index : lowerBounds  });
93            }
94        }
95
96        return recordsFound;
97    };
98
99    // purpose of method is to find a match with the searchValue in a set column
100   Search[_binarySearch2DFinder] = (array, col, searchValue) => {
101       let first = 0;
102       let last = array.length;
103       let middle = Math.floor((first + last)/2);
104       let upperAndLowerBoundMatches = [];
105       let recordsFound = [];
106
107
108
109       // loop that will keep going until last > first or the matching records are found
110       // the array[middle][col] ---> middle is the row that is currently under
          inspection, col is a constant
111       while (first <= last) {
112
113           // to avoid a crash, the program will check to see if middle is greater than
              the index of the last
114           // record and that if it is a negative value. In either instance the loop
              will be terminated
115           if (middle > array.length - 1 || middle < 0){
116               break;
117           }
118
119           if(array[middle][col] === searchValue){
120               //pushes the array and an index contained in a javascript object type
121               recordsFound.push({ record : array[middle], index : middle }); // adds
                  the record into records found array
122
123               // finds any matches in the upper and lower bounds of the
124               upperAndLowerBoundMatches = Search[_checkBoundsForMatches](array,
                  middle, col, searchValue);
125
126               upperAndLowerBoundMatches.forEach((match) => {
127                   recordsFound.push(match);
128               });
129               break; // gets out of loop
130           }
```

```
131
132             // terminates the loop if the middle value is 0 as it will cause a crash
133             // crash is due to the fact that middle will turn into a negative value
134             if (middle === 0) {
135                 break;
136             }
137
138             if(searchValue < array[middle][col]){
139                 last = middle - 1;
140                 middle = Math.floor((first + last)/2);
141             }
142             if(searchValue > array[middle][col]){
143                 first = middle + 1;
144                 middle = Math.floor((first + last)/2);
145             }
146         }
147
148         if (recordsFound.length > 0){
149             return recordsFound;
150         }
151     };
152
153
154
155
156     // returns a 2D array of matching records
157     // columnsToBeSearched should be an array of integers
158     // that determines what columns are to be searched
159     Search.binarySearch2D = (array, columnsToBeSearched, searchValue) => {
160
161         let recordsFoundInAllColumnsSearched = [];
162         let recordsFoundInOneColumn = [];
163         let col = 0;
164
165         // if there is only one record in the array then there is no point in running
        any more code
166
167
168         for (let i = 0; i < columnsToBeSearched.length; i++) {  // goes through all
        columns that need to be searched
169
170             col = columnsToBeSearched[i]; // col is made equal to the i index of the
            columnsToBeSearched array
171
172             // sorts the array with regard to the column that is being searched on
173             array = Sort.bubbleSort2D(array, col);
174
175
176             recordsFoundInOneColumn = Search[_binarySearch2DFinder](array, col,
            searchValue);
177
178
179             // if statement to check that what got returned is an
180             // array of records found and not just undefined
181             if (recordsFoundInOneColumn !== undefined){
182                 // goes through every record in the recordsFoundInOneColumnArray
183                 recordsFoundInOneColumn.forEach((currentRecord) => {
184                     recordsFoundInAllColumnsSearched.push(currentRecord);
185                 });
186
187             }
188
189         }
190         return recordsFoundInAllColumnsSearched;
191     };
192
193
194     //searches for a value and returns the index the element was found at
195     //or returns -1 if the search value was not found in the array
196
197     Search.linearSearch = (array, searchValue) => {
198         let found = false;
199         let i = 0;
```

```
    let length = array.length;
    while((!found) || (i < length)){
        if(array[i] === searchValue){
            found = true;
            return i;
        } else {
            i++;
        }
    }
    return -1;
};

Search.getRecords = (ArrayOfObjectsContainingRecordsWithIndexes) => {
    let arrayOfRecords = [];
    ArrayOfObjectsContainingRecordsWithIndexes.forEach((recordWithIndex) => {
        arrayOfRecords.push(recordWithIndex.record);
    });
    return arrayOfRecords;
};

module.exports = Search;
```

# Sort.js

Kieran Marcus Williams

```
1
2
3   function Sort (){}
4
5   const _comparisonOperator = Symbol('comparisonOperator');
6
7
8   // private funtion used so that dynamic comparisons can be made
9   // params should take a < or > type string in the comparissonOp
10  // value1 and value2 should be the two values that are going to be compared.
11  Sort[_comparisonOperator] = (comparisonOp, value1, value2) => {
12      if(comparisonOp === "<"){ return value1 < value2 }
13      if(comparisonOp === ">"){ return value1 > value2 }
14  };
15
16  // these methods can be considered an imitation of static methods
17
18  // sorts the array in ascending or descending order
19  // depending on if the ascending variable is set to true
20  // or false, and returns the sorted array
21  // THE ASCENDING PARAMETER COULD BE CONSIDERED POLYMORPHISM AS THE METHODS
    FUNCTIONALITY DIFFERS DEPENDENT ON
22  // WHETHER THE ARGUMENT PASSED SETS IT TO FALSE OR IS LEFT ALONE/ SET TO TRUE
23  Sort.bubbleSort = (array, ascending) => {
24
25      if (ascending === undefined) ascending = true;
26      // if no ascending argument is passed, method sorts array in ascending order
27
28      let n = array.length;
29      let sorted = false;
30      let temp = 0;
31      let comparisonOp = "";
32
33      if (ascending) comparisonOp = ">";
34      if (!ascending) comparisonOp = "<";
35
36          while(!sorted){
37
38              sorted = true;
39              for(let i = 0; i < n; i++){
40
41                  // if(array[i] > array[i+1]) is for ascending
42
43                  // uses the comparisonOperator method to make this
44                  // method dynamic in sorting both ascending and descending
45                  if (Sort[_comparisonOperator](comparisonOp,
46                      array[i], array[i+1])) {
47
48                      temp = array[i];
49                      array[i] = array[i+1];
50                      array[i+1] = temp;
51                      sorted = false;
52                  }
53              }
54          }
55
56      return array;
57  };
58
59  // works by comparing a set lot of columns from different
60  // records against each other. Swaps the records if the condition
61  // for a swap to be made is met (condition is determined by the boolean
62  // argument "ascending")
63  // column to sort by should be type number
64  // ascending should be boolean or not entered
65  Sort.bubbleSort2D = (array2D, columnToSortBy, ascending) => {
66
67      if (ascending === undefined) ascending = true;
68      // if no ascending argument is passed, method sorts array in ascending order
69
70      let n = array2D.length;
71      let sorted = false;
72      let temp = [];
```

```
73      let comparisonOp = "";
74
75      if (ascending) comparisonOp = ">";
76      if (!ascending) comparisonOp = "<";
77
78
79      while(!sorted){
80          sorted = true;
81          for(let i = 0; i < n -1 ; i++){
82
83              // if(array[i] > array[i+1]) is for ascending
84
85              // uses the comparisonOperator method to make this
86              // method dynamic in sorting both ascending and descending
87              if (Sort[_comparisonOperator](comparisonOp,
88                  array2D[i][columnToSortBy], array2D[i+1][columnToSortBy])) {
89
90
91                  temp = array2D[i];
92                  array2D[i] = array2D[i+1];
93                  array2D[i+1] = temp;
94                  sorted = false;
95
96
97              }
98          }
99      }
100
101     return array2D;
102 };
103
104
105
106 module.exports = Sort;
107
108
109
```

# Validation.js

Kieran Marcus Williams

```
function Validation () {}

const _stringLookupCheckForLowerCase = Symbol('stringLookupCheckForLowerCase');
const _stringLookupCheckForUpperCase = Symbol('stringLookupCheckForUpperCase');
const _stringLookupCheckForNumbers = Symbol('stringLookupCheckForNumbers');
const _stringLookupCheckForSpaces = Symbol('stringLookupCheckForSpaces');

Validation.presenceCheck = (value) => {
    return !(value === undefined || value === null || /^\s*$/.test(value));
};

Validation.lengthCheck = (value, length) => {
  return value.length === length;
};

Validation.rangeCheck = (value, lower, upper, stringLengthInRange) => {
    if (typeof value === "string"){
        if (stringLengthInRange) {
            value = value.length;
        }
        else {
            value = Number(value);
        }
    }

    if (upper === undefined || upper === null) {
        upper = lower;
    }

    return (value >= lower && value <= upper );

};

// checks if the value type is number
// the stringToNumberConversionNeeded parameter should take a boolean value
// if it is set to true then the string will get converted into a number before being checked
Validation.isaNumber = (value, stringToNumberConversionNeeded) => {

    if (stringToNumberConversionNeeded === true) {
        value = Number(value);
    }

    return typeof(value) === "number" && !isNaN(value);
};


Validation.isaInt = (value) => {
    return typeof(value) === "number" && !isNaN(value) && value % 1 === 0;
};

Validation.isaReal = (value) => {
    return typeof(value) === "number" && !isNaN(value) && value % 1 !== 0;
};

Validation.isaString = (value) => {
    return typeof(value) === "string";
};

Validation.isaBoolean = (value) => {
    return typeof(value) === "boolean";
};

Validation.isaFunction = (value) => {
    return typeof(value) === "function";
};

Validation.isaObject = (value) => {
    return typeof(value) === "object";
};

Validation.isaArray = (value) => {
    return Array.isArray(value);
```

```
};

Validation.isaCharacter = (value) => {
    return typeof(value) === "string" && Validation.lengthCheck(value, 1);
};

Validation.fileIsCSV = (file) => {
    let regex = /^([a-zA-Z0-9\s_\\.\-:])+(.csv)$/;
    return (regex.test(file));
};

Validation.fileIsTXT = (file) => {
    let regex = /^([a-zA-Z0-9\s_\\.\-:])+(.txt)$/;
    return (regex.test(file.value));
};


// value should be the date entry, futureDate parameter is for a boolean argument
// if futureDate has a true value passed then it will check that the value is
// a date into the future
Validation.dateFormat = (value, futureDate) => {

    let valueArray = [];
    let val = 0;
    let day = "";
    let month = "";
    let year = "";
    let usersCurrentDate = new Date();
    let usersDay = "";
    let usersMonth = "";
    let usersYear = "";
    // an object that stores the maximum number of days
    // any month could have
    let monthAndMaxDay = {
        jan: 31,
        feb: 28 || 29,
        mar: 31,
        apr: 30,
        may: 31,
        jun: 30,
        jul: 31,
        aug: 31,
        sep: 30,
        oct: 31,
        nov: 30,
        dec: 31
    };

    // function inside the static method to check if the day value is within the
    // bounds for that month
    let monthDayChecker = () => {
        let maxDay = 0;

        // uses a switch statement to check all possible cases for the months then
        gets the max
        // day possible for that month from the monthAndMaxDay object
        // a range check is then done using the maxDay as the inclusive upper bound
        // with 1 being the inclusive lower bound
        switch (month){
            case "01":
                maxDay = monthAndMaxDay.jan;
                if (!Validation.rangeCheck(day, 1, maxDay)){
                    return false;
                }
                break;
            case "02":
                maxDay = monthAndMaxDay.feb;
                if (!Validation.rangeCheck(day, 1, maxDay)){
                    return false;
                }
                break;
            case "03":
                maxDay = monthAndMaxDay.mar;
```

```
                if (!Validation.rangeCheck(day, 1, maxDay)){
                    return false;
                }
                break;
            case "04":
                maxDay = monthAndMaxDay.apr;
                if (!Validation.rangeCheck(day, 1, maxDay)){
                    return false;
                }
                break;
            case "05":
                maxDay = monthAndMaxDay.may;
                if (!Validation.rangeCheck(day, 1, maxDay)){
                    return false;
                }
                break;
            case "06":
                maxDay = monthAndMaxDay.jun;
                if (!Validation.rangeCheck(day, 1, maxDay)){
                    return false;
                }
                break;
            case "07":
                maxDay = monthAndMaxDay.jul;
                if (!Validation.rangeCheck(day, 1, maxDay)){
                    return false;
                }
                break;
            case "08":
                maxDay = monthAndMaxDay.aug;
                if (!Validation.rangeCheck(day, 1, maxDay)){
                    return false;
                }
                break;
            case "09":
                maxDay = monthAndMaxDay.sep;
                if (!Validation.rangeCheck(day, 1, maxDay)){
                    return false;
                }
                break;
            case "10":
                maxDay = monthAndMaxDay.oct;
                if (!Validation.rangeCheck(day, 1, maxDay)){
                    return false;
                }
                break;
            case "11":
                maxDay = monthAndMaxDay.nov;
                if (!Validation.rangeCheck(day, 1, maxDay)){
                    return false;
                }
                break;
            case "12":
                maxDay = monthAndMaxDay.dec;
                if (!Validation.rangeCheck(day, 1, maxDay)){
                    return false;
                }
                break;
            default:
                return false;
        }
    };

    // checks that the input is a string and that it has a length of 10 characters
    if(Validation.isaString(value) && Validation.lengthCheck(value, 10)){

        valueArray = value.split(""); // puts the string into an array

        // loops through every element in the array so that checks can be done on
        date, month and the year part of the date
        for(let i = 0; i < valueArray.length; i++){

            // will be using this a lot so to make it more readable, the current
```

```
            array element is set to the variable val
217             val = valueArray[i];
218
219             // the 2 an 5 indexes in the array should be a '/' if they aren't false
                is returned
220             if(i === 2 || i === 5){
221                 if(val !== "/"){
222                     return false;
223                 }
224             }
225             else {
226
227                 // checks that any other character is a number value, if it isn't
                    false is returned
228                 if (!Validation.isaNumber(Number(val))){
229                     return false;
230                 }
231             }
232         }
233
234         // sets the day, month and year variables to the relevant characters
235         day = valueArray[0] + valueArray[1];
236         month = valueArray[3] + valueArray[4];
237         year = valueArray[6] + valueArray[7] + valueArray[8] + valueArray[9];
238
239         // check that the day is between 1 and 31 inclusive, return false if it is not
240         if (!Validation.rangeCheck(day, 1, 31)){
241             return false;
242         }
243
244         // check that the month is between 1 and 12 inclusive, if it is not then
            return false
245         if (!Validation.rangeCheck(month, 1, 12)){
246             return false;
247         }
248
249         // check that the day is in the range for that month
250         monthDayChecker();
251
252         if (futureDate !== true) {
253             return true;
254         }
255         else { // if the future date is needed
256
257             // set variables fot the users current system date
258             usersDay = usersCurrentDate.getDate();
259             usersMonth = usersCurrentDate.getMonth() + 1;
260             usersYear = usersCurrentDate.getFullYear();
261
262             // check that the date entered is one for the future
263             if (Number(year) > usersYear){ return true; }
264             if (Number(year) === usersYear){
265                 if (Number(month) > usersMonth){ return true; }
266                 if (Number(month) === usersMonth) {
267                     return (Number(day) >= usersDay); // returns true if the entered
                        date is in the future or is the current day
268                 }
269                 else {
270                     return false;
271                 }
272             }
273             else {
274                 return false;
275             }
276
277         }
278
279     }
280     else {
281         return false;
282     }
283 };
284
```

```
Validation.timeFormat_HHMM = (value) => {

    let valueArray = [];
    let val = 0;
    let hour, minute;

    // checks that the input is a string and that it has a length of 5 characters
    if(Validation.isaString(value) && Validation.lengthCheck(value, 5)) {

        valueArray = value.split(""); // puts the string into an array

        // loops through every element in the array so that checks can be done on
        the hour and minute parts of the time
        for (let i = 0; i < valueArray.length; i++) {

            // will be using this a lot so to make it more readable, the current
            array element is set to the variable val
            val = valueArray[i];

            // the 2 index in the array as the element should be a ':' if it is not,
            false is returned
            if (i === 2) {
                if (val !== ":") {
                    return false;
                }
            }
            else {

                // checks that any other character is a number value, if it isn't
                false is returned
                if (!Validation.isaNumber(Number(val))) {
                    return false;
                }
            }
        }

        hour = valueArray[0] + valueArray[1];
        minute = valueArray[3] + valueArray[4];

        return ((Number(hour) >= 0 && Number(hour) <= 23 && Number(minute) >= 0  &&
        Number(minute) <= 59));
    }
    else {
        return false;
    }
};

Validation.emailFormat = (value) => {
    let emailRegex =
    /^(([^<>()\[\]\\.,;:\s@"]+(\.[^<>()\[\]\\.,;:\s@"]+)*)|(".+"))@((\[[0-9]{1,3}\.[0-
    9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}])|(([a-zA-Z\-0-9]+\.)+[a-zA-Z]{2,}))$/ ;
    return emailRegex.test(value);
};


Validation.timeFormat_HHMMSS = (value) => {

    let valueArray = [];
    let val = 0;
    let hour, minute, second;

    // checks that the input is a string and that it has a length of 5 characters
    if(Validation.isaString(value) && Validation.lengthCheck(value, 8)) {

        valueArray = value.split(""); // puts the string into an array

        // loops through every element in the array so that checks can be done on
        the hour and minute parts of the time
        for (let i = 0; i < valueArray.length; i++) {

            // will be using this a lot so to make it more readable, the current
            array element is set to the variable val
            val = valueArray[i];
```

```
349
350             // the 2 index in the array as the element should be a ':' if it is not,
                false is returned
351             if (i === 2 || i === 5) {
352                 if (val !== ":") {
353                     return false;
354                 }
355             }
356             else {
357
358                 // checks that any other character is a number value, if it isn't
                    false is returned
359                 if (!Validation.isaNumber(Number(val))) {
360                     return false;
361                 }
362             }
363         }
364
365         hour = valueArray[0] + valueArray[1];
366         minute = valueArray[3] + valueArray[4];
367         second = valueArray[6] + valueArray[7];
368
369         if (hour === "00" && minute === "00"){
370
371         }
372
373         return ((Number(hour) >= 0 && Number(hour) <= 23 && Number(minute) >= 0  &&
            Number(minute) <= 59
374             && Number(second) >= 0  && Number(second) <= 59));
375     }
376     else {
377         return false;
378     }
379 };
380
381
382 // EXAMPLE OF A POLYMORPHIC METHOD
383 // value two can be an array to check valueOne against multiple values in
384 // the valueTwo argument. although valueTwo can be a single value to check equality
385 Validation.comparisonCheck = (valueOne, valueTwo) => {
386     if (typeof valueTwo === "object") {
387         let foundMatchFlag = false;
388         // iterate through the array and compare valueOne to all the values in the
            array, if one matches then return true
389         // if there's no matches return false
390         for (let i = 0; i < valueTwo.length; i ++) {
391             if (valueOne === valueTwo[i]) {
392                 foundMatchFlag = true;
393             }
394         }
395         return foundMatchFlag;
396
397     } else { // if valueTwo is not an array then check for equality on the single
        value
398         return valueOne === valueTwo;
399     }
400
401 };
402
403
404 // preforms a look up check on the string comparing every character in the argument
    string
405 // to an array of acceptable characters. true is returned if there is a match found
    in the string to the list of
406 // acceptable characters, false returned if there is no match.
407 Validation[_stringLookupCheckForLowerCase] = (stringValue) => {
408     let lowerCaseFlag = false;
409     // all possible lower case letters put into an array of characters
410     let lowerCaseArray = "abcdefghijklmnopqrstuvwxyz".split("");
411     // The stringValue argument gets split up into an array of characters
412     let stringArray = stringValue.split("");
413     let currentCharacter = '';
414
```

```
415             // check for a lower case character in the stringValue
416             // by iterating through the stringArray and checking to see if a value
                matches any value in the lowerCaseArray
417             // this is achieved through using a NESTED FOR LOOP
418             for (let i = 0; i < stringArray.length; i++) {
419                 currentCharacter = stringArray[i];
420                 for (let j = 0; j < lowerCaseArray.length; j++) {
421                     // if a value in the stringArray matches the value in the
                    lowerCaseArray, set the lowerCaseFlag to true
422                     if ( currentCharacter === lowerCaseArray[j] ) {
423                         lowerCaseFlag = true;
424                     }
425                 }
426             }
427             // if the lowerCaseFlag is still set to false after the loop, then a
                lowerCase character doesn't exist, false is returned
428         return lowerCaseFlag;
429     };
430
431
432     // preforms a look up check on the string comparing every character in the argument
        string
433     // to an array of acceptable characters. true is returned if there is a match found
        in the string to the list of
434     // acceptable characters, false returned if there is no match.
435     Validation[_stringLookupCheckForUpperCase] = (stringValue) => {
436         let upperCaseFlag = false;
437         // all possible upper case letters put into an array of characters
438         let upperCaseArray = "ABCDEFGHIJKLMNOPQRSTUVWXYZ".split("");
439         // The stringValue argument gets split up into an array of characters
440         let stringArray = stringValue.split("");
441         let currentCharacter = '';
442
443         // check for a upper case character in the stringValue
444         // by iterating through the stringArray and checking to see if a value matches
            any value in the upperCaseArray
445         // this is achieved through using a NESTED FOR LOOP
446         for (let i = 0; i < stringArray.length; i++) {
447             currentCharacter = stringArray[i];
448             for (let j = 0; j < upperCaseArray.length; j++) {
449                 // if a value in the stringArray matches the value in the
                    upperCaseArray, set the upperCaseFlag to true
450                 if ( currentCharacter === upperCaseArray[j] ) {
451                     upperCaseFlag = true;
452                 }
453             }
454         }
455         // if the upperCaseFlag is still set to false after the loop, then an upperCase
            character doesn't exist, false is returned
456         return upperCaseFlag;
457     };
458
459
460     // preforms a look up check on the string comparing every character in the argument
        string
461     // to an array of acceptable characters. true is returned if there is a match found
        in the string to the list of
462     // acceptable characters, false returned if there is no match.
463     Validation[_stringLookupCheckForNumbers] = (stringValue) => {
464         let numbersFlag = false;
465         // all possible number characters  put into an array of characters
466         let numbersArray = "1234567890".split("");
467         // The stringValue argument gets split up into an array of characters
468         let stringArray = stringValue.split("");
469         let currentCharacter = '';
470
471         // check for a number character in the stringValue
472         // by iterating through the stringArray and checking to see if a value matches
            any value in the numbersArray
473         // this is achieved through using a NESTED FOR LOOP
474         for (let i = 0; i < stringArray.length; i++) {
475             currentCharacter = stringArray[i];
476             for (let j = 0; j < numbersArray.length; j++) {
```

```
            // if a value in the stringArray matches the value in the numbersArray, set the numbersFlag to true
            if ( currentCharacter === numbersArray[j] ) {
                numbersFlag = true;
            }
        }
    }
    // if the numbersFlag is still set to false after the loop, then a number character doesn't exist, false is returned
    return numbersFlag;
};


// preforms a look up check on the string comparing every character in the argument string
// to a space character. true is returned if there is a match found in the string, false returned if there is no match.
Validation[_stringLookupCheckForSpaces] = (stringValue) => {
    let spaceFlag = false;
    // the comparison space character
    let spaceCharacter = " ";
    // The stringValue argument gets split up into an array of characters
    let stringArray = stringValue.split("");
    let currentCharacter = '';

    // check for a space character in the stringValue
    // by iterating through the stringArray and checking to see if a value matches any value in the spaceCharacter
    for (let i = 0; i < stringArray.length; i++) {
        currentCharacter = stringArray[i];
        if (currentCharacter === spaceCharacter) {
            spaceFlag = true;
        }
    }
    // if the spaceFlag is still set to false after the loop, then a space character doesn't exist, false is returned
    return spaceFlag;
};


// preforms a look up check on the string comparing every character in the argument string
// to a comma character. true is returned if there is a match found in the string, false returned if there is no match.
Validation.stringLookupCheckForCommas = (stringValue) => {
    let commaFlag = false;
    // the comparison comma character
    let commaCharacter = ",";
    // The stringValue argument gets split up into an array of characters
    let stringArray = stringValue.split("");
    let currentCharacter = '';

    // check for a number character in the stringValue
    // by iterating through the stringArray and checking to see if a value matches any value in the commaCharacter
    for (let i = 0; i < stringArray.length; i++) {
        currentCharacter = stringArray[i];
        if (currentCharacter === commaCharacter) {
            commaFlag = true;
        }
    }
    // if the commaFlag is still set to false after the loop, then a comma character doesn't exist, false is returned
    return commaFlag;
};


Validation.stringLookupCheck = (stringValue, objectOfWhatNeedsToBeInTheString) => {

    // if there is no object parameter an error message is thrown to
    // notify that an object parameter is needed
    if (objectOfWhatNeedsToBeInTheString === undefined) {
        throw "ERROR NEED TO HAVE AN OBJECT PARAMETER: "
```

```
540             + "as the second parameter, set options of:\nlowerCase, upperCase, numbers:
                " +
541             " to true \nif you want to check the string contains one of them types of
                characters ";
542         }
543
544
545         // checks to see if there is a property in the object passed as argument into
            the method
546         // called lowerCase and checks to see if it's value is set to the boolean value
            true
547         if (objectOfWhatNeedsToBeInTheString.lowerCase === true) {
548             // preforms lookup check to see if there is any lowerCase characters in the
                string
549             // if there is not then false gets returned
550             if(!Validation[_stringLookupCheckForLowerCase](stringValue)){ return false }
551         }
552
553         // checks to see if there is a property in the object passed as argument into
            the method
554         // called upperCase and checks to see if it's value is set to the boolean value
            true
555         if (objectOfWhatNeedsToBeInTheString.upperCase === true) {
556             // preforms lookup check to see if there is any upperCase characters in the
                string
557             // if there is not then false gets returned
558             if(!Validation[_stringLookupCheckForUpperCase](stringValue)){ return false }
559         }
560
561         // checks to see if there is a property in the object passed as argument into
            the method
562         // called numbers and checks to see if it's value is set to the boolean value true
563         if (objectOfWhatNeedsToBeInTheString.numbers === true) {
564             // preforms lookup check to see if there is any number characters in the
                string
565             // if there is not then false gets returned
566             if(!Validation[_stringLookupCheckForNumbers](stringValue)){ return false }
567         }
568
569         // if false hasn't been returned yet then the string must contain all criteria
570         return true;
571
572     };
573
574     // method checks that the string passed in as an argument only contains lower case
        characters, upper case characters
575     // or number characters. if a character is found in the string that is not one of
        them, then the method returns false
576     // if all characters in the string that is passed as argument are either upper,
        lower or numbers then true is returned
577     Validation.checkStringIsMadeOfOnlyLowerUpperAndNumberChars = (stringValue,
        allowSpaces) => {
578         // iterate through each character in the string checking if it is a number,
            lowercase char or an uppercase char
579         let curChar = '';
580         let charArray = stringValue.split(""); // puts the string into an array of its
            characters
581         for (let i = 0; i < charArray.length; i++) {
582
583             curChar = charArray[i];
584
585             if (allowSpaces === true) {
586                 // if all of the checks come back false, false is returned
587                 if ((!Validation[_stringLookupCheckForLowerCase](curChar))
588                     && (!Validation[_stringLookupCheckForUpperCase](curChar))
589                     && (!Validation[_stringLookupCheckForNumbers](curChar))
590                     && (!Validation[_stringLookupCheckForSpaces](curChar)))
591                 {
592                     return false;
593                 }
594             }
595
596             else {
```

```
            // if all of the checks come back false, false is returned
            if ((!Validation[_stringLookupCheckForLowerCase](curChar))
                && (!Validation[_stringLookupCheckForUpperCase](curChar))
                && (!Validation[_stringLookupCheckForNumbers](curChar)))
            {
                return false;
            }
        }

    }

    // loop has finished, if false hasn't been returned it means that all the characters
    // in the string consist of either a number, lowercase char or an uppercase char
    return true;
};




module.exports = Validation;
```

# Validation Sub Classes Directory Contents

Kieran Marcus Williams

# AddNoteValidation.js

Kieran Marcus Williams

```
const Validation = require("./Validation");


// creating "private" properties for the class by utilising the Symbol function
const _note = Symbol("note");



// creating "private" method names for the class by utilising the Symbol function
const _presenceCheck = Symbol("presenceCheck");
const _stringTypeCheck = Symbol("stringTypeCheck");
const _rangeCheck = Symbol("rangeCheck");
const _basicCharactersOnlyLookUpCheck = Symbol("basicCharactersOnlyLookUpCheck");


class DashboardValidation {

    constructor (note) {
        this[_note] = note;
    }

    // presence check gets done on all the fields
    [_presenceCheck] () {

        let message = "     ____     Please Ensure All Text Fields Have Data Entered     ____     ";


        if (!Validation.presenceCheck(this[_note])) {
            return message;
        }

        return '';
    }



    // type check gets done on all the fields to check that the value is of type string
    [_stringTypeCheck] () {

        let message = "     ____     Please Ensure All Text Fields Are Of Type String     ____     ";

        // if any of the text fields are not of type string then the message will get returned
        if (!Validation.isaString(this[_note])) {
            return message;
        }
        // if no messages have been returned, the method will return an empty string
        return '';
    }


    // range check done on
    [_rangeCheck] () {

        let message = '';

        // if text field does not contain a set of data within the string length range a message will get returned
        if (!Validation.rangeCheck(this[_note], 1, 250, true)) {
            message += "     ____     Ensure That The Subject Field Is Within Range Of 1 - 250 Characters     ____     ";
        }


        // if no messages have been returned, the method will return an empty string
        return message;
    }


    // Only allows basic characters to be accepted validation done to help
```

```
        circumvent XSS and the user typing commas
    [_basicCharactersOnlyLookUpCheck] () {

        let message = "     _____     Ensure That The Note Only Contain a-z, A-Z , And 0-9 Characters     _____     ";


        if (!Validation.checkStringIsMadeOfOnlyLowerUpperAndNumberChars(this[_note], true)){
            return message;
        }

        return '';
    }




    // the public method that will be called on the instance of the class to validate it and
    // output any necessary messages if it failed the validation
    checkFieldsAreValid() {

        let outputMessage = '';

        // appends the return message from private methods to the outputMessage variable in teh public method
        outputMessage += this[_presenceCheck]();
        outputMessage += this[_basicCharactersOnlyLookUpCheck] ();
        outputMessage += this[_stringTypeCheck]();
        outputMessage += this[_rangeCheck]();


        // if the output message is comprised of just empty strings then the return value will be true
        if (!Validation.presenceCheck(outputMessage)){
            return true;
        }

        // if there is more than an empty string in the output message, the message will be returned
        return outputMessage;
    }

}

module.exports = DashboardValidation;

```

# ContactsValidation.js

Kieran Marcus Williams

```
const Validation = require("./Validation");


// creating "private" properties for the class by utilising the Symbol function
const _businessName = Symbol("businessName");
const _firstName = Symbol("firstName");
const _lastName = Symbol("lastName");
const _email = Symbol("email");
const _telephone = Symbol("telephone");
const _city = Symbol("city");
const _postcode = Symbol("postcode");
const _address = Symbol("address");
const _status = Symbol("status");


// creating "private" method names for the class by utilising the Symbol function
const _presenceCheck = Symbol("presenceCheck");
const _lengthCheck = Symbol("lengthCheck");
const _stringTypeCheck = Symbol("stringTypeCheck");
const _numberTypeCheck = Symbol("numberTypeCheck");
const _rangeCheck = Symbol("rangeCheck");
const _emailFormatCheck = Symbol("emailFormatCheck");
const _lookUpCheckForStatus = Symbol("lookUpCheckForStatus");
const _basicCharactersOnlyLookUpCheck = Symbol("basicCharactersOnlyLookUpCheck");


class ContactsValidation {

    constructor (businessName, firstName, lastName, email, telephone, city,
    postcode, address, status) {
        this[_businessName] = businessName;
        this[_firstName] = firstName;
        this[_lastName] = lastName;
        this[_email] = email;
        this[_telephone] = telephone;
        this[_city] = city;
        this[_postcode]= postcode;
        this[_address]= address;
        this[_status] = status;
    }

    // presence check gets done on all the fields
    [_presenceCheck] () {

        let message = "     ____     Please Ensure All Text Fields Have Data
        Entered     ____     ";


        if (!Validation.presenceCheck(this[_businessName])) {
            return message;
        }
        if (!Validation.presenceCheck(this[_firstName])) {
            return message;
        }
        if (!Validation.presenceCheck(this[_lastName])) {
            return message;
        }
        if (!Validation.presenceCheck(this[_email])) {
            return message;
        }
        if (!Validation.presenceCheck(this[_telephone])) {
            return message;
        }
        if (!Validation.presenceCheck(this[_city])) {
            return message;
        }
        if (!Validation.presenceCheck(this[_postcode])) {
            return message;
        }
        if (!Validation.presenceCheck(this[_address])) {
            return message;
        }
        if (!Validation.presenceCheck(this[_status])) {
```

```
            return message;
        }


        return '';
    }


    // checks that the length of the telephone attribute of the class has a length equal to 11 characters
    [_lengthCheck] () {
        if (!Validation.lengthCheck(this[_telephone], 11)){
            return "     _____     Please Ensure The Telephone Length Is 11 Characters Long     _____     ";
        }
        else {
            return '';
        }
    }

    // checks to see if the telephone is comprised of only numbers, if it contains other characters then a message of
    // what is wrong will be returned
    [_numberTypeCheck] () {
        if (!Validation.isaNumber(this[_telephone], true)){
            return "     _____     Please Ensure The Telephone Is Comprised Only Of Number Characters     _____     ";
        }
        else {
            return '';
        }
    }



    // type check gets done on all the fields to check that the value is of type string
    [_stringTypeCheck] () {

        let message = "     _____     Please Ensure All Text Fields Are Of Type String     _____     ";

        // if any of the text fields are not of type string then the message will get returned
        if (!Validation.isaString(this[_businessName])) {
            return message;
        }
        if (!Validation.isaString(this[_firstName])) {
            return message;
        }
        if (!Validation.isaString(this[_lastName])) {
            return message;
        }
        if (!Validation.isaString(this[_email])) {
            return message;
        }
        if (!Validation.isaString(this[_telephone])) {
            return message;
        }
        if (!Validation.isaString(this[_city])) {
            return message;
        }
        if (!Validation.isaString(this[_postcode])) {
            return message;
        }
        if (!Validation.isaString(this[_address])) {
            return message;
        }
        if (!Validation.isaString(this[_status])) {
            return message;
        }

        // if no messages have been returned, the method will return an empty string
```

```
138             return '';
139         }
140 
141         // range check done on
142         [_rangeCheck] () {
143 
144             let message = '';
145 
146             // if text field does not contain a set of data within the string length
                range a message will get returned
147             if (!Validation.rangeCheck(this[_businessName], 1, 25,true)) {
148                 message += "     ____     Ensure That The Business Name Field Is Within
                    Range Of 1 - 25 Characters     ____     ";
149             }
150             if (!Validation.rangeCheck(this[_firstName], 1, 15,true)) {
151                 message += "     ____     Ensure That The First Name Field Is Within Range
                    Of 1 - 15 Characters     ____     ";
152             }
153             if (!Validation.rangeCheck(this[_lastName], 1, 15,true)) {
154                 message += "     ____     Ensure That The Last Name Field Is Within Range
                    Of 1 - 15 Characters     ____     ";
155             }
156             if (!Validation.rangeCheck(this[_email], 4, 30,true)) {
157                 message += "     ____     Ensure That The Email Field Is Within Range Of 4
                    - 30 Characters     ____     ";
158             }
159             if (!Validation.rangeCheck(this[_city], 1, 15,true)) {
160                 message += "     ____     Ensure That The City Field Is Within Range Of 1
                    - 15 Characters     ____     ";
161             }
162             if (!Validation.rangeCheck(this[_postcode], 4, 12,true)) {
163                 message += "     ____     Ensure That The Post Code Field Is Within Range
                    Of 4 - 12 Characters     ____     ";
164             }
165             if (!Validation.rangeCheck(this[_address], 1, 30,true)) {
166                 message += "     ____     Ensure That The Address Field Is Within Range Of
                    1 - 30 Characters     ____     ";
167             }
168 
169             // if no messages have been returned, the method will return an empty string
170             return message;
171         }
172 
173 
174         [_emailFormatCheck] () {
175             if (!Validation.emailFormat(this[_email])){
176                 return "     ____     Please Ensure The Email Is Valid     ____     ";
177             }
178             else {
179                 return '';
180             }
181         }
182 
183         // does a look up check to ensure that the contact status is one of the
            acceptable values
184         [_lookUpCheckForStatus] () {
185             let acceptableValues = ['lead', 'dead-lead', 'recontact-lead', 'client',
                'past-client'];
186             if (!Validation.comparisonCheck(this[_status], acceptableValues)) {
187                 return '     ____     Ensure That The Contact Status Is One Of The
                    Acceptable Values     ____     ';
188             }
189             else {
190                 return '';
191             }
192         }
193 
194 
195         // Only allows basic characters to be accepted validation done to help
            circumvent XSS and the user typing commas
196         [_basicCharactersOnlyLookUpCheck] () {
197 
198             let message = "     ____     Ensure That All Fields  Other Than Email Only
```

```
            Contain a-z, A-Z , Spaces, And 0-9 Characters     ____     ";
199
200
201         if (!Validation.checkStringIsMadeOfOnlyLowerUpperAndNumberChars(this[_businessName], true)) {
202             return message;
203         }
204         if (!Validation.checkStringIsMadeOfOnlyLowerUpperAndNumberChars(this[_firstName], true)) {
205             return message;
206         }
207         if (!Validation.checkStringIsMadeOfOnlyLowerUpperAndNumberChars(this[_lastName], true)) {
208             return message;
209         }
210
211         if (!Validation.checkStringIsMadeOfOnlyLowerUpperAndNumberChars(this[_telephone], true)) {
212             return message;
213         }
214         if (!Validation.checkStringIsMadeOfOnlyLowerUpperAndNumberChars(this[_city], true)) {
215             return message;
216         }
217         if (!Validation.checkStringIsMadeOfOnlyLowerUpperAndNumberChars(this[_postcode], true)) {
218             return message;
219         }
220         if (!Validation.checkStringIsMadeOfOnlyLowerUpperAndNumberChars(this[_address], true)) {
221             return message;
222         }
223
224         return '';
225     }
226
227
228
229
230
231     // the public method that will be called on the instance of the class to validate it and
232     // output any necessary messages if it failed the validation
233     checkFieldsAreValid() {
234
235         let outputMessage = '';
236
237         // appends the return message from private methods to the outputMessage variable in teh public method
238         outputMessage += this[_presenceCheck]();
239         outputMessage += this[_lengthCheck]();
240         outputMessage += this[_numberTypeCheck]();
241         outputMessage += this[_stringTypeCheck]();
242         outputMessage += this[_lookUpCheckForStatus]();
243         outputMessage += this[_rangeCheck]();
244         outputMessage += this[_emailFormatCheck]();
245         outputMessage += this[_basicCharactersOnlyLookUpCheck]();
246
247
248         // if the output message is comprised of just empty strings then the return value will be true
249         if (!Validation.presenceCheck(outputMessage)){
250             return true;
251         }
252
253         // if there is more than an empty string in the output message the message will be returned
```

```
254             return outputMessage;
255         }
256
257
258
259
260
261
262
263
264     }
265
266     module.exports = ContactsValidation;
267
```

# DashboardValidation.js

Kieran Marcus Williams

```
const Validation = require("./Validation");


// creating "private" properties for the class by utilising the Symbol function
const _date = Symbol("date");
const _time = Symbol("time");
const _subject = Symbol("subject");
const _message = Symbol("message");


// creating "private" method names for the class by utilising the Symbol function
const _presenceCheck = Symbol("presenceCheck");
const _stringTypeCheck = Symbol("stringTypeCheck");
const _rangeCheck = Symbol("rangeCheck");
const _timeFormatCheck = Symbol("timeFormatCheck");
const _dateFormatCheck = Symbol("dateFormatCheck");
const _basicCharactersOnlyLookUpCheck = Symbol("basicCharactersOnlyLookUpCheck");


class DashboardValidation {

    constructor (date, time, subject, message) {
        this[_date] = date;
        this[_time] = time;
        this[_subject] = subject;
        this[_message] = message;
    }

    // presence check gets done on all the fields
    [_presenceCheck] () {

        let message = "     ____     Please Ensure All Text Fields Have Data Entered     ____     ";


        if (!Validation.presenceCheck(this[_date])) {
            return message;
        }
        if (!Validation.presenceCheck(this[_time])) {
            return message;
        }
        if (!Validation.presenceCheck(this[_subject])) {
            return message;
        }
        if (!Validation.presenceCheck(this[_message])) {
            return message;
        }

        return '';
    }


    // checks that the date is in the format dd/mm/yyyy and that the date entered is either the current date or a date in the future
    [_dateFormatCheck] () {
        if (!Validation.dateFormat(this[_date], true)) {
            return "     ____     Please Ensure The Date Is In The Format dd/mm/yyyy And The Date Is In The Future Or Today's Date     ____     "
        } else {
            return '';
        }
    }

    // checks that the time is in the format hh:mm
    [_timeFormatCheck] () {
        if (!Validation.timeFormat_HHMM(this[_time])) {
            return "     ____     Please Ensure The Time Is In The Format hh:mm Use The 24hr Clock     ____     "
        } else {
            return '';
        }
    }

```

```
70
71
72      // type check gets done on all the fields to check that the value is of type
        string
73      [_stringTypeCheck] () {
74
75          let message = "     ____     Please Ensure All Text Fields Are Of Type
            String    ____    ";
76
77          // if any of the text fields are not of type string then the message will
            get returned
78          if (!Validation.isaString(this[_date])) {
79              return message;
80          }
81          if (!Validation.isaString(this[_time])) {
82              return message;
83          }
84          if (!Validation.isaString(this[_subject])) {
85              return message;
86          }
87          if (!Validation.isaString(this[_message])) {
88              return message;
89          }
90
91          // if no messages have been returned, the method will return an empty string
92          return '';
93      }
94
95
96      // range check done on
97      [_rangeCheck] () {
98
99          let message = '';
100
101         // if text field does not contain a set of data within the string length
            range a message will get returned
102         if (!Validation.rangeCheck(this[_subject], 1, 40, true)) {
103             message += "     ____     Ensure That The Subject Field Is Within Range Of
                1 - 40 Characters    ____    ";
104         }
105         if (!Validation.rangeCheck(this[_message], 1, 200,true)) {
106             message += "     ____     Ensure That The Message Field Is Within Range Of
                1 - 200 Characters    ____    ";
107         }
108
109
110         // if no messages have been returned, the method will return an empty string
111         return message;
112     }
113
114
115     // Only allows basic characters to be accepted validation done to help
        circumvent XSS and the user typing commas
116     [_basicCharactersOnlyLookUpCheck] () {
117
118         let message = "     ____     Ensure That The Subject And Message Only Contain
            a-z, A-Z , And 0-9 Characters    ____    ";
119
120
121         if
            (!Validation.checkStringIsMadeOfOnlyLowerUpperAndNumberChars(this[_subject],
            true)){
122             return message;
123         }
124         if
            (!Validation.checkStringIsMadeOfOnlyLowerUpperAndNumberChars(this[_message],
            true)){
125             return message;
126         }
127
128         return '';
129     }
130
```

```
131
132
133
134
135         // the public method that will be called on the instance of the class to
            validate it and
136         // output any necessary messages if it failed the validation
137         checkFieldsAreValid() {
138
139             let outputMessage = '';
140
141             // appends the return message from private methods to the outputMessage
                variable in teh public method
142             outputMessage += this[_presenceCheck]();
143             outputMessage += this[_basicCharactersOnlyLookUpCheck] ();
144             outputMessage += this[_timeFormatCheck] ();
145             outputMessage += this[_stringTypeCheck]();
146             outputMessage += this[_dateFormatCheck] ();
147             outputMessage += this[_rangeCheck]();
148
149
150             // if the output message is comprised of just empty strings then the return
                value will be true
151             if (!Validation.presenceCheck(outputMessage)){
152                 return true;
153             }
154
155             // if there is more than an empty string in the output message, the message
                will be returned
156             return outputMessage;
157         }
158
159     }
160
161     module.exports = DashboardValidation;
162
```

# PasswordValidation.js

Kieran Marcus Williams

```
const Validation = require("./Validation");


// creating "private" properties for the class by utilising the Symbol function
const _password = Symbol("password");
const _confirmPassword = Symbol("confirmPassword");


// creating "private" method names for the class by utilising the Symbol function
const _presenceCheck = Symbol("presenceCheck");
const _stringTypeCheck = Symbol("stringTypeCheck");
const _rangeCheck = Symbol("rangeCheck");
const _basicCharactersOnlyLookUpCheck = Symbol("basicCharactersOnlyLookUpCheck");
const _stringLookUpCheck = Symbol("stringLookUpCheck");
const _comparisonCheck = Symbol("comparisonCheck");



class PasswordValidation {

    constructor (password, confirmPassword) {
        this[_password] = password;
        this[_confirmPassword] = confirmPassword;
    }

    // presence check gets done on all the fields
    [_presenceCheck] () {

        let message = "     ____     Please Ensure All Text Fields Have Data Entered     ____     ";


        if (!Validation.presenceCheck(this[_password])) {
            return message;
        }
        if (!Validation.presenceCheck(this[_confirmPassword])) {
            return message;
        }

        return '';
    }



    // type check gets done on all the fields to check that the value is of type string
    [_stringTypeCheck] () {

        let message = "     ____     Please Ensure All Text Fields Are Of Type String     ____     ";

        // if any of the text fields are not of type string then the message will get returned
        if (!Validation.isaString(this[_password])) {
            return message;
        }

        // if no messages have been returned, the method will return an empty string
        return '';
    }


    // range check done on
    [_rangeCheck] () {

        let message = "     ____     Ensure That The password Field Is Within Range Of 8 - 15 Characters     ____     ";

        // if text field does not contain a set of data within the string length range a message will get returned
        if (!Validation.rangeCheck(this[_password], 8, 15, true)) {
            return message;
        }
```

```
68
69
70          // if no messages have been returned, the method will return an empty string
71          return '';
72      }
73
74
75      // Only allows basic characters to be accepted validation done to help
        circumvent XSS and the user typing commas
76      [_basicCharactersOnlyLookUpCheck] () {
77
78          let message = "     ____     Ensure That The Note Only Contain a-z, A-Z , And
            0-9 Characters     ____     ";
79
80
81          if
            (!Validation.checkStringIsMadeOfOnlyLowerUpperAndNumberChars(this[_password],
            true)){
82              return message;
83          }
84
85          return '';
86      }
87
88      // checks that the password has at least one: uppercase character, lowercase
        character and number
89      [_stringLookUpCheck] () {
90
91          let message = "     ____     Ensure That Password Consists Of At Least One:
            Uppercase Character, Lowercase Character And Number     ____     ";
92
93          if (!Validation.stringLookupCheck(this[_password], {lowercase: true,
            uppercase: true, numbers: true})){
94              return message;
95          }
96
97          return '';
98      }
99
100
101     // checks that the password and confirm password are equal
102     [_comparisonCheck] () {
103         if (this[_password] !== this[_confirmPassword]){
104             return "     ____     Ensure That The Passwords Are Equal     ____     "
105         }
106         return '';
107     }
108
109
110
111
112     // will validate a the password to meet the specified criteria:
113     // greater than 8 characters and less than 20 characters , consisting of only
        numbers,
114     // uppercase letters and lowercase letters, and it contains at least one number,
115     // at least one number, at least one uppercase letter and at least one lowercase
        letter
116     // a comparison check is done lastly to ensure equality between both fields
117     // relevant messages will be returned if the password is not valid
118     // true is returned if the password is valid. can do this because JS is a
        dynamic type language
119
120     // the public method that will be called on the instance of the class to
        validate it and
121     // output any necessary messages if it failed the validation
122     checkFieldsAreValid() {
123
124         let outputMessage = '';
125
126         // appends the return message from private methods to the outputMessage
            variable in teh public method
127         outputMessage += this[_presenceCheck]();
128         outputMessage += this[_basicCharactersOnlyLookUpCheck] ();
```

```
129             outputMessage += this[_stringTypeCheck]();
130             outputMessage += this[_rangeCheck]();
131             outputMessage += this[_stringLookUpCheck]();
132             outputMessage += this[_comparisonCheck]();
133 
134             // if the output message is comprised of just empty strings then the return
                value will be true
135             if (!Validation.presenceCheck(outputMessage)){
136                 return true;
137             }
138 
139             // if there is more than an empty string in the output message, the message
                will be returned
140             return outputMessage;
141         }
142 
143     }
144 
145 
146     module.exports = PasswordValidation;
147 
148 
```

# Public Directory Contents

Kieran Marcus Williams

# Scripts Directory Contents

Kieran Marcus Williams

# AnalyticsScripts.js

Kieran Marcus Williams

```
// store the graph canvas elements into variables
let clientsGained = document.querySelector('#clients-gained').getContext('2d');
let clientsLost = document.querySelector('#clients-lost').getContext('2d');


// // dummy test data for the graphs
// let clientsGainedLabels = ['January', 'February', 'March', 'April', 'May', 'June', 'July', 'August', 'September', 'October', 'November', 'December'];
// let clientsGainedDataPoints = [20, 32,34,45,43,35,24,21,14,12,19,16];
//
// let clientsLostLabels = ['January', 'February', 'March', 'April', 'May', 'June', 'July', 'August', 'September', 'October', 'November', 'December'];
// let clientsLostDataPoints = [20, 32,45,23,54,21,25,31,45,17,11,8];




// when time scale option gets selected the form's action gets changed to the selected option's id
function changeAction() {
    let timeScale = $('#time-scale').val();
    console.log(timeScale);
    document.querySelector("#time-scale-form").action = "/analytics/" + timeScale;
    // line that changes the action attribute of the time scale form

}




// creates the graph for clients gained using the chart.js library
let clientsGainedGraph = new Chart(clientsGained , {
    type: 'line',
    data: {
        labels: clientsGainedLabels,
        datasets:[{
            label:'Number Of Clients Gained',
            data: clientsGainedDataPoints,
            backgroundColor: 'lightgreen',
            borderWidth: 1,
            borderColor: '#777',
        }]
    },
    options: {
        responsive: true,
        maintainAspectRatio: false,
        title: {
            display: true,
            text: 'Clients Gained',
            fontSize: 25,
        },
        legend:{
            display: false
        },
    }
});

// creates the graph for clients lost using the chart.js library
let clientsLostGraph = new Chart(clientsLost , {
    type: 'line',
    data: {
        labels: clientsLostLabels,
        datasets:[{
            label:'Number Of Clients Lost',
            data: clientsLostDataPoints,
            backgroundColor: 'lightcoral',
            borderWidth: 1,
            borderColor: '#777',
        }]
    },
    options: {
        responsive: true,
        maintainAspectRatio: false,
```

```
            title: {
                display: true,
                text: 'Clients Lost',
                fontSize: 25,
            },
            legend:{
                display: false
            },
        }
});

// // creates the graph for number of clients using the chart.js library
// let numberOfClientsGraph = new Chart(numberOfClients , {
//     type: 'line',
//     data: {
//         labels: numberOfClientsLabels,
//         datasets:[{
//             label:'Number Of Clients',
//             data: numberOfClientsDataPoints,
//             backgroundColor: 'lightblue',
//             borderWidth: 1,
//             borderColor: '#777',
//         }]
//     },
//     options: {
//         responsive: true,
//         maintainAspectRatio: false,
//         title: {
//             display: true,
//             text: 'Number Of Clients',
//             fontSize: 25,
//         },
//         legend:{
//             display: false
//         },
//     }
// });
```

# ChangePasswordScripts.js

Kieran Marcus Williams

```
1
2
3   //toggles the type in the password text boxes between password and text
4   // activated by the check box onClick()
5   function showPassword() {
6
7       let oldPasswordField = document.querySelector("#old-password");
8       let newPasswordField = document.querySelector("#new-password");
9       let confirmPasswordField = document.querySelector("#confirm-password");
10
11      if(oldPasswordField.type === "password" && newPasswordField.type === "password"
        && confirmPasswordField.type === "password"){
12          oldPasswordField.type = "text";
13          newPasswordField.type = "text";
14          confirmPasswordField.type = "text";
15      }
16      else {
17          oldPasswordField.type = "password";
18          newPasswordField.type = "password";
19          confirmPasswordField.type = "password";
20      }
21  }
22
```

# DashboardScripts.js

Kieran Marcus Williams

```
1
2   // function that prevents the enter/ return key from being used
3   function returnKeyBlocker(e) {
4       // sets e to be equal to either the event passed as argument or the window.event
        object for IE support
5       e = e || window.event;
6       // gets the key of e but if that is deprecated or isnt supported by the browser
        then then charCode is used
7       let keyCode = e.keyCode || e.charCode;
8       return keyCode !== 13;
9   }
```

# EncryptionNotesPasswordScripts.js

Kieran Marcus Williams

```
1
2
3   //toggles the type in the password text boxes between password and text
4   // activated by the check box onClick()
5   function showPassword() {
6
7       let PasswordField = document.querySelector("#password");
8       let confirmPasswordField = document.querySelector("#confirmPassword");
9
10      if(PasswordField.type === "password" && confirmPasswordField.type === "password"){
11          PasswordField.type = "text";
12          confirmPasswordField.type = "text";
13      }
14      else {
15          PasswordField.type = "password";
16          confirmPasswordField.type = "password";
17      }
18  }
19
```

# IndividualContactScripts.js

Kieran Marcus Williams

```
// function that prevents the enter/ return key from being used
function returnKeyBlocker(e) {
    // sets e to be equal to either the event passed as argument or the window.event object for IE support
    e = e || window.event;
    // gets the key of e but if that is deprecated or isnt supported by the browser then then charCode is used
    let keyCode = e.keyCode || e.charCode;
    return keyCode !== 13;
}

```

# Stylesheets Directory Contents

Kieran Marcus Williams

# AnalyticsStyles.css

Kieran Marcus Williams

```css
.top-half-of-analytics {
    height: 450px;
    display: flex;
    justify-content: space-around;
}

.bottom-half-of-analytics {
    display: flex;
    justify-content: space-between;
}

.middle-bottom-analytics {
    width: 250px;
}

.custom-select {
    width: 150px;
}

.time-scale-separator {
    display: flex;
    justify-content: space-between;
}

.conversion-rate-holder {
    padding-top: 20px;
    width: 400px;
    display: flex;
    justify-content: space-between;

}

.remove-margin {
    margin: 0;
}

.conversion-rate-table {
    width: 100%;
    height:75px;
    border: 1px solid black;
}

.row-of-buttons {
    width: 250px;
    margin: 10px;
    display: flex;
    justify-content: space-between;
}

.button-holder-analytics {
    width: fit-content;
}

td {
    border: 1px solid black;
    text-align: center;
    padding: 10px;
}

body {
    margin: 40px 80px;
}


canvas {
    width: 35vw;
}
```

# ChangePasswordStyles.css

Kieran Marcus Williams

```css
body {
    margin: 40px 40px;
}

.container {
    width: 800px;
}


.form-control {
    border: 1px solid grey;
}

.custom-select {
    border: 1px solid grey;
    height: 40px;
}

.btn-submit {
    position: relative;
    margin-top: 40px;
    margin-left: 20%;
    margin-right: 20%;
    width: 60%;
    height: 50px;
}


```

# ContactsListStyles.css

Kieran Marcus Williams

```css
body {
    margin: 40px 40px;
}

#modify-list-results-form {

    width: fit-content;
    padding: 10px;
}

#contact-management {
    margin-top: 30px;
    width: fit-content;
    padding: 10px;
}

#left-side-container {

    height: 700px;
    margin: 0;
    display: flex;
    flex-direction: column;
    justify-content: space-between;
}

#right-side-container {
    margin-right: 2%;
    height: 600px;
    width: 1000px;
    overflow: scroll;
}


.table {
    position: inherit;
    top: 0;
    bottom: 0;
    left: 0;
    right: 0;
    width: 100%;
}


.holder {
    margin: 30px 0;
    width: 100%;
    display: flex;
    justify-content: space-between;
}

.submit-col {
    display: flex;
}
```

# CreateContactsStyles.css

Kieran Marcus Williams

```css
body {
    margin: 40px 40px;
}

.container {
    width: 800px;
}

.form {
    margin-top: 20px;
    padding: 30px;
}

.form-control {
    border: 1px solid grey;
}

.custom-select {
    border: 1px solid grey;
    height: 40px;
}

.btn-submit {
    position: relative;
    margin-top: 40px;
    margin-left: 20%;
    margin-right: 20%;
    width: 60%;
    height: 50px;
}
```

# DashboardStyles.css

Kieran Marcus Williams

```css
#dashboard-separator {
    display: flex;
    flex-direction: row;
    justify-content: space-between;
}

#schedule-task-form {
    margin: 15px;
}

#today {
    margin: 15px;
}

#future {
    margin: 15px;
}

.scheduled-display {
    display: flex;
}

.trash-button {
    color: darkgray;
}

.trash-button:hover {

    color: grey;

}


.dashboard-slots {
    border: solid 1px grey;
    padding: 30px 25px;
    width: 400px;
    height: 700px;
    overflow-y: scroll;
    word-wrap: break-word;
}


ul {
    list-style: none;
    margin: 0;
    padding: 0;
}

textarea {
    resize: none;
}

body {
    margin: 40px 80px;
}
```

# EditContactStyles.css

Kieran Marcus Williams

```css
body {
    margin: 40px 40px;
}

.container {
    width: 800px;
}



.form-control {
    border: 1px solid grey;
}

.custom-select {
    border: 1px solid grey;
    height: 40px;
}

.btn-submit {
    position: relative;
    margin-top: 40px;
    margin-left: 20%;
    margin-right: 20%;
    width: 60%;
    height: 50px;
}
```

# HomeStyles.css

Kieran Marcus Williams

```css
body {
    margin: 20px 40px;
}

h1 {
    margin-top: 60px;
}

.link-separator {
    margin-left: 20px;
}

#login-btn {
    margin-right: 40px;
}
```

# IndividualContactStyles.css

Kieran Marcus Williams

```css

#halfway-separator {
    display: flex;
    justify-content: space-between;
}

#left-side {
    display: flex;
    flex-direction: column;
    justify-content: space-around;
}

.contact-details{
    width: fit-content;
}

.add-note-section {
    width: fit-content;
}

.btn-submit-add-note {
    margin-top: 10px;
    float: right;
}

.btn-container-extra-notes {
    float: right;
    margin-top: 10px;
    width: 40%;
}

#extra-notes-btn-group {
    display: flex;
    justify-content: space-evenly;
}

.btn-submit-extra-notes {

}

.btn-submit-extra-notes-encrypt {

}

span {
    font-weight: bold;
    font-size: x-large;
    margin-right: 1em;
}

.contact-management-buttons {
    margin-top: 30px;
    width: 40%;
    display: flex;
}

.btn-edit-contact {
    margin-right: 20%;
}


.table {
    position: inherit;
    top: 0;
    bottom: 0;
    left: 0;
    right: 0;
    width: 100%;
}

.td-boreder {
    border-left: 1px solid lightgrey;
}
```

```
74
75  .thead-grey {
76      background-color: lightgrey;
77  }
78
79
80  #table-holder-notes{
81      width: 680px;
82      height: 400px;
83      overflow: scroll;
84  }
85
86  textarea {
87      resize: none;
88  }
89
90  body {
91      margin: 40px 60px;
92  }
93
```

# NavbarLoggedInStyles.css

Kieran Marcus Williams

```css

.separator {
    width: 100%;
    display: flex;
    justify-content: space-between;
}

.nav-item{
    font-size: large;
    margin: 0 20px;
}

.small-margin-right {
    margin-left: 20px;
}

.mid-nav{
    width: 50%;
}

.spread-nav-items{
    width: 100%;
    display: flex;
    justify-content: space-around;
}
```

# SettingsStyles.css

Kieran Marcus Williams

```css
body {
    margin: 20px 40px;
}

h1 {
    margin-top: 60px;
}
```

# Routes Directory Contents

Kieran Marcus Williams

# analyticsRoutes.js

Kieran Marcus Williams

```
const express = require("express");
const router = express.Router();
const CSVHandler = require("../classes/CSVHandler");
const DateHandler = require("../classes/DateHandler");
const Analyticals = require("../classes/Analyticals");




////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

//                                          GET ROUTES

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////


router.get("/", (req, res) => {
    res.redirect("/analytics/daily");
});


// route that will give graph data for the past 14 days
router.get("/daily", (req, res) => {

    let clientsGainedFile = new CSVHandler(__dirname + '/dataStoreFiles/clientsGained.csv');
    let clientsLostFile = new CSVHandler(__dirname + '/dataStoreFiles/clientsLost.csv');
    let leadsLostFile = new CSVHandler(__dirname + '/dataStoreFiles/leadsLost.csv');

    let analytics = new Analyticals(clientsGainedFile, clientsLostFile, leadsLostFile);

    // find the conversion rate for the month in ratio and percentage ==> put into the ejs file
    let conversionRateObj = analytics.GetConversionRateForTheMonth();

    // find the data points and labels for the past 14 days ==> put into the ejs file
    let dataPointsAndLabels = analytics.getDataPointsAndLabelsForThePast14Days(); // will store data points and labels for clients gained and lost in data points



    res.render("Analytics", {conversionRate: conversionRateObj, graphData: dataPointsAndLabels, timeScale: "daily"});
});




// route that will give graph data for the past 12 weeks
router.get("/weekly", (req, res) => {

    let clientsGainedFile = new CSVHandler(__dirname + '/dataStoreFiles/clientsGained.csv');
    let clientsLostFile = new CSVHandler(__dirname + '/dataStoreFiles/clientsLost.csv');
    let leadsLostFile = new CSVHandler(__dirname + '/dataStoreFiles/leadsLost.csv');

    let analytics = new Analyticals(clientsGainedFile, clientsLostFile, leadsLostFile);

    // find the conversion rate for the month in ratio and percentage ==> put into the ejs file
    let conversionRateObj = analytics.GetConversionRateForTheMonth();

    // find the data points and labels for the past 14 days ==> put into the ejs file
    let dataPointsAndLabels = analytics.getDataPointsAndLabelsForThePast12Weeks();
    // will store data points and labels for clients gained and lost in data points

```

```
61
62
63      res.render("Analytics", {conversionRate: conversionRateObj, graphData:
        dataPointsAndLabels, timeScale: "weekly"});
64  });
65
66
67
68
69  // route that will give graph data for the past 12 months
70  router.get("/monthly", (req, res) => {
71
72      let clientsGainedFile = new CSVHandler(__dirname +
        '/dataStoreFiles/clientsGained.csv');
73      let clientsLostFile = new CSVHandler(__dirname +
        '/dataStoreFiles/clientsLost.csv');
74      let leadsLostFile = new CSVHandler(__dirname + '/dataStoreFiles/leadsLost.csv');
75
76      let analytics = new Analyticals(clientsGainedFile, clientsLostFile,
        leadsLostFile);
77
78      // find the conversion rate for the month in ratio and percentage ==> put into
        the ejs file
79      let conversionRateObj = analytics.GetConversionRateForTheMonth();
80
81      // find the data points and labels for the past 14 days ==> put into the ejs file
82      let dataPointsAndLabels = analytics.getDataPointsAndLabelsForThePast12Months();
        // will store data points and labels for clients gained and lost in data points
83
84
85
86      res.render("Analytics", {conversionRate: conversionRateObj, graphData:
        dataPointsAndLabels, timeScale: "monthly"});
87  });
88
89
90
91
92
93  ////////////////////////////////////////////////////////////////////////////////
    ////////////////////////////
94
95  //                                      POST ROUTES
96
97  ////////////////////////////////////////////////////////////////////////////////
    ////////////////////////////
98
99
100 // route that will add a record to clients gained file
101 router.get("/clientGained", (req, res) => {
102     let clientsGainedFile = new CSVHandler(__dirname +
        '/dataStoreFiles/clientsGained.csv');
103     let monthAndYear = DateHandler.getCurrentDateTime();
104     // makes the month and year ==> yyyy-mm and puts it back into a string
105     monthAndYear = (monthAndYear.split(""));
106     monthAndYear.splice(7,18);
107     monthAndYear = monthAndYear.join("");
108
109     clientsGainedFile.addRecord([monthAndYear], true, true);
110     res.redirect("/analytics");
111 });
112
113
114
115 // route that will add a record to clients lost file
116 router.get("/clientLost", (req, res) => {
117     let clientsLostFile = new CSVHandler(__dirname +
        '/dataStoreFiles/clientsLost.csv');
118     let monthAndYear = DateHandler.getCurrentDateTime();
119     // makes the month and year ==> yyyy-mm and puts it back into a string
120     monthAndYear = (monthAndYear.split(""));
121     monthAndYear.splice(7,18);
122     monthAndYear = monthAndYear.join("");
```

```
123
124         clientsLostFile.addRecord([monthAndYear], true, true);
125         res.redirect("/analytics");
126     });
127
128
129
130     // route that will add a record to leads lost file
131     router.get("/leadLost", (req, res) => {
132         let leadsLostFile = new CSVHandler(__dirname + '/dataStoreFiles/leadsLost.csv');
133         let monthAndYear = DateHandler.getCurrentDateTime();
134         // makes the month and year ==> yyyy-mm and puts it back into a string
135         monthAndYear = (monthAndYear.split(""));
136         monthAndYear.splice(7,18);
137         monthAndYear = monthAndYear.join("");
138
139         leadsLostFile.addRecord([monthAndYear], true, true);
140         res.redirect("/analytics");
141     });
142
143
144     ////////////////////////////////////////////////////////////////////////////////////////////
     ////////////////////////////
145
146     //                                          DELETE ROUTES
147
148     ////////////////////////////////////////////////////////////////////////////////////////////
     ////////////////////////////
149
150
151     router.delete("/analytics/removeAllData", (req, res) => {
152
153         // delete records
154         let leadsLostFile = new CSVHandler(__dirname + '/datastoreFiles/leadsLost.csv');
155         let clientsLostFile = new CSVHandler(__dirname +
         '/datastoreFiles/clientsLost.csv');
156         let clientsGainedFile = new CSVHandler(__dirname +
         '/datastoreFiles/clientsGained.csv');
157
158         leadsLostFile.deleteAllRecords();
159         clientsLostFile.deleteAllRecords();
160         clientsGainedFile.deleteAllRecords();
161
162         // redirect to the analytics page
163         res.redirect("/analytics")
164
165     });
166
167     module.exports = router;
168
```

# contactsRoutes.js

Kieran Marcus Williams

```
const
    express = require("express"),
    router = express.Router(),
    CSVHandler = require("../classes/CSVHandler"),
    StringUtility = require('../utilityMethods/StringUtility'),
    DateHandler = require("../classes/DateHandler"),
    Sort = require("../classes/Sort"),
    Encryption = require("../classes/Encryption"),
    PasswordValidation = require("../classes/PasswordValidation"),
    ContactsValidation = require("../classes/ContactsValidation"),
    AddNoteValidation = require("../classes/AddNoteValidation");


////////////////////////////////////////////////////////////////////////////////////////////////////

//                                        GET ROUTES

////////////////////////////////////////////////////////////////////////////////////////////////////




//renders the contacts list page that outputs all the contacts in a table
router.get("/", (req, res) => {
    let contactsFile = new CSVHandler(__dirname + '/dataStoreFiles/contacts.csv');
    let allContactRecords = contactsFile.getAllRecords(); // gets all the contact records and stores in a  2D array
    let message = req.session.message;
    let errorMessage = req.session.errorMessage;

    // an object with a property containing the 2D Array is passed to the ejs template
    // searchValue: undefined allows me to use the same ejs page in the .post("/search") route
    res.render("ContactsList", {contactRecords: allContactRecords, titleAddition: undefined, flashMessage: message, errorMessage: errorMessage},
        function (err, html) {
            req.session.message = undefined;
            req.session.errorMessage = undefined;
            res.send(html);
        });
});


// downloads the contacts file then redirects to the base route
router.get("/download", (req, res) => {

    res.download(__dirname + '/dataStoreFiles/contacts.csv',
        'contactRecords' + ':' + DateHandler.getCurrentDateTime() + '.csv', // sets the name to be contacts records with the current date in with the title
        function(err){
        if (err) {
            console.log(err); // logs the error to the console
        } else {
            console.log("downloaded: " + __dirname + '/dataStoreFiles/contacts.csv');
        }
    });
});




// renders the individual contact page for the contact with the id in the url
router.get("/viewContact:id", (req, res) => {

    let contactsFile = new CSVHandler(__dirname + '/dataStoreFiles/contacts.csv');
    let notesFile = new CSVHandler(__dirname + '/dataStoreFiles/contactsNotes.csv');
    let extraNotesFile = new CSVHandler(__dirname + '/dataStoreFiles/contactsExtraNotes.csv');
    let id = req.params.id;
    // find contact by id to get the contacts record
```

```
67      let contactDetailsRecord = contactsFile.selectRecordByID(id);
68
69
70      // gets all the notes relating to the current contact that is being viewed
71      // stored in the contactNotes variable as a 2D array with each inner array being
        a record from the notes csv file
72      let contactsNotes = notesFile.getRecordsOnlyByField([3], id);
73
74
75
76      let contactsExtraNotes = extraNotesFile.getRecordsOnlyByField([2], id); // gets
        the contacts extra notes
77
78      contactsNotes = Sort.bubbleSort2D(contactsNotes,[1],false); // sorts the notes
        ascending by their date time stamp
79
80
81      if (contactsExtraNotes.length !== 0) { // if there is data in the contacts extra
        notes 2d array
82          contactsExtraNotes = contactsExtraNotes[0][3]; // gets the data from the
            extra notes data column
83      }
84      if (req.session.decryptedData !== undefined) {
85          contactsExtraNotes = req.session.decryptedData;
86      }
87
88
89
90      let flashMessage = req.session.message;
91      let errorMessage = req.session.errorMessage;
92
93      res.render("ViewIndividualContact",
94          {
95              contactDetails: contactDetailsRecord[0].record,
96              flashMessage: flashMessage,
97              errorMessage: errorMessage,
98              contactNotes: contactsNotes,
99              extraNotes: contactsExtraNotes
100         },
101         (err, html) => {
102         req.session.message = undefined;
103         req.session.errorMessage = undefined;
104         req.session.decryptedData = undefined;
105         res.send(html);
106     });
107 });
108
109
110
111
112
113
114 //renders the create contact page
115 router.get("/new", (req, res) => {
116     let message = req.session.message; // if there is a message in the session
        object then it will be outputted to the user
117     let errorMessage = req.session.errorMessage;
118
119     let businessName = req.session.businessName;
120     let firstName = req.session.firstName;
121     let lastName = req.session.lastName;
122     let email = req.session.email;
123     let telephone = req.session.telephone;
124     let city = req.session.city;
125     let postcode = req.session.postcode;
126     let address = req.session.address;
127
128
129     let valuesStoredInSessionFromBody = [businessName, firstName, lastName, email,
        telephone, city, postcode, address];
130
131     res.render("CreateContact",
132         {
```

```
133             flashMessage: message,
134             savedBodyValue: valuesStoredInSessionFromBody,
135             errorMessage: errorMessage
136 
137         },
138         function (err, html) {
139 
140         // sets all the session properties to undefined after the page has been
            rendered so that they don't
141         // get rendered into other pages that they shouldn't be in
142             req.session.message = undefined;
143             req.session.errorMessage = undefined;
144 
145             req.session.businessName = undefined;
146             req.session.firstName = undefined;
147             req.session.lastName = undefined;
148             req.session.email = undefined;
149             req.session.telephone = undefined;
150             req.session.city = undefined;
151             req.session.postcode = undefined;
152             req.session.address = undefined;
153 
154             res.send(html);
155     });
156 
157 });
158 
159 
160 
161 
162 
163 //renders the edit contact page Requires an id in the url to know which contact to
    look up
164 router.get("/edit:id", (req, res) => {
165     let contactsFile = new CSVHandler(__dirname + '/dataStoreFiles/contacts.csv');
166 
167     // req.params.id to get contact id from the url
168     let id = req.params.id; // gets the contacts id from the url
169     let contactsRecord = contactsFile.selectRecordByID(id);
170     let message = req.session.message;
171 
172     // handles the case where no contact with that Id was found in the files
173     // returns a basic html page with a link to go back to the contacts show page
174     if (contactsRecord.length === 0) {
175         res.send('<h1>Error 404: Contact Not Found</h1><br><<a href="/contacts">Go
            Back</a>');
176     }
177     else {
178         // ACCESS ANY RECORD ELEMENT VIA .record and then [element index]
179         res.render("EditContact", {contact: contactsRecord[0], flashMessage:
            message}, (err, html) => {
180             req.session.message = undefined;
181             res.send(html);
182         });
183     }
184 });
185 
186 
187 
188 
189 
190 ////////////////////////////////////////////////////////////////////////////////
    ////////////////////////////
191 
192 //                                      POST ROUTES
193 
194 ////////////////////////////////////////////////////////////////////////////////
    ////////////////////////////
195 
196 
197 
198 
199 
```

```
200
201 // post route to add a record to the contacts page
202
203 router.post("/", (req, res) => {
204     let contactsFile = new CSVHandler(__dirname + '/dataStoreFiles/contacts.csv');
205
206     // stores the values from the text fields into variables
207     // the some of the values get capitalized to aid the data maintainability
208     let businessName = StringUtility.capitalizeString(req.body.businessName);
209     let firstName =  StringUtility.capitalizeString(req.body.firstName);
210     let lastName =  StringUtility.capitalizeString(req.body.lastName);
211     let email =  req.body.email;
212     let telephone = req.body.telephone;
213     let city =  StringUtility.capitalizeString(req.body.city);
214     let postcode = req.body.postcode;
215     let address =  StringUtility.capitalizeString(req.body.address);
216     let status = req.body.status;
217
218     // do validation here
219
220     let Validation = new ContactsValidation(businessName,firstName, lastName, email,
        telephone, city, postcode, address, status);
221     let validationResult = Validation.checkFieldsAreValid(); // this will either
        store the boolean value true or a message of what needs to be fixed
222
223     if (validationResult === true) {
224         // if it passes validation, add record and redirect to contacts list page
225         contactsFile.addRecord([businessName, firstName, lastName, email, telephone,
            city, postcode, address, status]);
226         req.session.message = "Successfuly Created A Contact";
227         res.redirect("/contacts/");
228     }
229
230     else {
231         // if something does not pass the validation then the user will get
            redirected to the
232         // create contacts page and a req.session.message will get set to the
            validationResult output
233         // this can then be accessed at the /contacts/new route to notify the user
            where they went wrong
234
235         req.session.message = validationResult; // validation message
236
237         // there will also be other properties attached to the session memory
            property containing the entered form data
238         // so that the fields don't go blank if any of the user's inputs did not
            pass the validation
239         req.session.businessName = businessName;
240         req.session.firstName = firstName;
241         req.session.lastName = lastName;
242         req.session.email = email;
243         req.session.telephone = telephone;
244         req.session.city = city;
245         req.session.postcode = postcode;
246         req.session.address = address;
247
248
249         res.redirect("/contacts/new")
250
251     }
252
253 });
254
255
256
257
258
259 //renders the contacts list page that outputs any contacts that match the search value
260 router.post("/search", (req, res) => {
261     let contactsFile = new CSVHandler(__dirname + '/dataStoreFiles/contacts.csv');
262
263     let searchValue = req.body.searchBar; // gets the search value from the form
264
```

```
265         // finds all matches of the search value by looking in every column for one
266         let matchingRecords =
            contactsFile.getRecordsOnlyByField([0,1,2,3,4,5,6,7,8,9,10], searchValue);
267
268
269         // if there is no match then do the search again but with the search value
            capitalized
270         if (matchingRecords.length === 0){
271             searchValue = StringUtility.capitalizeString(searchValue);
272             matchingRecords =
                contactsFile.getRecordsOnlyByField([0,1,2,3,4,5,6,7,8,9,10], searchValue);
273
274             // if there is still no match, the user gets redirected to the contacts list
                page with all the contacts
275             if (matchingRecords.length === 0) {
276                 req.session.errorMessage = "No Matches found";
277                 res.redirect("/contacts/")
278             } else {
279
280                 res.render("ContactsList", {
281                     contactRecords: matchingRecords,
282                     titleAddition: "Searched for: " + searchValue,
283                     flashMessage : matchingRecords.length.toString() + " Matches Found",
284                     errorMessage: undefined
285                 });
286             }
287         } else {
288             res.render("ContactsList", {
289                 contactRecords: matchingRecords,
290                 titleAddition: "Searched for: " + searchValue,
291                 flashMessage : matchingRecords.length.toString() + " Matches Found",
292                 errorMessage: undefined
293             });
294         }
295
296     });
297
298
299
300
301
302     router.post("/sortBy", (req, res) => {
303         let contactsFile = new CSVHandler(__dirname + '/dataStoreFiles/contacts.csv');
304         let sortByValue = req.body.sortBy; // will return undefined if no value is
            selected
305         let ascOrDec = req.body.order; // will be either ascending or descending
306         let sortedArray = [];
307
308         // turns the names gotten back from the sort form into the index that the
            records will need to be sorted by
309         switch (sortByValue) {
310             case "date":
311                 sortByValue = {index: 1, label: "Date"}; // date is stored at index 1 in
                    the contacts csv file
312                 break;
313             case "city":
314                 sortByValue = {index: 7, label: "City"}; // city is stored at index 7 in
                    the contacts csv file
315                 break;
316             case "businessName":
317                 sortByValue = {index: 2, label: "Business Name"}; // business name is at
                    index 2 in the contacts csv file
318                 break;
319             default:
320                 // if undefined is returned because there is no value selected in the form
321                 // then the sortByValue will be set to the value 3 which is the index of
                    the first name
322                 sortByValue = {index: 3, label: "First Name"};
323         }
324
325         // turns the ascending/ descending option into a boolean value which can be used
            by the sortBy method
326         switch (ascOrDec) {
```

```
            case "ascending":
                ascOrDec = true;
                break;
            case "descending":
                ascOrDec = false;
                break;
            default:
                ascOrDec = false;
        }

        // sort the array by the sortByValue in ascending or descending order dependent on the ascOrDec boolean argument that
        // gets passed in. the sorted array gets stored into the sortedArray variable
        sortedArray = contactsFile.sortBy(sortByValue.index, ascOrDec);

        res.render("ContactsList",
            {
                contactRecords: sortedArray,
                titleAddition: "Sorted By: " + sortByValue.label,
                flashMessage: undefined,
                errorMessage: undefined

            });

    });









    // this route is used to add a note to the contactNotes file
    router.post("/addnote:id", (req, res) => {
        let contactsNotes = new CSVHandler(__dirname + '/dataStoreFiles/contactsNotes.csv');

        let contactId = req.params.id; // gets the contacts id from the url
        let note = req.body.addNote; // gets the data in the text area and stores it
        // gets the current date and then converts it into dd/mm/yyyy format
        let date = DateHandler.standardDateTo_Date_DDMMYYYY(DateHandler.getCurrentDateTime());


        // do validation on the note
        let validation = new AddNoteValidation(note);
        let validationValue = validation.checkFieldsAreValid();

        if (validationValue === true) { // if it passes the validation then add the record
            // add note to the file as:  noteId, dateTimeStamp, contactId, note
            contactsNotes.addRecord([date, contactId, note]);
            req.session.message = "Successfully Added A Note";
            res.redirect("/contacts/viewContact" + contactId);
        }

        else {
            // if the data does not pass the validation the user will get sent back to the same page with a
            // message telling where they went wrong
            req.session.errorMessage = validationValue;
            res.redirect("/contacts/viewContact" + contactId);
        }



    });









    // will take the user to the password page so that the user can enter a password to encrypt or decrypt the
    // Extra Notes text area data
```

```
395
396 router.post("/viewContact:id/encryptNotes_:encOrDec", (req, res) => {
397     let id = req.params.id; // get id from the url
398     let encryptOrDecrypt = req.params.encOrDec; // stores if the user wants to
        encrypt or decrypt the data
399     let message = req.session.message;
400
401     if (req.body.extraNotes !== undefined) {
402         req.session.extraNotesData = req.body.extraNotes; // get the extra notes
            data from the text area and store in the session memory
403     }
404
405     res.render("EncryptNotes", {contactId: id, encryptOrDecrypt: encryptOrDecrypt,
        flashMessage: message});
406 });
407
408
409
410 // preform encryption on the data and append a new record to the database if there
    is no record containing the contact's id
411 // if a record is found with the users id then the record will be edited to contain
    the new encrypted data
412 router.post("/viewContact:id/encrypt", (req, res) => {
413
414     let id = req.params.id;
415     let extraNotesFile = new CSVHandler(__dirname +
        '/dataStoreFiles/contactsExtraNotes.csv');
416
417
418     // does password validation and gets either true or a string value returned
419     let passwordValidation = new PasswordValidation(req.body.password,
        req.body.confirmPassword);
420     let validationResult = passwordValidation.checkFieldsAreValid();
421     let encryptedData;
422     let extraNotesRecord;
423
424     // do password validation, if it passes validation then continue
425     if (validationResult === true) {
426
427         let extraNotesData = req.session.extraNotesData; // stores the
            extraNotesData from the session memory
428         req.session.extraNotesData = undefined; // removes the property from the
            session memory
429
430         // encrypt data with the password
431
432         encryptedData = Encryption.xorEncrypt(extraNotesData, req.body.password);
433
434         // look for a record with an id of the contacts
435         // id found ==> edit the data to store the encrypted data
436         // id not found ==> add record : [id, timestamp, contact id, data]
437
438         // search for a match for the contacts id in the contactsExtraNotes file
439
440         extraNotesRecord = extraNotesFile.getRecordsOnlyByField([2], id);
441         if(extraNotesRecord.length === 0){ // if no match is found
442             extraNotesFile.addRecord([id, encryptedData]); // add record
443             req.session.message = "Successfully Encrypted Data";
444             res.redirect("/contacts/viewContact" + id);
445         }
446
447         // match is found
448         else {
449             // edit the data in that record
450             extraNotesFile.editRecord(extraNotesRecord[0][0], [["3", encryptedData]]);
451             req.session.message = "Successfully Changed The Encrypted Data";
452             res.redirect("/contacts/viewContact" + id);
453         }
454
455
456     }
457
458     else {
```

```
459             // if not, redirect to the /contacts/viewContact:id/encryptNotes_encrypt
460             // with a message of what went wrong
461             req.session.message = validationResult;
462             // redirects to the post route /viewContact:id/encrypt
463             res.redirect(307, "/contacts/viewContact" + id + "/encryptNotes_encrypt");
464         }
465
466
467
468
469
470
471
472     });
473
474
475
476
477     // preform decryption on the data stored in req.session.extraNotesData
478     // if req.session.extraNotesData === undefined then redirect to the individual
        contacts page
479     router.post("/viewContact:id/decrypt", (req, res) => {
480         // do password validation, if it passes validation then continue
481         // if not, redirect to the /contacts/viewContact:id/encryptNotes_decrypt
482         // with a message of what went wrong
483
484         let id = req.params.id;
485         let extraNotesFile = new CSVHandler(__dirname +
            '/dataStoreFiles/contactsExtraNotes.csv');
486
487
488         // does password validation and gets either true or a string value returned
489         let passwordValidation = new PasswordValidation(req.body.password,
            req.body.confirmPassword);
490         let validationResult = passwordValidation.checkFieldsAreValid();
491         let extraNotesRecord;
492         let encryptedData;
493         let decryptedData;
494
495         // do password validation, if it passes validation then continue
496         if (validationResult === true) {
497
498             // decrypt data with the password
499
500
501             // look for a record with an id of the contacts
502             // id found ==> edit the data to store the encrypted data
503             // id not found ==> add record : [id, timestamp, contact id, data]
504
505             // search for a match for the contacts id in the contactsExtraNotes file
506             extraNotesRecord = extraNotesFile.getRecordsOnlyByField([2], id)[0];
507
508             if(extraNotesRecord.length === 0){ // if no match is found
509                 req.session.errorMessage = "No Data To Decrypt";
510                 res.redirect("/contacts/viewContact" + id);
511             }
512
513             // match is found
514             else {
515                 // decrypt the encrypted data and put that into session memory
516                 encryptedData = extraNotesRecord[3];
517                 decryptedData = Encryption.xorDecrypt(encryptedData, req.body.password);
518                 req.session.message = "Successfully Decrypted The Data";
519                 req.session.decryptedData = decryptedData;
520                 res.redirect("/contacts/viewContact" + id);
521             }
522
523
524         }
525
526         else {
527             // if not, redirect to the /contacts/viewContact:id/encryptNotes_encrypt
528             // with a message of what went wrong
```

```
529             req.session.message = validationResult;
530             // redirects to the post route /viewContact:id/encrypt
531             res.redirect(307, "/contacts/viewContact" + id + "/encryptNotes_decrypt");
532         }
533
534
535
536     });
537
538
539
540
541
542
543
544
545
546     ////////////////////////////////////////////////////////////////////////////////////////////////////
547
548     //                                          PUT ROUTES
549
550     ////////////////////////////////////////////////////////////////////////////////////////////////////
551
552
553
554
555     // route for updating files containing data
556     router.put("/:id", (req, res) => {
557         let contactsFile = new CSVHandler(__dirname + '/dataStoreFiles/contacts.csv');
558         let record = req.body; // stores the data from the edit contact form in an object
559         let recordArray = Object.values(record); // turns the record object into an array
560
561         let arrayOfEdits = [];
562
563         // populate an array of changes to be made
564         for (let i = 0; i < recordArray.length; i++) {
565             // capitalizes the values stored in the array at the bellow index's
566             if (i === 0 || i === 1 || i === 2 || i === 5 || i === 7 ) {
567                 recordArray[i] = StringUtility.capitalizeString(recordArray[i]);
568             }
569             // puts the data into the format needed for the edit record method to work
570             let recordChange = [(i+2).toString(), recordArray[i]];
571             arrayOfEdits.push(recordChange);
572         }
573
574         // do validation here
575
576         let Validation = new ContactsValidation(record.businessName, record.firstName,
            record.lastName,
577             record.email, record.telephone, record.city, record.postcode,
                record.address, record.status);
578
579         let validationResult = Validation.checkFieldsAreValid(); // this will either
            store the boolean value true or a message of what needs to be fixed
580
581         if (validationResult === true) {
582             // update all fields in the record that has been updated
583             contactsFile.editRecord(req.params.id, arrayOfEdits);
584             req.session.message = "Contact Successfully Edited";
585             res.redirect("/contacts/"); // takes user back to the contact list page
586         }
587
588         else {
589             // if something does not pass the validation then the user will get
                redirected to the
590             // create contacts page and a req.session.message will get set to the
                validationResult output
591             // this can then be accessed at the /contacts/new route to notify the user
                where they went wrong
592
593             req.session.message = validationResult; // validation message
```

```
594
595
596         // if validation not met then a message gets generated and the user gets
            sent to the edit contact page
597         // with that message stored in the session memory object
598
599         res.redirect("/contacts/edit" + req.params.id); // takes user back to the
            contacts list page
600     }
601 });
602
603
604
605
606 ////////////////////////////////////////////////////////////////////////////////
    ////////////////////////////
607
608 //                                      DELETE ROUTES
609
610 ////////////////////////////////////////////////////////////////////////////////
    ////////////////////////////
611
612
613 // delete route to remove a record
614 // when a contact is deleted, all of its notes and its extra notes records should
    also get deleted
615 router.delete("/delete:id", (req, res) => {
616     let contactsFile = new CSVHandler(__dirname + '/dataStoreFiles/contacts.csv');
617     let contactsExtraNotesFile = new CSVHandler(__dirname +
        '/dataStoreFiles/contactsExtraNotes.csv');
618     let id = req.params.id; // gets the contacts id from the url
619     let contactsExtraNoteRecord;
620
621     let contactsNotesTempFileForDelete;
622
623     contactsFile.deleteRecord(id); // deletes the contact record from the
        contacts.csv file
624
625     // gets the record with the contacts id at index 2 Then returns a 2D array with
        the record in the only array
626     // within the 2D array. record accessed with the [0] at the end
627     contactsExtraNoteRecord = contactsExtraNotesFile.getRecordsOnlyByField([2],
        id)[0];
628     contactsExtraNotesFile.deleteRecord(contactsExtraNoteRecord[0]); // the [0] gets
        the data at the 0 index which is the id
629
630     res.redirect("/contacts/");
631 });
632
633
634
635 module.exports = router;
```

# dashboardRoutes.js

Kieran Marcus Williams

```
const express = require("express");
const router = express.Router();
const CSVHandler = require("../classes/CSVHandler");
const StringUtility = require('../utilityMethods/StringUtility');
const DateHandler = require('../classes/DateHandler');
const Sort = require('../classes/Sort');
const DashboardValidation = require('../classes/DashboardValidation');



////////////////////////////////////////////////////////////////////////////////////////////////////
//                                        GET ROUTES
////////////////////////////////////////////////////////////////////////////////////////////////////




// renders the Dashboard page that has a form to add a scheduling and outputs the
// current days scheduled tasks and any future scheduled tasks

router.get("/", (req, res) => {
    let scheduleFile = new CSVHandler(__dirname + '/dataStoreFiles/scheduled.csv');
    let tasksWithCurrentDate = []; // will store a 2D array of all the tasks that
    that match the current Date
    let tasksWithFutureDate = [];
    let idsOfTasksWithPastDates = [];
    let timeStampOfTask = "";
    let date = "";
    let time = "";
    let message  = req.session.message;
    let errorMessage = req.session.errorMessage;

    // get all scheduled tasks
    let allScheduledTasks = scheduleFile.getAllRecords();
    let currentDateTime = DateHandler.getCurrentDateTime(); // gets the current date
    and time in a timestamp



    // determine which tasks have today's date
    allScheduledTasks.forEach((scheduledTaskRecord) => {


        timeStampOfTask = scheduledTaskRecord[2]; // will store the date time stamp
        in this element

        if (DateHandler.isToday(timeStampOfTask)) { // checks to see if the date of
        the task is equal to today's date
            tasksWithCurrentDate.push(scheduledTaskRecord);
        }
        else {
            // determine which tasks have a future date
            if (timeStampOfTask > currentDateTime) {
                tasksWithFutureDate.push(scheduledTaskRecord);
            }
            else {
                // determine which tasks that have a date that has passed
                if (timeStampOfTask < currentDateTime) {
                    idsOfTasksWithPastDates.push(scheduledTaskRecord[0]); // appends
                    the schedule record's id to an past dates array
                }
            }
        }
    });

    // gets rid of a record that has a date passed every time the route gets called
    // this is done to help manage the data stored in the scheduled.csv file
    // deleting multiple records every time can cause errors if the file doesn't
```

```
        have enough time to complete
67      // the method but this time is variable and could take too long so one is
        deleted at a time instead
68      if (idsOfTasksWithPastDates.length > 0) {
69          scheduleFile.deleteRecord(idsOfTasksWithPastDates[0]);
70      }
71
72      // sorts the tasks by their time stamps
73
74
75      tasksWithCurrentDate = Sort.bubbleSort2D(tasksWithCurrentDate, 2, true);
76      tasksWithFutureDate  = Sort.bubbleSort2D(tasksWithFutureDate, 2, true);
77
78
79      res.render("Dashboard",
80          {
81              scheduledToday: tasksWithCurrentDate,
82              scheduledFuture:tasksWithFutureDate,
83              flashMessage: message,
84              errorMessage: errorMessage
85          },
86          (err, html) => {
87          req.session.message = undefined;
88          req.session.errorMessage = undefined;
89          res.send(html);
90      });
91
92  });
93
94
95
96
97
98
99
100 ////////////////////////////////////////////////////////////////////////////////
    ////////////////////////////
101
102 //                                      POST ROUTES
103
104 ////////////////////////////////////////////////////////////////////////////////
    ////////////////////////////
105
106
107
108
109
110
111 // post route to add a scheduled task
112 router.post("/", (req, res) => {
113     let scheduleFile = new CSVHandler(__dirname + '/dataStoreFiles/scheduled.csv');
114
115     let date = req.body.date;
116     let time = req.body.time;
117     let subject = StringUtility.capitalizeString(req.body.subject);
118     let message = req.body.message;
119
120     // do validation on the fields and if they pass then add record and redirect to
        dashboard
121
122     let validation = new DashboardValidation(date, time, subject, message);
123     let validationResult = validation.checkFieldsAreValid();
124
125     if (validationResult === true) {
126
127         // record will contain id, date created, date and time in standard date time
            format, date Of scheduling, time of the scheduling, subject, message
128         scheduleFile.addRecord([DateHandler.createTimeStamp(date, time), date, time,
            subject, message]); // appends the record to the csv file
129         req.session.message = "Successfully Added A Scheduled Task";
130         res.redirect("/dashboard/")
131
132     }
```

```
133
134         // if they fail create a session message to say they have failed and redirect to
            dashboard
135         else {
136             req.session.errorMessage = validationResult; // validation result will have
                the details of what needs changing
137             res.redirect("/dashboard/");
138         }
139
140     });
141
142
143
144     ////////////////////////////////////////////////////////////////////////////////////
        ////////////////////////////
145
146     //                                        DELETE ROUTES
147
148     ////////////////////////////////////////////////////////////////////////////////////
        ////////////////////////////
149
150
151     //route to delete a schedule record
152     router.delete("/delete:id", (req, res) => {
153         let scheduleFile = new CSVHandler(__dirname + '/dataStoreFiles/scheduled.csv');
154         let id = req.params.id;
155         scheduleFile.deleteRecord(id);
156         req.session.message = "Successfully Deleted A Scheduled Task";
157         res.redirect("/dashboard/");
158     });
159
160
161
162     module.exports = router;
```

# index.js

Kieran Marcus Williams

```
const express = require("express");
const router = express.Router();

router.get("/", (req, res) => {
    res.render("Home");
});


router.get("/settings", (req, res) => {
    res.render("Settings");
});

router.get("/termsAndConditions", (req, res) => {
    res.render("TermsAndConditions");
});



module.exports = router;
```

# Utility Methods Directory Contents

Kieran Marcus Williams

# stringUtility.js

Kieran Marcus Williams

```
// utility functions so that I can avoid adding things to the prototype object of the String object
// as this could result in conflict with other properties

function StringUtility() {}

// static method (can be called without making an instance of the class in other objects, functions and classes)
// turns the first character of every word separated by a space into a capital letter if it is already lower case and a-z

StringUtility.capitalizeString = (inputString) => {

    let words = inputString.split(" ");
    let currentWordsCharacters = [];

    let lowerCaseArray = "abcdefghijklmnopqrstuvwxyz".split("");
    let upperCaseArray = "ABCDEFGHIJKLMNOPQRSTUVWXYZ".split("");

    for (let i = 0; i < words.length; i++) {
        currentWordsCharacters = words[i].split(""); // splits the current word into an array of characters

        // compare character to all the characters in the lowerCaseArray, if no match is found
        // don't do anything with it.

        for (let x = 0; x < 26; x++) {
            // if match is found, replace that character element with the
            // character stored in the same index but in the upperCaseArray to capitalize it
            if (lowerCaseArray[x] === currentWordsCharacters[0]) {
                currentWordsCharacters[0] = upperCaseArray[x];
                break;
            }
        }
        currentWordsCharacters = currentWordsCharacters.join(""); // puts the characters back together to form the string
        words[i] = currentWordsCharacters; // the capitalized word replaces the non capitalized word in that index of the words array
    }

    words = words.join(" ");
    return words;

};

module.exports = StringUtility;

```

# Views Directory Contents

Kieran Marcus Williams

# AnalyticsView.ejs

Kieran Marcus Williams

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Analytics</title>

    <link rel="stylesheet"
    href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css"
    integrity="sha384-Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm
    " crossorigin="anonymous">
    <link rel="stylesheet"
    href="https://use.fontawesome.com/releases/v5.6.3/css/all.css"
    integrity="sha384-UHRtZLI+pbxtHCWp1t77Bi1L4ZtiqrqD80Kn4Z8NTSRyMA2Fd33n5dQ8lWUE00s/
    " crossorigin="anonymous">
    <script
    src="https://cdnjs.cloudflare.com/ajax/libs/Chart.js/2.7.3/Chart.bundle.min.js"></
    script>
    <link rel="stylesheet" type="text/css" href="/stylesheets/AnalyticsStyles.css">
    <link rel="stylesheet" type="text/css"
    href="/stylesheets/NavbarLoggedInStyles.css">
    <link rel="stylesheet" type="text/css"
    href="/stylesheets/NavbarNotLoggedInStyles.css">
</head>
<body>


<% include partials/NavbarLoggedIn.ejs %>
<br><br>

<h1>Analytics</h1>
<br><br><br>

<!--top section with the graphs-->

<div class="split-screen-horizontal">

    <div class="top-half-of-analytics">
        <div class="first-graph">
            <canvas id="clients-gained"></canvas>
        </div>
        <div class="second-graph">
            <canvas id="clients-lost"></canvas>
        </div>
        <!--<div class="third-graph">-->
            <!--<canvas id="number-of-clients"></canvas>-->
        <!--</div>-->
    </div>
    <br><br><br>


<!--bottom section for conversion rate and the input analytical data buttons-->

    <div class="bottom-half-of-analytics">

        <div class="left-side-bottom-analytics">

           <div class="conversion-rate-holder">
                <h3>Conversion Rate This Month</h3>
                <table class="conversion-rate-table">
                    <tr>
                        <td><h4
                        class="remove-margin"><%=conversionRate.ratio%></h4></td>
                        <td><h4
                        class="remove-margin"><%=conversionRate.percentage%></h4></td>
                    </tr>
                </table>
            </div>

        </div>
```

```
61
62          <div class="middle-bottom-analytics">
63              <label for="time-scale">Time Scale</label>
64              <form id="time-scale-form" action="/analytics/" method="get">
65                  <div class="time-scale-separator">
66                      <select onchange="changeAction()" class="custom-select"
                        name="time-scale" id="time-scale">
67
68                          <!--ejs tags help determine which of the options should be
                            made selected -->
69
70                          <% if (timeScale === "daily") { %>
71                              <option selected class="time-scale-option"
                                value="daily">Daily</option>
72                          <% } else { %>
73                              <option selected class="time-scale-option"
                                value="daily">Daily</option>
74                          <% } %>
75
76                          <% if (timeScale === "weekly") { %>
77                              <option selected class="time-scale-option"
                                value="weekly">Weekly</option>
78                          <% } else { %>
79                              <option class="time-scale-option"
                                value="weekly">Weekly</option>
80                          <% } %>
81
82                          <% if (timeScale === "monthly") { %>
83                              <option selected class="time-scale-option"
                                value="monthly">Monthly</option>
84                          <% } else { %>
85                              <option class="time-scale-option"
                                value="monthly">Monthly</option>
86                          <% } %>
87
88                      </select>
89                      <button id="time-scale-btn" class="btn
                        btn-success">Submit</button>
90                  </div>
91              </form>
92          </div>
93
94          <div class="right-side-bottom-analytics">
95
96              <div class="button-holder-analytics">
97
98
99                  <div class="row-of-buttons">
100
101                     <a class="btn btn-primary" id="client-lost-btn"
                        href="/analytics/clientLost">Client Lost</a>
102
103                     <a class="btn btn-primary" id="client-gained-btn"
                        href="/analytics/clientGained">Client Gained</a>
104
105                 </div>
106
107                 <div class="row-of-buttons">
108
109                     <a class="btn btn-primary" id="lead-lost-btn"
                        href="/analytics/leadLost">Lead Lost</a>
110
111                     <a class="btn btn-danger" onclick="return confirm('Are You Sure
                        You Want To Delete ALL Analytics Data?')"
112                        id="remove-all-data-btn"
                           href="/analytics/removeAllData">Remove All Data</a>
113
114                 </div>
115             </div>
116         </div>
117
118
```

```
        </div>

    </div>









    <script src="https://code.jquery.com/jquery-3.2.1.slim.min.js"
    integrity="sha384-KJ3o2DKtIkvYIK3UENzmM7KCkRr/rE9/Qpg6aAZGJwFDMVNA/GpGFF93hXpG5KkN"
    crossorigin="anonymous"></script>
    <script
    src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.9/umd/popper.min.js"
    integrity="sha384-ApNbgh9B+Y1QKtv3Rn7W3mgPxhU9K/ScQsAP7hUibX39j7fakFPskvXusvfa0b4Q"
    crossorigin="anonymous"></script>
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js"
    integrity="sha384-JZR6Spejh4U02d8jOt6vLEHfe/JQGiRRSQQxSfFWpi1MquVdAyjUar5+76PVCmYl"
    crossorigin="anonymous"></script>




    <!-- sets the variables for the graphs to use - data points and labels-->
    <script>
        let clientsGainedLabels = "<%=graphData.clientsGained.labels%>".split(",");
        let clientsGainedDataPoints =
        "<%=graphData.clientsGained.dataPoints%>".split(",");

        let clientsLostLabels = "<%=graphData.clientsLost.labels%>".split(",");
        let clientsLostDataPoints = "<%=graphData.clientsLost.dataPoints%>".split(",");

    </script>

    <script src="/scripts/AnalyticsScripts.js"></script>

    </body>
    </html>

```

# ContactsListView.ejs

Kieran Marcus Williams

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Contacts</title>
    <link rel="stylesheet"
    href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css"
    integrity="sha384-Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm
    " crossorigin="anonymous">
    <link rel="stylesheet"
    href="https://use.fontawesome.com/releases/v5.6.3/css/all.css"
    integrity="sha384-UHRtZLI+pbxtHCWp1t77Bi1L4ZtiqrqD80Kn4Z8NTSRyMA2Fd33n5dQ8lWUE00s/
    " crossorigin="anonymous">
    <link rel="stylesheet" href="/stylesheets/ContactsListStyles.css">
    <link rel="stylesheet" href="/stylesheets/NavbarLoggedInStyles.css">
    <link rel="stylesheet" href="/stylesheets/NavbarNotLoggedInStyles.css">
</head>
<body>


<% include partials/NavbarLoggedIn.ejs %>


<%if (flashMessage !== undefined){%>
<div class="alert alert-success">
    <%= flashMessage %>
</div>
<%}%>

<%if (errorMessage !== undefined){%>
<div class="alert alert-danger">
    <%= errorMessage %>
</div>
<%}%>


<!--page for displaying the users contacts in a table-->

<h1>Contacts List
    <% if(titleAddition !== undefined){ %>
        -  <em style="margin-left: 15px;">"<%= titleAddition %>"</em>
    <%}%>
</h1>

<!--page is split up vertically-->

<div class="holder">

    <!--left side will contain the searching and sorting options along with options
    to add a contact, import contacts with a csv file
        and export contacts as a csv file-->

    <div id="left-side-container">
        <div class="jumbotron">
            <div id="modify-list-results-form">

                <form action="/contacts/search" method="post">
                    <label for="searchBar">Search</label><br>
                    <input name="searchBar" id="searchBar" type="text">
                    <button class="btn btn-sm btn-outline-dark"><i class="fas
                    fa-search"></i></button>
                </form>
                <br>

                <form action="/contacts/sortBy" method="post">

                    <h5>Sort By</h5>

                    <input type="radio" name="sortBy" id="date" value="date"> <label
                    for="date">Date</label> <br>

                    <input type="radio" name="sortBy" id="city" value="city"> <label
```

```
                    for="city">City</label> <br>
64
65                  <input type="radio" name="sortBy" id="business-name"
                    value="businessName"> <label for="business-name">Business
                    Name</label> <br><br>
66
67                  <label for="order">Order</label>
68                  <select class="custom-select" name="order" id="order">
69                      <option value="ascending">ascending</option>
70                      <option value="descending">descending</option>
71                  </select> <br><br>
72
73                  <input class="btn btn-success" type="submit">
74              </form>
75          </div>
76
77          <hr>
78
79          <div id="contact-management">
80
81              <a class="btn btn-primary" id="add-contact" href="/contacts/new">Add
                Contact</a>
82
83              <hr>
84
85              <!-- UNCOMMENT TO BE ABLE IMPORT CONTACTS HTML WHEN IT IS TO GET
                IMPLEMENTED -->
86
87              <!---->
88              <!--<form>-->
89                  <!--<div class="form-group">-->
90                      <!--<label for="import-contacts">Import Contacts (.CSV
                        file)</label>-->
91                      <!--<input type="file" class="form-control-file"
                        id="import-contacts">-->
92                  <!--</div>-->
93              <!--</form>-->
94              <!--<br>-->
95
96
97              <!--<form action="#">-->
98                  <!--<button class="btn btn-secondary"
                    id="import-contact-btn">import Contact</button>-->
99              <!--</form>-->
100
101             <br>
102
103             <form action="/contacts/download" method="get">
104                 <button class="btn btn-secondary" id="export-contact-btn">Export
                    Contacts</button>
105             </form>
106
107
108
109         </div>
110
111     </div>
112
113 </div>
114
115 <!--on the right side there is a table containing all the contacts with a link
    to the contact's individual page in the name field-->
116 <div id="right-side-container">
117     <table class="table">
118         <thead class="thead-dark">
119             <tr>
120                 <th>Name</th>
121                 <th>Business</th>
122                 <th>City</th>
123                 <th>Email</th>
124                 <th>Telephone</th>
125             </tr>
```

```
126             </thead>
127
128             <tbody>
129
130                 <!--gets the contacts details and puts it into the table row by row-->
131                 <% contactRecords.forEach((record) => { %>
132                     <tr>
133                         <td><a
                            href="/contacts/viewContact<%=record[0]%>"><%=record[3] + "
                            " + record[4]%></a></td>
134                         <td><%=record[2]%> </td>
135                         <td><%=record[7]%></td>
136                         <td><%=record[5]%></td>
137                         <td><%=record[6]%></td>
138                     </tr>
139                 <%});%>
140
141             </tbody>
142
143
144         </table>
145     </div>
146 </div>
147
148 <script src="/scripts/ContactListScripts.js"></script>
149 <script src="https://code.jquery.com/jquery-3.2.1.slim.min.js"
    integrity="sha384-KJ3o2DKtIkvYIK3UENzmM7KCkRr/rE9/Qpg6aAZGJwFDMVNA/GpGFF93hXpG5KkN"
    crossorigin="anonymous"></script>
150 <script
    src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.9/umd/popper.min.js"
    integrity="sha384-ApNbgh9B+Y1QKtv3Rn7W3mgPxhU9K/ScQsAP7hUibX39j7fakFPskvXusvfa0b4Q"
    crossorigin="anonymous"></script>
151 <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js"
    integrity="sha384-JZR6Spejh4U02d8jOt6vLEHfe/JQGiRRSQQxSfFWpi1MquVdAyjUar5+76PVCmYl"
    crossorigin="anonymous"></script>
152 </body>
153 </html>
```

# CreateContactView.ejs

Kieran Marcus Williams

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Create Contact</title>
    <link rel="stylesheet"
    href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css"
    integrity="sha384-Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm
    " crossorigin="anonymous">
    <link rel="stylesheet" href="/stylesheets/CreateContactStyles.css">
    <link rel="stylesheet" href="/stylesheets/NavbarLoggedInStyles.css">
    <link rel="stylesheet" href="/stylesheets/NavbarNotLoggedInStyles.css">
    <script src="/scripts/CreateContactScripts.js"></script>
</head>
<body>

<% include partials/NavbarLoggedIn.ejs %>



<%if (flashMessage !== undefined){%>
<div class="alert alert-danger">
    <%= flashMessage %>
</div>
<%}%>

<!--basic form using bootstrap that will be used as a data capture to make create a
contact-->

<h1>Create Contact</h1>
<div class="container">
    <div class="jumbotron">
        <form class="form" action="/contacts" method="post">
            <h3>Contact Details</h3><br>
            <div class="row">
                <div class="col">
                    <label for="business">Business</label>
                    <input class="form-control" id="business" name="businessName"
                    value="<%=savedBodyValue[0]%>" type="text"><br>
                </div>
                <div class="col">
                    <label for="first-name">First Name</label>
                    <input class="form-control" id="first-name" name="firstName"
                    value="<%=savedBodyValue[1]%>" type="text"><br>
                </div>
            </div>
            <div class="row">
                <div class="col">
                    <label for="last-name">Last Name</label>
                    <input class="form-control" id="last-name" name="lastName"
                    value="<%=savedBodyValue[2]%>" type="text"><br>
                </div>
                <div class="col">
                    <label for="email">Email</label>
                    <input class="form-control" id="email" name="email"
                    value="<%=savedBodyValue[3]%>" type="text"><br>
                </div>
            </div>
            <div class="row">
                <div class="col">
                    <label for="telephone">Telephone</label>
                    <input class="form-control" id="telephone" name="telephone"
                    value="<%=savedBodyValue[4]%>" type="text"><br>
                </div>
                <div class="col">
                    <label for="city">City</label>
                    <input class="form-control" id="city" name="city"
                    value="<%=savedBodyValue[5]%>" type="text"><br>
                </div>
            </div>
            <div class="row">
                <div class="col">
```

```
                        <label for="postcode">Postcode</label>
                        <input class="form-control" id="postcode" name="postcode"
                        value="<%=savedBodyValue[6]%>" type="text"><br>
                    </div>
                    <div class="col">
                        <label for="address">Address</label>
                        <input class="form-control" id="address" name="address"
                        value="<%=savedBodyValue[7]%>" type="text"><br>
                    </div>
                </div>
                <div class="row">
                    <div class="col">
                        <label for="status">Contact Status</label>
                        <select class="custom-select" name="status" id="status">
                            <option value="lead">Lead</option>
                            <option value="recontact-lead">Recontact Lead</option>
                            <option value="dead-lead">Dead Lead</option>
                            <option value="current-client">Current Client</option>
                            <option value="past-client">Past Client</option>
                        </select>
                    </div>
                </div>
                <input class="btn btn-primary btn-lg btn-submit" type="submit">
            </form>
        </div>
    </div>



    <script src="https://code.jquery.com/jquery-3.2.1.slim.min.js"
    integrity="sha384-KJ3o2DKtIkvYIK3UENzmM7KCkRr/rE9/Qpg6aAZGJwFDMVNA/GpGFF93hXpG5KkN"
    crossorigin="anonymous"></script>
    <script
    src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.9/umd/popper.min.js"
    integrity="sha384-ApNbgh9B+Y1QKtv3Rn7W3mgPxhU9K/ScQsAP7hUibX39j7fakFPskvXusvfa0b4Q"
    crossorigin="anonymous"></script>
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js"
    integrity="sha384-JZR6Spejh4U02d8jOt6vLEHfe/JQGiRRSQQxSfFWpi1MquVdAyjUar5+76PVCmYl"
    crossorigin="anonymous"></script>
    </body>
    </html>
```

# DashboardView.ejs

Kieran Marcus Williams

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Dashboard</title>

    <link rel="stylesheet"
    href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css"
    integrity="sha384-Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm
    " crossorigin="anonymous">
    <link rel="stylesheet"
    href="https://use.fontawesome.com/releases/v5.6.3/css/all.css"
    integrity="sha384-UHRtZLI+pbxtHCWp1t77Bi1L4ZtiqrqD80Kn4Z8NTSRyMA2Fd33n5dQ8lWUE00s/
    " crossorigin="anonymous">
    <link rel="stylesheet" href="/stylesheets/NavbarLoggedInStyles.css">
    <link rel="stylesheet" href="/stylesheets/NavbarNotLoggedInStyles.css">
    <link rel="stylesheet" href="/stylesheets/DashboardStyles.css">


</head>
<body>

<% include partials/NavbarLoggedIn.ejs %>


<%if (flashMessage !== undefined){%>
<div class="alert alert-success">
    <%= flashMessage %>
</div>
<%}%>

<%if (errorMessage !== undefined){%>
<div class="alert alert-danger">
    <%= errorMessage %>
</div>
<%}%>



<h1>Dashboard</h1>

<!-- The Dashboard is split into 3 sections  -->
<div id="dashboard-separator">


    <!--on the left there is the form to create a scheduling -->

    <div id="schedule-task-form">

        <h3>Schedule Task</h3>


        <form class="dashboard-slots" title="Schedule Task" action="/dashboard/"
        method="post" id="Schedule Task">
            <label for="date">date:</label> <br>
            <input id="date" name="date" title="date" type="text"
            placeholder="dd/mm/yyyy"> <br><br>

            <label for="time">time:</label><br>
            <input id="time" name="time" title="time" type="text" placeholder="24
            hour time: hh:mm"> <br><br>

            <label for="subject">subject:</label><br>
            <input id="subject" name="subject" type="text"> <br><br>

            <label for="message">message:</label> <br>
            <textarea onkeypress="return returnKeyBlocker(event)" id="message"
            title="message" name="message" cols="30" rows="10"></textarea> <br><br>

            <input class="btn btn-success" type="submit">
        </form>
    </div>
```

```
63
64      <!-- in the middle there will be an array that shows anything that is
        scheduled.csv for the current date -->
65      <div id="today">
66          <h3>Today</h3>
67          <div class="dashboard-slots">
68
69
70
71
72
73              <%scheduledToday.forEach((schedule) => {%>
74
75                  <tr>
76                      <td>
77
78                          <div class="schedule-info">
79                              <div class="time-display-today"><%=schedule[3]%></div>
80                              <div class="date-display-today"><%=schedule[4]%></div>
81                              <span style="font-weight: bold"><div
                                class="subject-display-today"><%=schedule[5]%></div></span
                                >
82                              <div class="message-display-today"><%=schedule[6]%></div>
83                          </div>
84
85                      </td>
86
87                      <td>
88                          <form
                            action="/dashboard/delete<%=schedule[0]%>?_method=DELETE"
                            method="post">
89                              <button style="float: right; margin-bottom: 10px;
                                font-size: x-large;" class="btn"><i class="trash-button
                                fas fa-trash-alt"></i></button>
90                          </form>
91
92                      </td>
93
94                  </tr>
95              <br><br><hr>
96
97              <%});%>
98
99          </div>
100     </div>
101
102
103     <!--on the right all the scheduled.csv tasks further on into the future from
        just the the current day will be put into this list -->
104
105     <div id="future">
106         <h3>Future</h3>
107         <div class="dashboard-slots">
108             <%scheduledFuture.forEach((schedule) => {%>
109
110             <tr>
111                 <td>
112
113                     <div class="schedule-info">
114                         <div class="time-display-today"><%=schedule[3]%></div>
115                         <div class="date-display-today"><%=schedule[4]%></div>
116                         <span style="font-weight: bold"><div
                            class="subject-display-today"><%=schedule[5]%></div></span>
117                         <div class="message-display-today"><%=schedule[6]%></div>
118                     </div>
119
120                 </td>
121
122                 <td>
123                     <form action="/dashboard/delete<%=schedule[0]%>?_method=DELETE"
                        method="post">
124                         <button style="float: right; margin-bottom: 10px; font-size:
```

```
                        x-large;" class="btn"><i class="trash-button fas
                        fa-trash-alt"></i></button>
                    </form>
                </td>

            </tr>
            <br><br><hr>



            <%});%>
        </div>
    </div>
</div>



<script src="/scripts/DashboardScripts.js"></script>
<script src="https://code.jquery.com/jquery-3.2.1.slim.min.js"
integrity="sha384-KJ3o2DKtIkvYIK3UENzmM7KCkRr/rE9/Qpg6aAZGJwFDMVNA/GpGFF93hXpG5KkN"
crossorigin="anonymous"></script>
<script
src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.9/umd/popper.min.js"
integrity="sha384-ApNbgh9B+Y1QKtv3Rn7W3mgPxhU9K/ScQsAP7hUibX39j7fakFPskvXusvfa0b4Q"
crossorigin="anonymous"></script>
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js"
integrity="sha384-JZR6Spejh4U02d8jOt6vLEHfe/JQGiRRSQQxSfFWpi1MquVdAyjUar5+76PVCmYl"
crossorigin="anonymous"></script>
</body>
</html>
```

# EditContactView.ejs

Kieran Marcus Williams

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Edit Contact</title>
    <link rel="stylesheet"
    href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css"
    integrity="sha384-Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm
    " crossorigin="anonymous">
    <link rel="stylesheet" href="/stylesheets/EditContactStyles.css">
    <link rel="stylesheet" href="/stylesheets/NavbarLoggedInStyles.css">
    <link rel="stylesheet" href="/stylesheets/NavbarNotLoggedInStyles.css">
    <script src="/scripts/EditContactScripts.js"></script>
</head>
<body>


<% include partials/NavbarLoggedIn.ejs %>

<%if (flashMessage !== undefined){%>
<div class="alert alert-danger">
    <%= flashMessage %>
</div>
<%}%>

<!--form containing a contacts data in the text fields-->
<h1>Edit Contact</h1>
<div class="container">

    <div class="jumbotron">
         <!--action url is set to put to have the correct http verb for updating db-->
        <form class="form" action="/contacts/<%= contact.record[0]%>?_method=PUT"
        method="post">
            <h3>Contact Details</h3><br>
            <div class="row">
                <div class="col">
                    <label for="business">Business</label>
                    <input class="form-control" id="business" type="text"
                    name="businessName" value="<%= contact.record[2]%>"><br>
                </div>
                <div class="col">
                    <label for="first-name">First Name</label>
                    <input class="form-control" id="first-name" type="text"
                    name="firstName" value="<%= contact.record[3]%>"><br>
                </div>
            </div>
            <div class="row">
                <div class="col">
                    <label for="last-name">Last Name</label>
                    <input class="form-control" id="last-name" type="text"
                    name="lastName" value="<%= contact.record[4]%>"><br>
                </div>
                <div class="col">
                    <label for="email">Email</label>
                    <input class="form-control" id="email"  type="text" name="email"
                    value="<%= contact.record[5]%>"><br>
                </div>
            </div>
            <div class="row">
                <div class="col">
                    <label for="telephone">Telephone</label>
                    <input class="form-control" id="telephone" type="text"
                    name="telephone" value="<%= contact.record[6]%>"><br>
                </div>
                <div class="col">
                    <label for="city">City</label>
                    <input class="form-control" id="city" type="text" name="city"
                    value="<%= contact.record[7]%>"><br>
                </div>
            </div>
            <div class="row">
                <div class="col">
```

```
                    <label for="postcode">Postcode</label>
                    <input class="form-control" id="postcode" type="text"
                    name="postcode" value="<%= contact.record[8]%>"><br>
                </div>
                <div class="col">
                    <label for="address">Address</label>
                    <input class="form-control" id="address" type="text"
                    name="address" value="<%= contact.record[9]%>"><br>
                </div>
            </div>
            <div class="row">
                <div class="col">
                    <label for="status">Contact Status</label>
                    <select class="custom-select" name="status" id="status">

                        <!--the bellow conditionals check to see the value of the
                        clients current status and makes the
                            the right one the selected value out of the drop down
                            menu-->

                        <%if(contact.record[10] === "lead") {%>
                            <option selected value="lead">Lead</option>
                        <%} else { %>
                            <option value="lead">Lead</option>
                        <%}%>

                        <%if(contact.record[10] === "recontact-lead") {%>
                            <option selected value="recontact-lead">Recontact
                            Lead</option>
                        <%} else { %>
                            <option value="recontact-lead">Recontact Lead</option>
                        <%}%>

                        <%if(contact.record[10] === "dead-lead") {%>
                            <option selected value="dead-lead">Dead Lead</option>
                        <%} else { %>
                            <option value="dead-lead">Dead Lead</option>
                        <%}%>

                        <%if(contact.record[10] === "current-client") {%>
                            <option selected value="current-client">Current
                            Client</option>
                        <%} else { %>
                            <option value="current-client">Current Client</option>
                        <%}%>


                        <%if(contact.record[10] === "past-client") {%>
                        <option selected value="past-client">Past Client</option>
                        <%} else { %>
                        <option value="past-client">Past Client</option>
                        <%}%>

                    </select>
                </div>
            </div>
            <input class="btn btn-primary btn-lg btn-submit" type="submit">
        </form>
    </div>
</div>




<script src="https://code.jquery.com/jquery-3.2.1.slim.min.js"
integrity="sha384-KJ3o2DKtIkvYIK3UENzmM7KCkRr/rE9/Qpg6aAZGJwFDMVNA/GpGFF93hXpG5KkN"
crossorigin="anonymous"></script>
<script
src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.9/umd/popper.min.js"
integrity="sha384-ApNbgh9B+Y1QKtv3Rn7W3mgPxhU9K/ScQsAP7hUibX39j7fakFPskvXusvfa0b4Q"
crossorigin="anonymous"></script>
```

```
124   <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js"
      integrity="sha384-JZR6Spejh4U02d8jOt6vLEHfe/JQGiRRSQQxSfFWpi1MquVdAyjUar5+76PVCmYl"
      crossorigin="anonymous"></script>
125   </body>
126   </html>
```

# EncryptNotesView.ejs

Kieran Marcus Williams

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Encrypt Extra Notes</title>
    <link rel="stylesheet"
    href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css"
    integrity="sha384-Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm
    " crossorigin="anonymous">
    <link rel="stylesheet" href="/stylesheets/ChangePasswordStyles.css">
    <link rel="stylesheet" href="/stylesheets/NavbarLoggedInStyles.css">
    <link rel="stylesheet" href="/stylesheets/NavbarNotLoggedInStyles.css">
    <script src="/scripts/EncryptionNotesPasswordScripts.js"></script>
</head>
<body>


<% include partials/NavbarLoggedIn.ejs %>


<%if (flashMessage !== undefined){%>
<div class="alert alert-danger">
    <%= flashMessage %>
</div>
<%}%>


<h1>Encryption Password</h1>
<div class="container">



    <!--basic  form to change the password-->
    <div class="jumbotron">
        <form class="form"
        action="/contacts/viewContact<%=contactId%>/<%=encryptOrDecrypt%>"
        method="post">
            <div class="row">

                <div class="col">
                    <label for="password">Password</label>
                    <input class="form-control" name="password" id="password"
                    type="password"><br>
                </div>
                <div class="col">
                    <label for="confirmPassword">Confirm Password</label>
                    <input class="form-control" name="confirmPassword"
                    id="confirmPassword" type="password"><br>
                </div>

            </div>

            <div class="row">
                <div class="col">
                    <div class="form-check">
                        <input class="form-check-input" id="see-password"
                        type="checkbox" onclick="showPassword();">
                        <label class="form-check-label" for="see-password">See
                        Password</label>
                    </div>
                </div>
            </div>
            <small id="password-help" class="form-text text-muted">
                Your password must be longer than 8 characters, contain upper and
                lower case letters, numbers,
                and must not contain spaces, special characters, or emoji.
            </small>
            <input class="btn btn-primary btn-lg btn-submit" type="submit">
        </form>

    </div>
</div>
```

```
63
64
65
66
67
68    <script src="https://code.jquery.com/jquery-3.2.1.slim.min.js"
      integrity="sha384-KJ3o2DKtIkvYIK3UENzmM7KCkRr/rE9/Qpg6aAZGJwFDMVNA/GpGFF93hXpG5KkN"
      crossorigin="anonymous"></script>
69    <script
      src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.9/umd/popper.min.js"
      integrity="sha384-ApNbgh9B+Y1QKtv3Rn7W3mgPxhU9K/ScQsAP7hUibX39j7fakFPskvXusvfa0b4Q"
      crossorigin="anonymous"></script>
70    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js"
      integrity="sha384-JZR6Spejh4U02d8jOt6vLEHfe/JQGiRRSQQxSfFWpi1MquVdAyjUar5+76PVCmYl"
      crossorigin="anonymous"></script>
71    </body>
72    </html>
```

# HomeView.ejs

Kieran Marcus Williams

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Home</title>
    <link rel="stylesheet"
    href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css"
    integrity="sha384-Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm
    " crossorigin="anonymous">
    <link rel="stylesheet"
    href="https://use.fontawesome.com/releases/v5.6.3/css/all.css"
    integrity="sha384-UHRtZLI+pbxtHCWp1t77Bi1L4ZtiqrqD80Kn4Z8NTSRyMA2Fd33n5dQ8lWUE00s/
    " crossorigin="anonymous">
    <link rel="stylesheet" href="/stylesheets/HomeStyles.css">
    <link rel="stylesheet" href="/stylesheets/NavbarLoggedInStyles.css">
    <link rel="stylesheet" href="/stylesheets/NavbarNotLoggedInStyles.css">
    <script src="/scripts/HomeScripts.js"></script>
</head>
<body>

<% include partials/NavbarLoggedIn.ejs %>

<!--contains a description of the site and links to register an account or log in-->

<div class="container">

    <h1>Welcome To KW CRM</h1>

    <br><br>

    <div class="jumbotron">
        <h3>About This Customer Relationship Management System</h3><br><br>
        <p>Many large organisations will have their own bespoke software that they
        would use as a customer relationship management system (CRM). However, this
        site is
            a generic CRM that is web based and will allow for many businesses that
            are unable to afford the bespoke software to have access to the services
            provided by them.
            The problem with many of the other online CRM's is that they offer too
            many services and overcomplicate things for many end users. The aim of
            this system is to incorporate
            only what small business owners and freelancers need and not overwhelm
            users with an abundance of features.
            Other online CRMS also often cost a monthly subscription to use.</p><br>

        <hr>
        <br>
        <h4><em>KW CRM is your best option for a free simple Customer Relationship
        Management System</em></h4><br><br>
    </div>
</div>




<script src="https://code.jquery.com/jquery-3.2.1.slim.min.js"
integrity="sha384-KJ3o2DKtIkvYIK3UENzmM7KCkRr/rE9/Qpg6aAZGJwFDMVNA/GpGFF93hXpG5KkN"
crossorigin="anonymous"></script>
<script
src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.9/umd/popper.min.js"
integrity="sha384-ApNbgh9B+Y1QKtv3Rn7W3mgPxhU9K/ScQsAP7hUibX39j7fakFPskvXusvfa0b4Q"
crossorigin="anonymous"></script>
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js"
integrity="sha384-JZR6Spejh4U02d8jOt6vLEHfe/JQGiRRSQQxSfFWpi1MquVdAyjUar5+76PVCmYl"
crossorigin="anonymous"></script>
</body>
</html>
```

# IndividualContactView.ejs

Kieran Marcus Williams

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Individual Contact</title>

    <link rel="stylesheet"
    href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css"
    integrity="sha384-Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm
    " crossorigin="anonymous">
    <link rel="stylesheet"
    href="https://use.fontawesome.com/releases/v5.6.3/css/all.css"
    integrity="sha384-UHRtZLI+pbxtHCWp1t77Bi1L4ZtiqrqD80Kn4Z8NTSRyMA2Fd33n5dQ8lWUE00s/
    " crossorigin="anonymous">

    <link rel="stylesheet" href="/stylesheets/ViewIndividualContactStyles.css">
    <link rel="stylesheet" href="/stylesheets/NavbarLoggedInStyles.css">
    <link rel="stylesheet" href="/stylesheets/NavbarNotLoggedInStyles.css">

</head>
<body>

<% include partials/NavbarLoggedIn.ejs %>

<!--if any error or success messages need to be outputted to the user then they will
be entered in here-->

<%if (flashMessage !== undefined){%>
    <div class="alert alert-success">
        <%= flashMessage %>
    </div>
<%}%>

<%if (errorMessage !== undefined){%>
<div class="alert alert-danger">
    <%= errorMessage %>
</div>
<%}%>


<!--this page will show all of the details of an individual contact-->
<!--the page is vertically split in half-->
<div id="halfway-separator">

    <!--contacts main details are here-->
    <div id="left-side">
        <h2><%= contactDetails[3] + " " + contactDetails[4]%></h2><br>
        <div class="contact-details">
            <div class="jumbotron">
                <table>
                    <tr>
                        <td>
                            <h5><span>Business: </span></h5>
                        </td>
                        <td>
                            <h5><%= contactDetails[2]%></h5>
                        </td>
                    </tr><tr>
                        <td>
                            <h5><span>Email: </span></h5>
                        </td>
                        <td>
                            <h5><%= contactDetails[5]%></h5>
                        </td>
                    </tr><tr>
                        <td>
                            <h5><span>Phone Number: </span></h5>
                        </td>
                        <td>
                            <h5><%= contactDetails[6]%></h5>
                        </td>
                    </tr><tr>
```

```
                        <td>
                            <h5><span>City: </span></h5>
                        </td>
                        <td>
                            <h5><%= contactDetails[7]%></h5>
                        </td>
                    </tr><tr>
                        <td>
                            <h5><span>Postcode: </span></h5>
                        </td>
                        <td>
                            <h5><%= contactDetails[8]%></h5>
                        </td>
                    </tr><tr>
                        <td>
                            <h5><span>Address: </span></h5>
                        </td>
                        <td>
                            <h5><%= contactDetails[9]%></h5>
                        </td>
                    </tr>

                    <!-- below code changes the data from the csv file for client
                    status into something that looks better
                            eg. current-client ===> Current
                            Client
                              -->

                    <tr>
                        <td>
                            <h5><span>Contacts Status: </span></h5>
                        </td>
                        <td>
                            <%if(contactDetails[10] === "lead") {%>
                                <h5>Lead</h5>
                            <%}%>

                            <%if(contactDetails[10] === "recontact-lead") {%>
                                <h5>Recontact Lead</h5>
                            <%} %>

                            <%if(contactDetails[10] === "dead-lead") {%>
                                <h5>Dead Lead</h5>
                            <%}%>


                            <%if(contactDetails[10] === "current-client") {%>
                                <h5>Current Client</h5>
                            <%}%>


                            <%if(contactDetails[10] === "past-client") {%>
                                <h5>Past Client</h5>
                            <%}%>
                        </td>
                    </tr>

                </table>


            </div>
        </div>

        <!--this text area form allows for the user to add a note which will get
        displayed in the notes table to the left-->
        <div class="add-note-section">

            <form action="/contacts/addnote<%=contactDetails[0]%>" method="post">

                <h3><label for="addNote">Add Note</label><br></h3>
                <textarea name="addNote" onkeypress="return returnKeyBlocker(event)"
                id="addNote" cols="60" rows="7"></textarea>
```

```
                    <br>
                    <input class="btn-submit-add-note btn btn-primary" type="submit">
                </form>

            </div>

            <!--edit contact button will redirect the user to the edit contact page with
            that contacts details filled in-->
            <!--the delete contact will ask for verification before deleting the
            contact-->

            <div class="contact-management-buttons">

                <a class="btn btn-secondary btn-edit-contact"
                href="/contacts/edit<%=contactDetails[0]%>">Edit Contact</a>

                <form action="/contacts/delete<%=contactDetails[0]%>?_method=DELETE"
                onsubmit="return confirm('Are You Sure You Want To Delete This
                Contact');"  method="post">
                    <button class="btn btn-danger btn-delete-contact">Delete
                    Contact</button>
                </form>

            </div>
        </div>


        <!--on the right side there is another text area that can be used to add, edit
        data and view data-->

        <div id="right-side">
            <div class="extra-notes">
                <br><br><br>

                <!--the place holder will display instructions as to how to save,
                encrypt and decrypt the data as
                    it is not super intuitive at first -->
                <form
                action="/contacts/viewContact<%=contactDetails[0]%>/encryptNotes_encrypt"
                method="post">
                    <h3><label for="extraNotes">Extra Notes</label></h3>
                    <textarea name="extraNotes" id="extraNotes" onkeypress="return
                    returnKeyBlocker(event)" cols="80" rows="10"
                              placeholder="
REMEMBER THE PASSWORD! There Is No Way Of Getting The Data Back If
You Forget It. Type Data into text area and press save & encrypt to save the data to
the file with xor encryption done to
to it. this data can be changed by first decrypting it, changing the data and then
pressing the save & encrypt button again.
When data is decrypted it will not save anything back to the file. If the wrong
decrypt password is entered, the decrypted text will
not be the original text
                                        "><%=extraNotes%></textarea><br>

                    <!--when the save and encrypt btn is pressed, the data in the text
                    field will get encrypted and saved to the database-->
                    <!--when the decrypt button is pressed the data will get retrieved
                    from the database and outputted to the text area-->
                    <!--a password page will appear before the encryption or decryption
                    happens though-->
                    <div class="btn-container-extra-notes">

                        <div id="extra-notes-btn-group">

                            <input class="btn-submit-extra-notes btn btn-secondary"
                                   type="submit" value="Decrypt"
                                   formaction="/contacts/viewContact<%=contactDetails[0]%>
                                   /encryptNotes_decrypt" formmethod="post">

                            <input class="btn-submit-extra-notes btn btn-primary"
                            type="submit" value="Save & Encrypt">

```

```
                    </div>

                </div>
            </form>
        </div>

        <br><br><br><br><br>

        <!--this is a table that will display all the notes and the date that each
        note was created next to it -->
        <div id="table-holder-notes">
            <table class="table ">
                <thead class="thead-grey">
                <tr>
                    <th>Date</th>
                    <th>Notes</th>
                </tr>
                </thead>

                <tbody>

                    <%contactNotes.forEach((noteRecord) => {%>

                        <tr>
                            <!-- the notes creation date gets put into here-->
                            <td class="td-boreder">
                                <%=noteRecord[2]%>
                            </td>
                            <!--the note data will be in here-->
                            <td class="td-boreder">
                                <%=noteRecord[4]%>
                            </td>

                        </tr>

                    <%});%>

                <tbody>
            </table>
        </div>


    </div>


</div>


<script src="/scripts/ViewIndividualContactScripts.js"></script>
<script src="https://code.jquery.com/jquery-3.2.1.slim.min.js"
integrity="sha384-KJ3o2DKtIkvYIK3UENzmM7KCkRr/rE9/Qpg6aAZGJwFDMVNA/GpGFF93hXpG5KkN"
crossorigin="anonymous"></script>
<script
src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.9/umd/popper.min.js"
integrity="sha384-ApNbgh9B+Y1QKtv3Rn7W3mgPxhU9K/ScQsAP7hUibX39j7fakFPskvXusvfa0b4Q"
crossorigin="anonymous"></script>
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js"
integrity="sha384-JZR6Spejh4U02d8jOt6vLEHfe/JQGiRRSQQxSfFWpi1MquVdAyjUar5+76PVCmYl"
crossorigin="anonymous"></script>

</body>
</html>
```

# SettingsView.ejs

Kieran Marcus Williams

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Terms And Conditions</title>
    <link rel="stylesheet"
    href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css"
    integrity="sha384-Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm
    " crossorigin="anonymous">
    <link rel="stylesheet"
    href="https://use.fontawesome.com/releases/v5.6.3/css/all.css"
    integrity="sha384-UHRtZLI+pbxtHCWp1t77Bi1L4ZtiqrqD80Kn4Z8NTSRyMA2Fd33n5dQ8lWUE00s/
    " crossorigin="anonymous">
    <link rel="stylesheet" href="/stylesheets/SettingsStyles.css">
    <link rel="stylesheet" href="/stylesheets/NavbarLoggedInStyles.css">
    <link rel="stylesheet" href="/stylesheets/NavbarNotLoggedInStyles.css">
    <script src="/scripts/SettingsScripts.js"></script>
</head>
<body>

<% include partials/NavbarLoggedIn.ejs %>

<!--This page will display links to the edit account page and to the Terms and
Conditions Page-->

<h1>Settings</h1>

<br><br><br>

<div class="jumbotron">

    <!-- UNCOMMENT WHEN IMPLEMENTATION OF USERS IS ADDED IN -->
    <!--<h4><a href="#">Edit Account Details</a></h4>-->

    <br>

    <h4><a href="/termsAndConditions">Terms And Conditions</a></h4>
</div>



<script src="https://code.jquery.com/jquery-3.2.1.slim.min.js"
integrity="sha384-KJ3o2DKtIkvYIK3UENzmM7KCkRr/rE9/Qpg6aAZGJwFDMVNA/GpGFF93hXpG5KkN"
crossorigin="anonymous"></script>
<script
src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.9/umd/popper.min.js"
integrity="sha384-ApNbgh9B+Y1QKtv3Rn7W3mgPxhU9K/ScQsAP7hUibX39j7fakFPskvXusvfa0b4Q"
crossorigin="anonymous"></script>
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js"
integrity="sha384-JZR6Spejh4U02d8jOt6vLEHfe/JQGiRRSQQxSfFWpi1MquVdAyjUar5+76PVCmYl"
crossorigin="anonymous"></script>
</body>
</html>
```

# TermsAndConditionsView.ejs

Kieran Marcus Williams

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Terms And Conditions</title>
    <link rel="stylesheet"
    href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css"
    integrity="sha384-Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm
    " crossorigin="anonymous">
    <link rel="stylesheet"
    href="https://use.fontawesome.com/releases/v5.6.3/css/all.css"
    integrity="sha384-UHRtZLI+pbxtHCWp1t77Bi1L4ZtiqrqD80Kn4Z8NTSRyMA2Fd33n5dQ8lWUE00s/
    " crossorigin="anonymous">
    <link rel="stylesheet" href="/stylesheets/TermsAndConditionsStyles.css">
    <link rel="stylesheet" href="/stylesheets/NavbarLoggedInStyles.css">
    <link rel="stylesheet" href="/stylesheets/NavbarNotLoggedInStyles.css">
</head>
<body>

<% include partials/NavbarLoggedIn.ejs %>

<br><br>
<h1>Terms And Conditions</h1>
<br><br>

<div class="jumbotron">
    <h5>By Using This Website You Are Agreeing To These Terms And
    Conditions.</h5>
    <br>
    <p>
        Any data held on your contacts is your responsibility. <br>
        You agree to be liable for any GDPR law breeches that are related to the
        contacts you have created. <br><br>

        If you wish to receive data that we hold regarding your usage of the system
        <br>
        you must make a formal request in the form of an email to
        kieranmarcus@KWCRM.com
    </p>

</div>


<script src="https://code.jquery.com/jquery-3.2.1.slim.min.js"
integrity="sha384-KJ3o2DKtIkvYIK3UENzmM7KCkRr/rE9/Qpg6aAZGJwFDMVNA/GpGFF93hXpG5KkN"
crossorigin="anonymous"></script>
<script
src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.9/umd/popper.min.js"
integrity="sha384-ApNbgh9B+Y1QKtv3Rn7W3mgPxhU9K/ScQsAP7hUibX39j7fakFPskvXusvfa0b4Q"
crossorigin="anonymous"></script>
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js"
integrity="sha384-JZR6Spejh4U02d8jOt6vLEHfe/JQGiRRSQQxSfFWpi1MquVdAyjUar5+76PVCmYl"
crossorigin="anonymous"></script>
</body>
</html>
```