|  |  |
| --- | --- |
| Title | Statement of Work For Shell & Employee Function |
| Release | 1.0 |

|  |  |  |
| --- | --- | --- |
| Version | Date | Status |
| 0.1 | 5/31/14 | Initial draft |
| 0.2 | 6/25/14 | Revisions after API re-write |
| 0.3 | 7/3/14 | Include API calls for employeeDetail and listData |

# Summary

This document sets out the initial Statement of Work between redPanda and NP Compete.

The aim of this work package is to:

1. Prove out the chosen technology stack (AngularJS, Bootstrap and Java APIs)
2. Test the remote methods of working between redPanda and NP Compete (communications, updates, testing and hand-offs)
3. Benchmark the time to develop so that subsequent screens/functions can be better budgeted

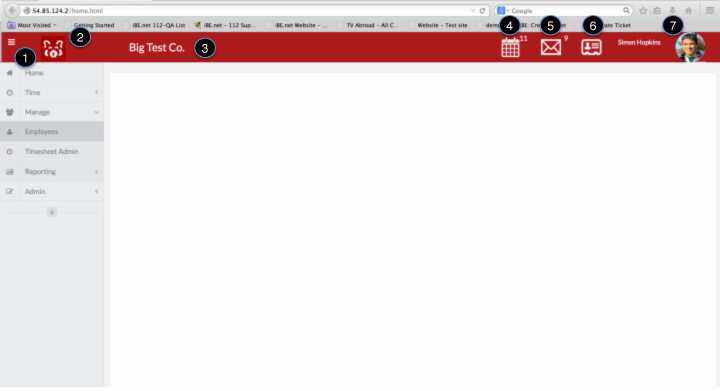
The work package has two parts:

1. Change the current static pages into one extendible AngularJS shell which can be used to run all of the intended functions from
2. Using the Employee maintenance function as an example, create the two screens that are needed to create, update and delete employees (by calling the appropriate API)

# Screen(s)

There are two screens that are needed, both of which have the same application shell. The application shell will incorporate menu, and title bar with defined functions:

## Application Shell



Notes:

1. Menu: the existing static html gives a good idea of the menu functionality. The structure is as follows:
   * Home
   * Time
     + Timesheets
   * Manage
     + Employees
     + Timesheet Admin
   * Reporting
     + Item Data
     + Dashboards
   * Admin
     + Set Up
     + User Admin
     + Interfaces

The menu will disappear to increase the available screen size, when the Macintosh HD:Users:simonhopkins:Library:Application Support:Evernote:quick-note:simonhopkins88___Evernote:quick-note-Vfc1cz:attachment--a88ZMi:screenshot.pngicon is clicked.

The menu collapse and will show only icons when the  icon is tapped.

(Note this functionality is already supported in the current pages)

Question: There will be different menus for different user types – how do we best pass this as a parameter

1. Customer Icon – this icon will be passed to the shell via the login API. If no icon is passed then use the redPandaIcon.png default

Also, when the icon is clicked, the user will be returned to the “Home” screen.

1. Customer Name – will be passed via the login API
2. Calendar: When clicked will show all appointments today (see list in static page). An API will be called to get this list – and the last option on the list will take the person to the appointments/calendar screen.

There is a badge on this icon to show the number of appointments in the list (***future***)

Note: there is an outstanding design item on how to refresh this list, i.e. when to call the API

1. Messages: When clicked, will show list of all unread/unactioned messages. An API will be called to get this list. The bottom option will take the user to the Home screen where messages are shown.

There is a badge on this icon to show the number of messages in the list (***future***)

Note: there is an outstanding design item on how to refresh this list, i.e. when to call the API

1. Contacts: Clicking on this icon will take the user to the contacts list screen (***future***)
2. User Name/Avatar: This will show the user’s name and avatar. This data will be passed via the login API

Clicking in this area will show a drop-down menu with 2 options:

Profile – which will navigate to the user’s profile modal pop-up (***future***)

Logout – which will log the user out of the system (do something with session tokens?) and return to the login screen (***future***)

Other

Any item marked (***future***) will navigate to a screen/function that has yet to be defined. This will not form part of this SOW

Much of this data will be passed in via the Login API (*future*) – for the creation of this shell – please use default/static data.

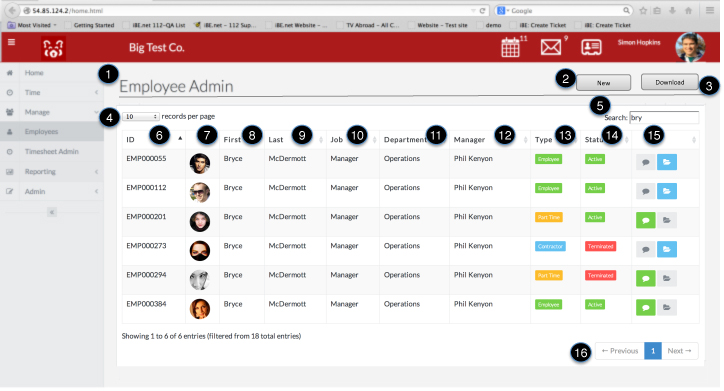
The screen is responsive and can be viewed on desktop, tablet and mobile device.

The color of the shell will be passed via API (users will have the opportunity to configure their own space) – there will be 2 parameters:

Color of background (currently #a91e22, which will be default)

Color of text (currently #ffffff, which will be default)

## Employee List Screen



Notes:

1. Function Title: This will show the name of the function (i.e. menu item) that has been chosen.
2. “New” button: Will navigate to the details screen in cerate mode
3. “Download” button, will download the current list to a Excel/CSV file (***future***)
4. Results per page, limits the number of items shown on a page (***current static page functionality***)
5. Search – will limit the displayed results to those that match the search criterion (***current static page functionality***)
6. Employee ID: returned from local model “identification” (see below)
7. Avatar/Thumbnail: returned from local model “thumbnail”
8. First name: returned from local model
9. Last Name: returned from local model
10. Job: returned from local model
11. Department: returned from local model
12. Manager: returned from local model
13. Type: returned from local model where:

If (contractor)

“Contractor”, use css class = “label label-info”

else if (partTime)

“Part Time”, use css class = “label label-warning”

else

“Employee”, use css class = “label label-success”

1. Status: returned from model where:

If (inactive)

“Inactive”, use css class = “label label-important”

else

“Active”, use css class = “label label-success”

1. There are 2 buttons:

Comments

- Show as green if “commentsExist = true”, else show grey (use css class = “label label-success”)

– show comments modal screen if clicked (***future***)

Attachments

* Show as blue if “attachmentsExist = true”, else show grey use css class = “label label- info”
* Show attachments modal screen (***future***)

1. The paging functionality will navigate from page to page (***current static page functionality***)

Other:

Columns may be sorted and searched (***current static page functionality***), using some combination of javascript, JQuery, CSS or other open-source code. This functionality needs to be retained.

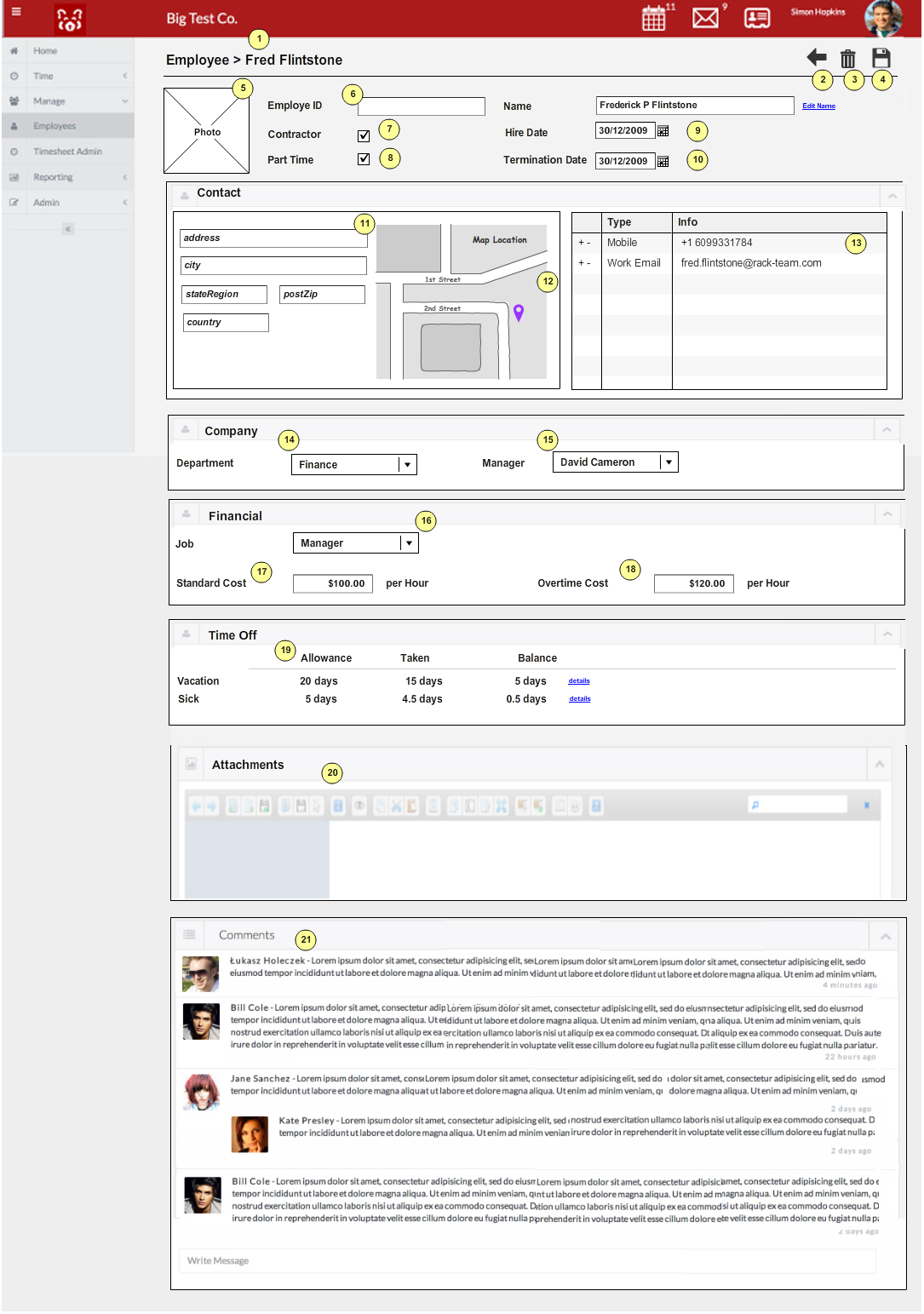
The desired effect is to highlight a row when it is selected (with a single mouse click), and navigate to the detail screen with a double-click. In the current static pages it is not how the code works.

### Local Model for list screen

The following UI model is needed to support the screen and will be populated via the employeeList API (see later)

|  |  |  |
| --- | --- | --- |
| Name | Type | Populated from (database) |
| id | \_id | from contact.id |
| employeeId | String | from contact.employee.identification |
| thumbUrl | url | From contact.thumbId |
| thumbnail | Binary | call binaryDownload with employee.thumbId |
| firstName | String | from contact.name.first |
| lastName | String | from contact.name.last |
| departmentName | String | call “find” for all unique departments and then populate with department.name |
| managerName | string | From nickname of employee with id = managerId |
| partTime | Boolean | From contact.employee.partTime |
| inactive | Boolean | From contact.employee.termDate |
| commentsExist | Boolean | Do comments exists for this employee |
| attachmentsExist | Boolean | Do attachments exist for this employee |
| Timestamp | Date | Timestamp of when the employee list model was last updated |

## Employee Detail Screen

****

Notes:

1. The name of the Screen (this will be the standard format for screens):

Employee: <**fullName**>

1. Back button – when pressed the user is returned to the list screen (if any changes have been made they will be saved automatically, and a call to …api/employeeDetails/update made)
2. Delete Button – when pressed there will be a pop-up to confirm the action

“Employee Record will be deleted” OK Cancel

1. Save button – when pressed the data is saved, and a call to …api/employeeDetails/update made
2. Employee’s thumbnail (taken from **thumbUrl**)

*FUTURE* - details for adding/deleting/changing are to be added

1. Employee Identification: Use **employeeId,** placeholder=**identification**

Employee Name: Use **nickname,** placeholder=**name**

*FUTURE* - There is a button to the right where the employee’s full name can be edited

1. Contractor: use **isContractor**
2. Part Time: use **isPartTime**
3. Hire date: use **hireDate,** placeholder=none

Use date picker – see CSS

<input type="text" class="form-control date-picker" id="date01" data-date-format="mm/dd/yyyy"/>

1. Termination date: use **termDate,** placeholder=none

Use date picker – see CSS

<input type="text" class="form-control date-picker" id="date01" data-date-format="mm/dd/yyyy"/>

1. Contact: this is a box that can be minimized (see static pages) – see CSS

<div class="box">

<div class="box-header">

<h2><i class="fa fa-edit"></i>Contacts</h2>

<div class="box-icon">

<a href="form-elements.html#" class="btn-minimize"><i class="fa fa-chevron-up"></i></a>

</div>

</div>

<div class="box-content">

…

address: use **addressStreet,** placeholder=address

city: use **addressCity,** placeholder=address

stateRegion: use **addressState,** placeholder=state

postZip: use **addressZip,** placeholder=zip

country: use **addressCountry,** placeholder=US

1. Map: using addressLat, and addressLong call google maps API

https://developers.google.com/maps/documentation/javascript/

1. Table showing list of contact numbers

Use the **contactNumbers** array

Use this CSS for the table

<table class="table table-bordered table-striped">

Note: we need to be able to add/delete items – choose easiest way to do this i.e. a “+” and a “x” button

For new items, the type must be a valid “contact number” type

1. Company: this is a box that can be minimized (see static pages), containing Department and Manager – see CSS detailed above

Department, use **departmentName**. For changes to the field use a drop-down list, check out

http://silviomoreto.github.io/bootstrap-select/

class="selectpicker" multiple

Note that when the departmentName is changed, both the departmentName and departmentId need to be changed in the model object.

1. Manager, use **managerName**. For changes to the field use a drop-down list, check out

http://silviomoreto.github.io/bootstrap-select/

class="selectpicker" multiple

Note that when the departmentName is changed, both the departmentName and departmentId need to be changed in the model object.

1. Financial: this is a box that can be minimized (see static pages), containing Job, Standard Cost and Overtime Cost – see CSS detailed above

Job: use job

1. Standard Cost: use **stdCostAmt** and **stdCostCur**

Note: we need to display currency amounts for different countries. It would be good to display the results as follows:



Where the currency was the ISO currency code of symbol and the amount was the 2dp float for the currency value.

When the currency symbol is clicked, a list of allowed currencies is shown

Note: if we can get this right here we can use the same CSS all over the app

1. Overtime Cost: use **otCostAmt** and **otCostCur**
2. Time off: this is a box that can be minimized (see static pages), containing sick and vacation details – see CSS detailed above

This is a table that will show the figures for vacation and sick, use:

**timeOff.type**

**timeoff.alowance**

**timeoff.taken**

Note: the balance column is computed from “Allowance” - “taken”

*FUTURE* – there is a button marked details that will take the user to a list of absences

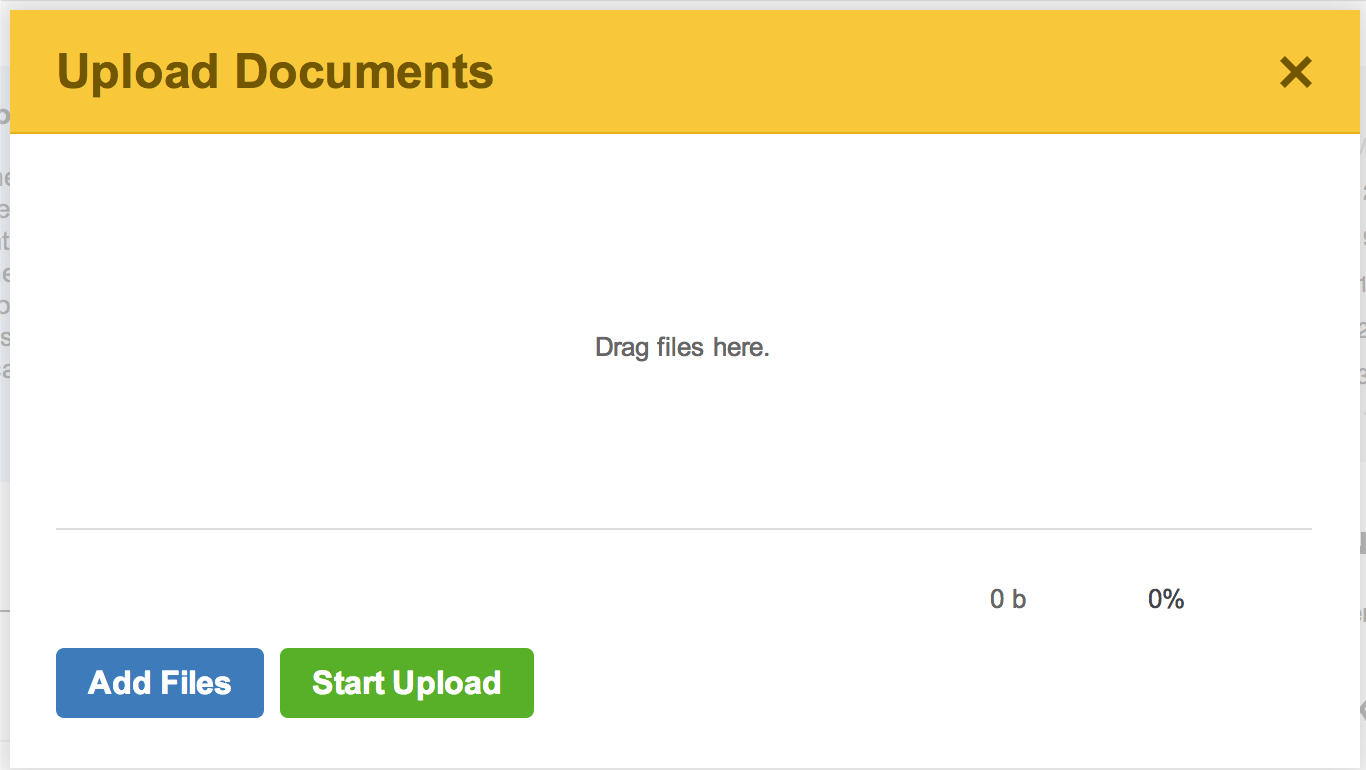
1. Attachments: this is a box that can be minimized (see static pages), containing the attachments table – see CSS detailed above

*FUTURE*

The aim here is to connect simple file manager that will upload files (photos, documents, pdfs) and associate them with the employee. Currently the template uses an open-source file manager – not sure if this is needed

<http://elfinder.org/>

<http://www.sitepoint.com/10-jquery-file-manager-plugins/>



1. Comments: this is a box that can be minimized (see static pages), containing the comments, authors and their thumbnail associated with each entry – see CSS detailed above

### Local Model for list screen

The following UI model is needed to support the Employee screen and will be populated via the employeeDetails API (see later)

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| id | String | Unique database id of Employee |
| employeeId | String | Given ID of Employee |
| thumbUrl | url | URL of the Employee’s thumbnail image |
| thumbnail | Binary | The binary for the thumbnail (??? Is this stored in the model???) |
| firstName | String | Employee’s first name |
| lastName | String | Employee’s last name |
| fullName | String | Employee’s full name |
| nickname | String | Employee’s nickname |
| isPartTime | Bool | Is the employee part time |
| isContractor | Bool | Is the employee a contractor (or full time employee) |
| addressStreet | String | Address street |
| addressCity | String | Address city |
| addressState | String | Address state (region for other regions) |
| addressZip | String | Address zip code (post code for other regions) |
| addressCountry | String | Address country |
| addressLat | Long | Address’ Latitude |
| addressLong | Long | Address’ Longitude |
| contactNumbers | Array | type – type of contact number (e.g. mobile, telephone)  details – number or URL  seq – order of the item |
| departmentName | String | Name of Department |
| departmentId | String | id of Department |
| managerName | String | Manager Name |
| managerId | String | Id of Manager |
| job | String | Employee’s Job name |
| stdCostAmt | Float | 2dp float for standard cost |
| stdCostCur | String | Currency of standard cost |
| otCostAmt | Float | 2dp float for overtime cost |
| otCostCur | String | Currency of ot cost |
| timeOff | Array | Type – will be either vacation or sick  Allowance – the amount that can be taken in a year |
| Timestamp | Date | Timestamp of when the employee model was last updated |

The following data can be queried from the database and may/may not for part of the local model

**contactNumberType** – e.g. phone, mobile, skype etc.

This is needed for the contact numbers list. Data can be obtained by calling the “listData” api (see API section) and specifying “contactNumbertypes”

**jobs** – e.g. manager, consultant etc.

This is needed for the jobs list. Data can be obtained by calling the “listData” api (see API section) and specifying “jobs”

**Departments** – e.g. finance, sales etc.

This is needed for the departments list. Data can be obtained by calling the “find” api (see API section) and specifying “department”

**Managers** – the data for valid managers can be found in the local model for the employee list. Manager’s “nickName” should be displayed in the selection list

all jobs

# APIs

|  |  |
| --- | --- |
| API | Description/Comment |
| …/api/employeeList?  timestamp=2014-05-31T09:30-0500 | Called from Employee List screen (on first population and refresh)  Return all employees to be shown in the list. The return payload should contain all of the fields needed on the UI’s model.  Note, the timestamp parameter is optional.   * If it is not specified all employees will be returned every time * If specified then only changes after that date will be retuned * If the server data is current an http message status “304” (data unchanged) will be returned with an empty payload   This API is called as a “GET”  Errors:  If the API fails, success = false, and the messages parameter will contain one or more text messages which should be output to the screen  Sample Output:  {"success":true,"total":5,"dataType":"employeeList","data":[{"id":"53ac1cac9c1c37083b3d38ab","employeeId":"PER00002","thumbUrl":"/data/resources/53b067f2e4b0c37f2d898b4f","firstName":"Simon","lastName":"Hopkins","departmentName":"Development","managerName":"Peter Blake","isPartTime":false,"isContractor":false,"inactive":false,"commentsExist":false,"attachmentsExist":false},{"id":"53ac275ee4b03340b6de4947","employeeId":"PER00003","thumbUrl":"/data/resources/53b06855e4b0c37f2d898b50","firstName":"Peter","lastName":"Blake","departmentName":"Operations","managerName":"Peter Blake","isPartTime":false,"isContractor":false,"inactive":false,"commentsExist":false,"attachmentsExist":false},{"id":"53ac2791e4b03340b6de4948","employeeId":"PER00005","thumbUrl":"/data/resources/53b0686fe4b0c37f2d898b51","firstName":"Jennifer","lastName":"Harvey","departmentName":"Consulting","managerName":"Peter Blake","isPartTime":false,"isContractor":false,"inactive":false,"commentsExist":false,"attachmentsExist":false},{"id":"53ac27b0e4b03340b6de4949","employeeId":"PER00009","thumbUrl":null,"firstName":"Danna","lastName":"Hargett","departmentName":"Operations","managerName":"Peter Blake","isPartTime":false,"isContractor":false,"inactive":false,"commentsExist":false,"attachmentsExist":false},{"id":"53ac2fa29c1c37083b3d38b0","employeeId":"PER00099","thumbUrl":"/data/resources/53b06889e4b0c37f2d898b52","firstName":"Richard","lastName":"Minney","departmentName":"Operations","managerName":"Peter Blake","isPartTime":false,"isContractor":true,"inactive":true,"commentsExist":false,"attachmentsExist":false}]} |
| …/api/employeeDetail/<id> | Called when the employee details screen is accessed.  The id is the mongodb \_id of the employee and is a mandatory parameter  All data needed for the Employee Details screen UI model will be returned.  This API is called as a “GET”  Errors:  If the API fails, success = false, and the messages parameter will contain one or more text messages which should be output to the screen  Sample output:  {"success":true,"total":0,"data":{"id":"53ac1cac9c1c37083b3d38ab","employeeId":"PER00002","photoUrl":"/data/resources/53b2fd06e4b0123558f41fe4.jpeg","thumbUrl":"/data/resources/thumbnail.53b2fd06e4b0123558f41fe4.jpeg","firstName":"Simon","lastName":"Hopkins","fullName":"Simon Hopkins","nickname":"Simon Hopkins","isPartTime":false,"isContractor":false,"addressStreet":"15 Tarkington Court","addressCity":"Princeton","addressState":"NJ","addressZip":"08540","addressCountry":"US","departmentId":"53ac290d9c1c37083b3d38ac","job":"Developer","stdCostAmt":88.5,"stdCostCur":"USD","timeOff":[{"vacAll":20.0,"vacTaken":2.0,"vacBal":0.0,"sickAll":5.0,"sickTaken":2.5,"sickBal":0.0}]}} |
| …/api/employeeDetail/update/<id>?  timestamp=2014-05-31T09:30-05:00 | Called when the employee details screen is exited, or the employee is specifically saved.  The timestamp parameter is mandatory and will be taken from the local model  Success:   * An http message “200” is sent and the current (possibly updated) employee data is returned in the payload   Errors:  If there is a conflict with the data on the server, a “409” http message is sent and the current Employee data is sent as a payload  This API is called as a “POST”  <In Development> |
| …/api/listData/jobs | Will return a list of valid Jobs for the drop down selector  Sample output  {"success":true,"total":1,"data":["Manager","Consultant","Tea Boy"]} |
| …/api/listData/ contactnumbertypes | Will return a list of valid contactNumberTypes for the drop down selector  Sample output  {"success":true,"total":1,"data":["Phone","email","Skype","LinkedIn"]} |
| …/api/find/department | Will return a list of the company’s departments. Note it is necessary to send both the department name as well as the id, since the Name is used for display/selection but it is the id that is stored on the employee record  Sample output:  {"success":true,"total":4,"dataType":"department","data":[{"name":"Development","id":"53ac290d9c1c37083b3d38ac"},{"name":"Operations","id":"53ac29159c1c37083b3d38ad"},{"name":"Finance","id":"53ac291c9c1c37083b3d38ae"},{"name":"Consulting","id":"53ac29229c1c37083b3d38af"}]} |

Logic to populate model:

1. Call find API for employeeList:
   1. Specify no timestamp if UI model is empty
   2. Specify timestamp = current date/time for model refresh

# Logic

|  |  |  |
| --- | --- | --- |
| Element | Action | Result |
|  |  |  |
|  |  |  |

# Validations

## List Screen

No validations are needed with respect to the list screen

## Details Screen

|  |  |
| --- | --- |
| Condition | Message |
|  |  |
|  |  |

# Test Scenarios

|  |  |  |
| --- | --- | --- |
| Step | Action | Expected Result |
|  |  |  |
|  |  |  |