MySite – Functional Specification

This document defines the Functions of MySite.

MySite is a server-side website using Node.JS, Express, Jade(Pug), and JavaScript. I developed a breadboard of this system in late 2016. I had no experience with any of these components. I do have a long history of developing software systems from 1975 to 2015 (40 years is a long time). So, everything I did relative to this website was a brand-new experience. After using my breadboard system for 1 year, I am in the process of doing thing closer to the right ways. This Functional Specification's Document is based on the new release 2.0 of MySite10.

Primary Requirement is to be able to build a website using a smart phone or tablet without having to know anything about website development.

Contents

1. Pages

There are only 6 Pages. Four are static (don't change), and 2 (Photos and Videos) that are dynamic (change).

1.1. Main, About, Contact, and Download Pages

These Pages are standard web pages. They have the Logo/Banner, Slider, Main Menu, Content, and Footer/Copyright Sections.

1.1.1. Main

The Main Page describes the purpose of this website, in the content section. The file MyWebSite/txt_main_contents.pug in the MyWebsite folder of cloud service contains the text to include in this section.

1.1.2. About

The About Page describes what this website is about in the content section.

The file MyWebSite/txt_about_contents.pug contains the text to include in this section.

1.1.3. Contact

The Contact Page is intended to allow users to send me an eMail. It is intended to be free advertisement for my website. The file

MyWebSite/txt_contact_contents.pug in the MyWebsite folder of cloud service contains the text to include in this section. This page contains the following input information that will be emailed to my eMail address:

- First Name
- Last Name
- Their eMail Address
- Message
- Submit Button

Validate all fields have been entered and eMail has proper format. If company name part of form, this is Spam. ignore it.

1.1.4. Download

Connect to a Cloud Service for Photos

A dropdown menu including the cloud services supported. Select the cloud service you want the Photos downloaded from.

Connect to a Cloud Service for Videos

A dropdown menu including the cloud services supported. Select the cloud service you want the Videos downloaded from.

Download All Photos & Videos

This menu selection will delete all Photos and Videos on the server and download them from the cloud service. Both Photos and Videos Cloud Service must be assigned to the same.

Download All Photos

This menu selection will delete all Photos on the server and download them from the cloud service.

Download All Videos

This menu selection will delete all Videos on the server and download them from the cloud service.

Download Just New Photos

This menu selection will download just new folders and new photo files in the cloud service Photos folder.

Download Just New Videos

This menu selection will download just new folders and new video files in the cloud service Videos folder.

1.2.Photos Page

There is just one Photos Page (photos.pug) file. It contains 3 parts that will change, each time a menu item is selected from the Alternate Navigation Menu (AltNavM) or LeftSide Navigation Menu (LftSdNavM). These are as follows:

1. Photos Include – This is an include file (photosInclude.pug) that contains the attribute records for displaying the photos in a gallery and full page when selected. Magnific Popup is the s/w product used to display the gallery and photos. This file is built at the end of Download function and stored in the associated menu structure on the server.
2. AltNavM Include – This is an include file (altNav.pug) that contains the List Attribute for the menus selected to get to this place in the menu structure. This file is built at the end of Download function and stored in the associated menu structure on the server.
3. LftSdNavM Include – This is an include file (leftSideNav.pug) that contains the List Attribute for the menus available from this place in the menu structure. This file is built at the end of Download function and stored in the associated menu structure on the server.

At the end of the Download process, these files are built for each folder encountered. For Photos the following file extensions are associated with photos files (.jpg, .jpeg, .png, .gif, .tiff, .bmp). Video file extensions (.m4v, .ogv, .web). This module will be called buildPhotoIncludeFiles.js and buildVideoIncludeFiels.js.

1.2.1. Magnific PopUp

Magnific PopUp is the software component used to display the gallery and individual photos. Each photo in a gallery can have one of four caption options:

- No Caption – There is no .txt. file with the same filename as the image file.

The next three options have a .txt file with the same filename as the image file.

- Caption –
  {

```
    "Title": "Describes this photo"
}
```

- URL –

```
{
    "Title": "Describes what this url points to",
    "DataSource":"url"
}
```

- PDF -

```
{
    "Title": " Describes what this pdf file is",
    "DataSource": "filename.pdf"
}
```

This image file and .pdf file must be in the same folder.

1.3.Videos Page

This design is like the Photos Page.

1.3.1. jPlayer & jPlayerPlayList

Use jPlayer to play using m4v, ogv, oga, wav, webm video filed. If there are more than one video file in a folder, use the j{layerPlayList to display the different videos.

2. File System Structure on Cloud Services

The names of the folders are the name used in the menus. The apostrophe character (‘) is removed from folder names because that character is not a validate file naming character.

File system at the server.

- Folder Name
  - File Name

Following is an example of the file/folder structure in the root of the Cloud Service.

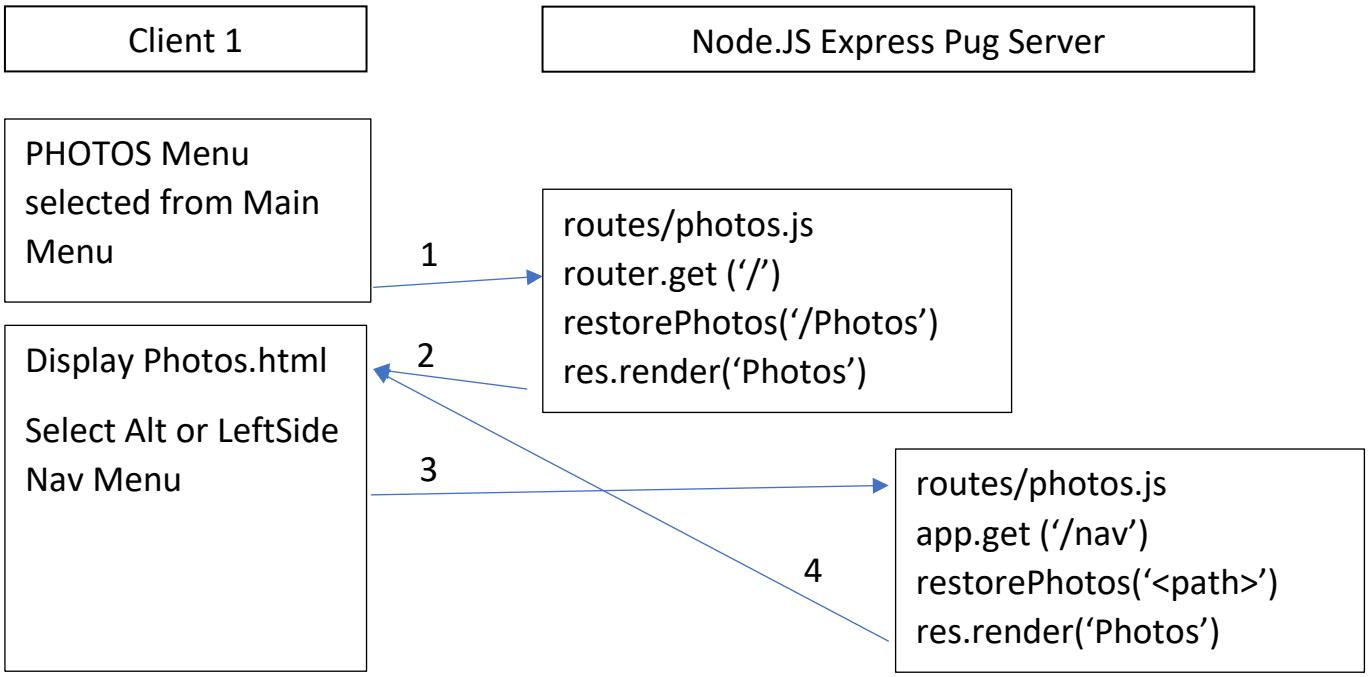
- MyWebsite
  - Photos
    - photosInclude.pug

        - altNav.pug
        - leftSideN.pug
        - <Some Photos Files>
        - Family & Friends
            - photosInclude.pug
            - altNav.pug
            - leftSideN.pug
            - <Some Photos Files>
            - Family
                - photosInclude.pug
                - altNav.pug
                - leftSideN.pug
                - <Some Photos Files>
- Videos
    - videosInclude.pug
    - altNav.pug
    - leftSideN.pug
    - <Some Video Files>
    - Family & Friends
        - videoInclude.pug
        - altNav.pug
        - leftSideN.pug
        - <Some Video Files>

3. General Menu Flow

Below defines the process of the user selecting menu items on MySite10.

1. User at browser clicks on the PHOTOS menu on Main Menu. Browser sends HTML Get Request to Server. Node.JS/Express sends this request to the module (routes/photos.js) at the router.get() function('/Photos') with the base path of the Photos structure. This function copies the three include files associated with this folder path to views/includes folder. The Photos.html has three 'include' commands for these three .pug files.
2. Photos.html set back to the Client browser.
3. User at browser clicks on a menu item from Alternate Navigation Menu or Leftside Navigation Menu. Browser sends HTML Get Request to Server. Node.JS/Express sends this request to the module (routes/photos.js) at the app.get() function('<path to structure where this menu points to> ') with the base path of the Photos structure. This function copies the three include files associated with this folder path to views/includes folder.
4. Photos.html set back to the Client browser. GoTo 2 to keep selecting menu items.



Problem: A known problem is a small window that exits of more than one client requests different photos or videos pages at the same time. I could queue all request if the previous request has not been completed. At this time, there are no customers that would create this condition.

4. Download

Download is the critical part of this website. Its function is to download the photos and videos from a cloud service to the servers while creating the three pug include files that control the two navigation menus, Magnific Popup(photos gallery), and jPlayer(video player). CloudRail-si is the software component used to interface with the 4 cloud services (Microsoft OneDrive, Google Drive, DropBox, and Box)

4.1. CloudRail-si

CloudRail-si is a software component that provides the functionality to download folders and files from different cloud services. Each different cloud service has different interfaces. CloudRail provides a common interface to these cloud services.

CRITICAL: When I first used this component, it was free. Since then, they have started charging for the use of this component. My development websites do not have to pay for the use of this component. However, when customers of mine use this component, Cloud Rail will charge me.

4.2. Download Flow

There are 6 different functions that support the Download:

- downloadAllImages

This function downloads the special files in the MyWebsite folder, as well as everything in the Photos and the Videos folder from the cloud service(cs) to the server file system(fs).

- downloadAllPhotos

This function downloads the Photos folder from the cloud service(cs) to the server file system(fs).

downloadAllVideos

This function downloads the Videos folder from the cloud service(cs) to the server file system(fs).

- downloadNewImages

This function downloads the new folders and photos/videos from the Photos and Videos folders from the cloud service(cs) to the server file system(fs).

- downloadNewPhotos

This function downloads the new folders and photos from the Photos folder from the cloud service(cs) to the server file system(fs).

- downloadNewVideos

This function downloads the new folders and videos from the Videos folder from the cloud service(cs) to the server file system(fs).

There are 2 common functions that support the above 6 functions:

- downloadAll (cs_folder, fs_folder)

This function is passed 2 parameters pointing to a cs folder(source) and fs folder(destination). This function deletes the fs. Then it downloads (recursively) from the cs to the fs.

- DownloadNew (cs_folder, fs_folder)

This function is passed 2 parameters pointing to a cs folder(source) and fs folder(destination). This function downloads (recursively) from the cs and determine if it exits on the fs. If the directory does not exist, it is created in the fs. If a photos/video file does not exist, it is downloaded from cs and saved in fs.

11.1 Thresholds

It was determined during the breadboard testing that the cloud services were generating many errors during the download process. It was determined that the cloud services restricted the rate and volume of data downloaded. The different cloud services have different limiting rates and volumes. I assumed the CloudRail service would take care of this situation. They don't and won't. The main reason I

used Node.JS was because of the asynchronous, non-blocking operations (fast performance). Now I must start slowing things down. I had to implement a Queue-Throttling function to slow things down. Each of the 4 different cloud services uses different rate and volume (thresholds). These rates are also dependent on the file sizes.

5. Cloud Services

Following are the 4 cloud services supported:

- Microsoft OneDrive
- Google Drive
- DropBox
- Box

I have 2 different accounts for each one of these (me and my wife). So, I have 8 cloud services that I am storing my MyWebsite folder and files on.

6. CloudRail-si

CloudRail requires a Client_ID and Client_Secert for each of the different cloud service accounts. This is so the proper authorization (OAuth 2 (https://oauth.net/2/)). To get these codes, an Application Interface (API) project must be defined with each of the cloud services. Each of the cloud service companies provide an online process to create one of these for each of the different Accounts you have with them. This process provides the information required to allow for the Authorization to permit MySite10 to access another person's cloud service files. MySite10 just reads the folders and files within the MyWebsite folder.

Note: MySite10 does NOT make any changes on the Cloud Services. Just Read Folder and Read Files are issued.

There are just three CloudRail functions and a redirectReceiver process.

6.1. Exits Function

This is the function used to connect to the Cloud Service. The "OAuth 2" process will request for the UsersName and Password associated with the Cloud Service. If the connection worked and the folder exits, as successful status will be returned.

6.2. Get Children Function

Returns all folder and files in the folder passed as a parameter.

6.3. Download File Function

Returns the readable stream of the file's content

6.4. RedirectReceiver process

Reference: https://blog.cloudrail.com/simplified-oauth-with-cloudrail/

You must create a RedirectReceiver that you pass on to the service's constructor. The RedirectReceiver will be called by the CloudRail SDK every time it performs OAuth authentication.

7. Throttle Queue

Throttle Queue is the process needed to slow down the 'Get Children' and 'Download File' CloudRail functions, so errors are not generated from the Cloud Services. Each of the different Cloud Services has thresholds. Only so many CloudRail 'Get Children' and 'Download File' functions will be allowed every x milliseconds.

Since the Download function is asynchronous (non-blocking), and reclusiveness is used, there will be many CloudRail functions on the Node.JS stack to be performed. These must go through the ThrottleQueue so they can be performed at a fixed rate (like one every 125ms).

7.1. Throttle Parameters

Each Cloud Service will have a 'Get Children' and 'Download File' throttle parameter. These parameters must be configurable for an installation. The number of folder/files is unknown as is the file sizes.

8. Download Results File

A .csv (comma-separated values) file must be kept at the server for tracking the results of every download function. It must contain (Date/Time, CloudServie, Throttle Parameters, download function, number of bytes transferred, number of different errors if any.

9. Error Log File

An mysiteDebug.log file must be kept at the server for tracking errors. It must contain Date/Time, Domain Name, Module Name, Error/Notification Message.

10. Digital Oceans Server Installation & Configuration

An Installation & Configuration document must exist that defines in detail how to install and configure the server.

10.1. Customer Dependent Information

A Spreadsheet must be maintained that has all customer dependent information:

- Person's Account (Pat & Janet)
- Domain Name
- Digital Oceans root: password; username: password
- Google Analytics ID
- DropBox (Client_ID & Client-Secret)
- OneDrive (Client_ID & Client-Secret)
- Box (Client_ID & Client-Secret)
- Drive (Client_ID & Client-Secret)
- Photos & Videos Username & Password (if used)
- Download Username & Password
- Google gMail, Client_ID, Client_Secert, Refresh Token, Access Token

11. Label 11

12. Label 12