#ALOYE Code Requisition

Given Facts / General Work Scope

1. I have an **Ionic V1 Cordova Application**.
2. Inside the application is a page named **mypage.html**
3. On the mypage.html page there are two text boxes and two buttons
    a. We will call the text boxes **TextBox001** and **TextBox002** for the sake of this requisition.
    b. We will call the buttons **Button001**, and **Button002** for the sake of this requisition.
4. TextBox001 will contain string text that may consist of letters, numbers, and special characters.
5. TextBox002 will contain a password field in the form of a text string that may consist of letters, numbers, and special characters.
6. Button001 is a button that when clicked must do the task set detailed in **Button001 Job** below
7. Button002 is a button that when clicked must do the task set detailed in **Button002 Job** below.
8. **A Web Service will need to be built** that sits in my windows hosting environment and does two things.
    a. We will call these things **Thing001** and **Thing002**.
    b. We will call the task set performed by Thing001, **Thing001 Task Set**
    c. We will call the task set performed by Thing002, **Thing002 Task Set**
9. **HTML code and JavaScript** components must be written to perform the Client Side/App Side **XMLHTTPRequest** Tasks when Button001 and Button002 are clicked.
10. **Web Service C# Code** must be written to perform the Server Side/Web Service Side Tasks when Thing001 and Thing002 are accessed via the XMLHTTPRequest
11. A **Database** will be provided that contains a set of user ids and passwords, to be used by the Web Service to facilitate authentication.

Provided Resources

1. A hosted SQL Server database containing a table with a set of user ids and password, the connection string, and related access material will be provided.

2. A hosted SQL Server database containing a table to be used for storing tokens, the respective userid, password, and time out value in UCT time. The connection string and related access material will be provided.

3. A Visual Studio 2012 Web Service Project containing the web service descriptions will be provided. The developer will get the VS Project as well as access to the URL where it is hosted.

4. The hosted web service space will look something like https://my.webservice.com/webservice.asmx

5. A copy of the Ionic V1 Cordova Project will be provided. For security purposes, the Ionic V1 Cordova Project will contain only the components that are needed to perform the defined tasks.

6. Direct access to Jim Aloye will be provided via telephone and video conference between the hours of 11:00 PM GMT and 5:00 AM GMT

Defined Web Service Component Tasks

Thing001 Task Set - the set of tasks performed by the Web Service to save local files to the cloud.

1. The Thing001 Web Service must receive an XMLHTTPREQUEST that contains a user id, password, and a list of files that can be of an unknown number.

2. The Thing001 Web Service must take the user id and password sent via the XMLHTTPREQUEST and run a query up against a database using a connection string specified in the web.config file to check if the user id and password exist as entered. If they do not the web service must stop and do no more after sending back a message that the user does not exist.

3. If the user id and password exist, the web service must then generate a token, insert it into the Token Table in the Token Database Table, and send back confirmation that the user exists by passing the token to the client via the callback.

4. A new XMLHTTPREQUEST will come in with the token specified above, along with the userid and password, at which point the web service verifies the token is valid and did not expire by querying the Token Database table, along with re-verifying the userid and password, and then must point to a url/file store area on the server and use the user id that is passed to it as the folder in which to direct the xmlhttprequest.

    a. For example if the XMLHTTPREQUEST comes in with a user id of jim, a password of aloye, and a list of files, it must first execute a sql query against a user table in a database to verify that the passed user id and password exist.

    b. If they exist, the web service must then send the callback w/ token and point to a destination of https://www.mywebsite.com/Jim for that token.

        i. If the passed user id is billybob3 then the destination folder is https://www.mywebsite.com/billybob3

ii. This destination is best understood to be https://www.mywebsite.com/<passeduserid> Where <passeduserid> is whatever the user id is that is passed in the XMLHTTPREQUEST.

c. The token must expire after after 10 minutes, or when the entire process is complete.

5. The Thing001 Web Service must first delete any and all files that currently exist in the backup folder located at https://www.mywebsite.com/<passeduserid>

6. Then the Thing001 Web Service must copy all the files that currently exist in the <passeduserid> folder into a sub folder named https://www.mywebsite.com/<passeduserid>/backup

    a. If no files exist in the <passeduserid> folder, then no process of copying them to the backup subfolder is necessary.

7. The Thing001 Web Service must then verify that the files have been copied successfully and close any locks it may have on them.

8. The Thing001 Web Service must then delete all the files in https://www.mywebsite.com/<passeduserid> folder, but not touch, alter, or delete the /backup folder or its contents in any way.

9. Then the Thing001 Web Service must take the files it receives from the XMLHTTPREQUEST and save them all to the <passeduserid> folder

10. Next, the Thing001 Web Service must verify that the files were saved and are fully intact (via checksum? Or other means)

11. Then the Thing001 Web Service must return a success message via the callback confirming the files were saved.

12. If a failure occurs, the Thing001 Web Service must delete the debris from the <passeduserid> folder and restore the files from the backup folder to the parent <passeduserid> folder, verifying that the files have copied successfully. There is no need to delete the files in the backup folder. Leave them there for added protection since a failure has occurred.

Thing002 Task Set - the set of tasks performed by the Web Service to copy the cloud files locally.

1. The Thing002 Web Service must receive an XMLHTTPREQUEST that contains a user id, password, and a list of files that can be of an unknown number.

2. The Thing002 Web Service must take the user id and password sent via the XMLHTTPREQUEST and run a query up against a database using a connection string specified in the web.config file to check if the user id and password exist as entered. If they do not the web service must stop and do no more after sending back a message that the user does not exist.

3. If the user id and password exist, the web service must then send back confirmation that the user exists by generating a token, storing it in the Token table along with the userid, password, and time out period in UTC time, then passing token back to the client via the callback.

4. A new XMLHTTPREQUEST will come in with the token specified above, along with the userid and password, at which point the web service must verify the token, and re-verify the userid and password, and then point to a url/file store area on the server and use the user id that is passed to it as the folder in which to direct the xmlhttprequest.

    a. For example if the XMLHTTPREQUEST comes in with a user id of jim, a password of aloye, and a list of files, it must first execute a sql query against a user table in a database to verify that the passed user id and password exist.

    b. If they exist, the web service must then point to a destination of https://my.website.com/<passeduserid> where passeduserid is the user id that was passed via the XMLHTTPREQUEST

5. The Thing002 Web Service must then pass/copy/download all the files from the <passeduserid> folder to the local folder which sits at myapp.app/www/myfolder1/www/savefolder1

    a. It must do this AFTER first backing up and deleting the contents of the local save folder as detailed in the Button002 section below.

6. A combination of client technology and the Thing002 Web Service must verify that the files have been copied successfully.

7. If a failure occurs, the client that interacts with the Thing002 Web Service must delete the debris from the myapp.app/www/myfolder1/www/savefolder1 folder and restore the files from the savefolder2 folder to the savefolder1 folder, verifying that the files have copied successfully. There is no need to delete the files in the savefolder2 folder. Leave them there for added protection since a failure has occurred. Then return a message to the user via the mypage.html that the save to device from cloud failed.

Defined Client Side Component Tasks

Button001 Task Set - the set of tasks performed by the Web Service to save local files to the cloud.

1. Save.html is located in myapp.app/www/myfolder1/www/save.html and will be the client side page we work with.

2. When Button001 is pressed a function must copy all the files from myapp.app/www/myfolder1/www/savefolder1 to a list or an array or something of that nature.

3. The client must verify that the files are complete and fully intact. Then it must hold the files until it is time to pass it to the cloud via Thing001 Web Service.

4. Then the client function must check for and verify internet connectivity exists.

    a. If internet connectivity exists, it must connect to the Thing001 Web Service listed above and pass it the values from TextField001 and TextField002 via XMLHTTPREQUEST

5. Once the Thing001 Web Service verifies the user and password exist, and passes confirmation back along with a token, the client must then pass another XMLHTTPREQUEST to the Thing001 Web Service, containing the token, userid, password, and the files it collected and stored previously.

6. The Thing001 Web Service must receive the files and save them to the server per the Thing001 Web Service description above, then pass back confirmation/verification via the callback function to the client.

7. Once the client receives verification that all the files were saved successfully, it must notify the user on the save.html page that the upload to cloud successfully completed.

8. If a failure has occurred display a message with that info on the mypage.html page.

Button002 Task Set - the set of tasks performed by the Web Service to save cloud files to the local storage in the app.

1. Save.html is located in myapp.app/www/myfolder1/www/save.html and will be the client side page we work with.

2. When Button002 is pressed a function must check for and verify internet connectivity exists.

3. If internet connectivity exists, it must connect to the Thing002 Web Service and pass the userid and password.

4. Once the Thing002 Web Service verifies the user and password exist, the Thing002 Web Service must retrieve a copy of all the files located in the directory on the server which is the same name as the <passeduserid>, but not write them anywhere yet. It must hold them until we are ready to save them.

5. Next, the Thing002 Web Service must pass the files to the client via the callback.

6. The client must receive them and verify the are fully intact (run a checksum? Or some similar process).

7. After verification, the client must first delete all the files located in myapp.app/www/myfolder1/www/savefolder2

8. Then it must verify the folder is blank.

9. Once deleted it must then copy all of the files from myapp.app/www/myfolder1/www/savefolder1 to myapp.app/www/myfolder1/www/savefolder2 as a kind of backup.

10. Then it must verify that the contents of savefolder2 is identical to savefolder1

11. If anything bad happened, it must remediate the issue and copy the missing files so that save folder2 contains a copy of all the files in savefolder1

12. Once done, it must delete all the files in savefolder1

13. Then it must take the files it has been holding when it retrieved a copy of the files from the server, and write them to savefolder1

14. Then it must verify that the save process is successfully and all the files have been written.

15. Then it must close any locks it may have on the files and notify the user the files have been restored successfully

16. If a failure has occurred, it must delete the debris from savefolder1 and then copy all the files from savefolder2 back into savefolder1. There is no need to delete anything from savefolder2 at this point. Leave the files there as an extra precaution since the save to device process failed. As always, display a message on mypage.html that an error has occurred.