|  |  |
| --- | --- |
| **Team:** | **Team C** |
| **Team Members:** | **Jeremy Adams, Taunyl Bailey, Tim Olson, Rachel Spiegelhoff** |
| **Date:** | **7/26/14** |
| **Project Title:** | **Taylor’s Professional Services – Web Site eCommerce** |
| **Team Leader:** | **Tim Olson** |

# Scope

This section will define the scope of the project by defining and describing the System, the major functions of the application, and the database.

## System Description

Taylor’s Professional Services has requested a web site with the following features:

* Allow clients to complete staffing request online
* Provide clients with a listing of potential candidates
* Provide automated response to client that staffing request will be validated in 24 hours
* Allow TPS staff members to update their resumes and pictures online

The completed TPS system will have three types of accounts, Client, Staff and Manager. Only clients with valid contract numbers and passwords will be allowed to enter the client area and only staff with valid employee numbers and passwords will be allowed to enter the staff area. Only contract managers will have access to both locations.

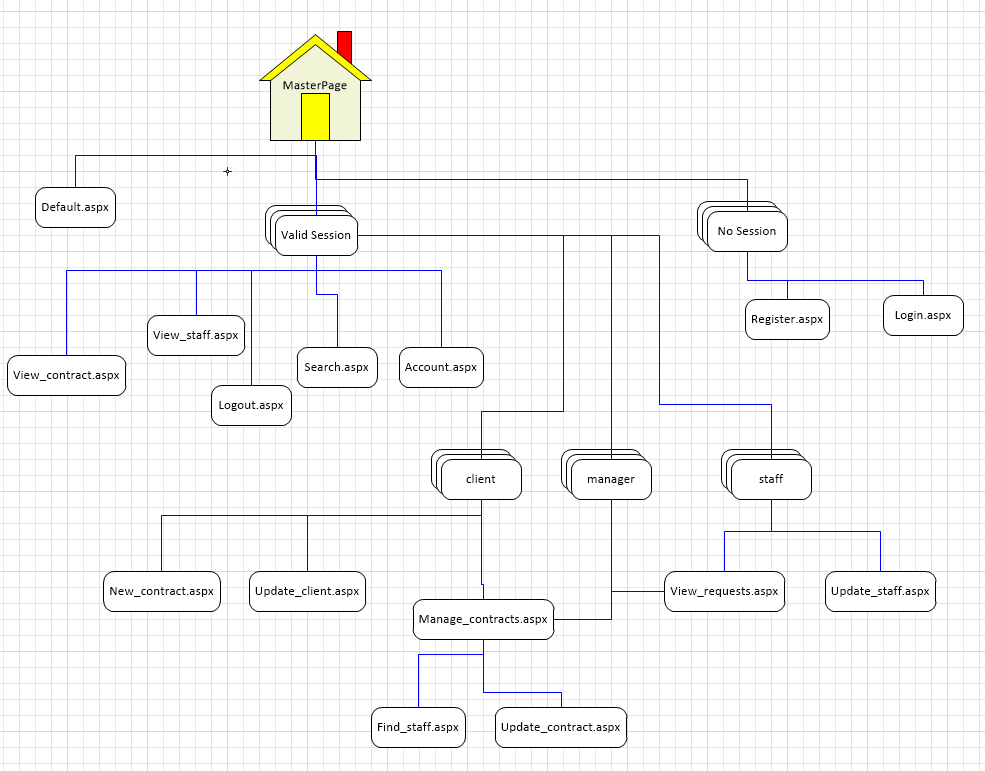
The finished website will be hosted on a GoDaddy windows IIS server.

Databases will be also hosted by GoDaddy using MSSQL.

Files can be uploaded using team member’s ftp accounts.

Database changes can be made using SQL scripts.

## Major Software Functions



1.2.1

1.2.4

1.2.3

1.2.2

1.2.10

1.2.6

1.2.15

1.2.5

1.2.9

1.2.8

1.2.11

1.2.14

1.2.7

1.2.13

1.2.12

### MasterPage.master - Provides a template (styling and navigation) for all content pages

1.2.2 Register.aspx – Allow new user to create an account on the database so they can access more content

1.2.3 Login.aspx – allows authentication of users

1.2.4 Default.aspx – Default homepage visible to all, gives information about what TPS is.

1.2.5 Account.aspx – Allow users to modify account information such as username, password, email and phone number.

1.2.6 View\_staff.aspx – Allows valid users to see details about a specific staff member

1.2.7 Update\_staff.aspx – allows staff members to modify the information that is displayed to others such as there resume and profile picture.

1.2.8 Update\_client.aspx – allows clients to change how there company name is displayed

1.2.9 New\_contract.aspx – allows clients to create new contracts (jobs)

1.2.10 View\_contract.aspx – display details about a specific contract to verified users

1.2.11 Manage\_contract.aspx – allow clients and managers to see what contracts are needing to be filled and what requests have been accepted

1.2.12 Update\_contract.aspx – allows clients to modify contract details

1.2.13 Find\_staff.aspx – allows clients to find staff members matching a specific contracts requirements

1.2.14 View\_requests.aspx – allow staff and managers to see/accept/remove requests made by clients

1.2.15 Search.aspx – allow users to search for specific contracts or staff members

## Database Description

The type of database that will be used for the website is MSSQL.

We have selected this because GoDaddy servers already offer MSSQL and MySQL databases to use with your hosting account.

Since we are using ASP.NET implementing a MSSQL database would require less configuration time to setup and use.

Microsoft SQL Server Management Studio will be used to run scripts that can modify the database structure and to create backup files of the database.

## Design Constraints and Limitations

The following are the design constraints and limitations for this project:

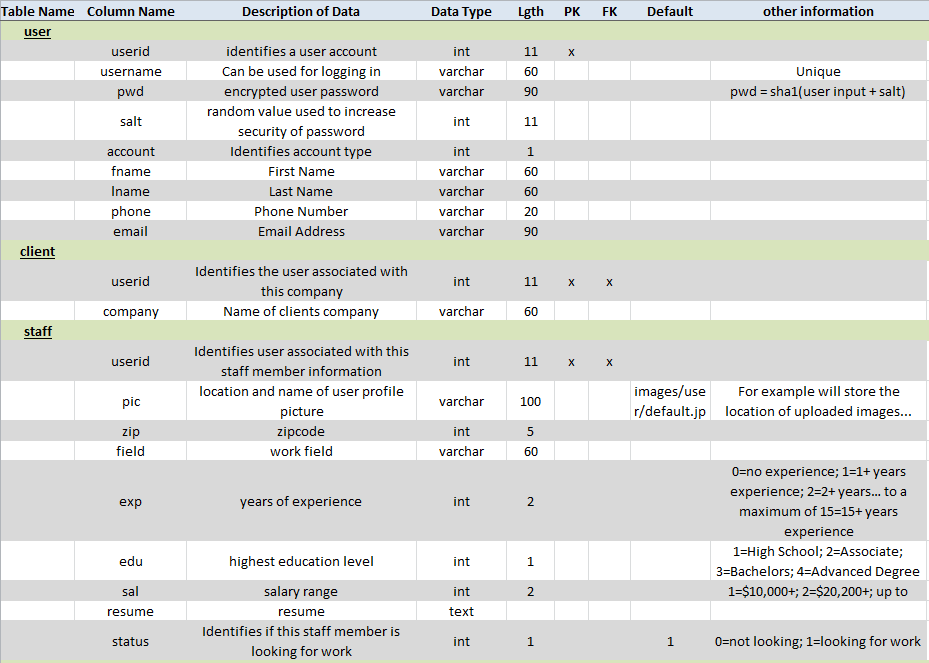
* Short period of time to get everything designed, coded and working
* Using GoDaddy server that only supports ASP.Net 4.0 and not ASP.Net 4.5
* Only 4 team members instead of the original 5 we were supposed to have
* Coordinating between the team members different schedules and timezones
* Authorization is handled by using Sessions and has already been built into the code. There are three sessions types that can be used to check authorization if new pages are to be added.
  + Session[“account”]
  + Session[“userid”]
  + Session[“username”]
* Sessions is the only way to identify the account type (manager, client, staff), user ID and the user name of the current user.
* When pulling passwords from the database the only way to check to see what the value is, is to encrypt using SHA1 a text value with the salt that was generated for the user and see if the newly encrypted value matches the encrypted value of the database.
  + Because of this password recovery is not possible, only password reset. The new generated password can be emailed to the user so they can regain access to their account.
* PHP is supported but the original system was designed for using ASP.Net 4.0 and C#. So it is recommended that these are the main languages to be used for any updates later.
* A current version of CSS must be supported for the webpage design to be displayed properly.
* JavaScript code must be supported to run scripts such as the password strength checker. Although the webpage could run fine if this was removed.
* The google Ajax/Jquery source code is included in the MasterPage so any Ajax or Jquery code will run fine in any child document.

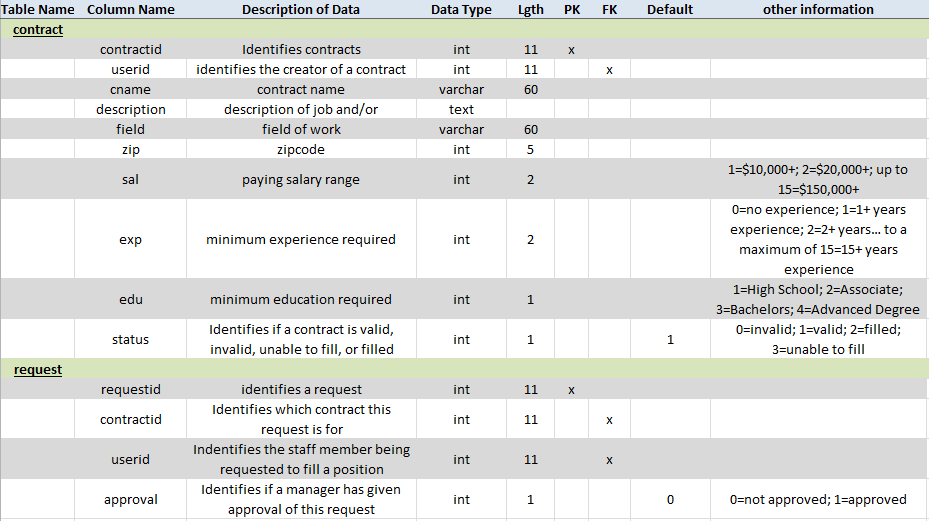
# Design Description

This section will provide an overview of the application design by providing a description of the data to be used and managed by the application, define the flow of data through the system, describe the architecture of the programs within the application and define the how the components will interface.

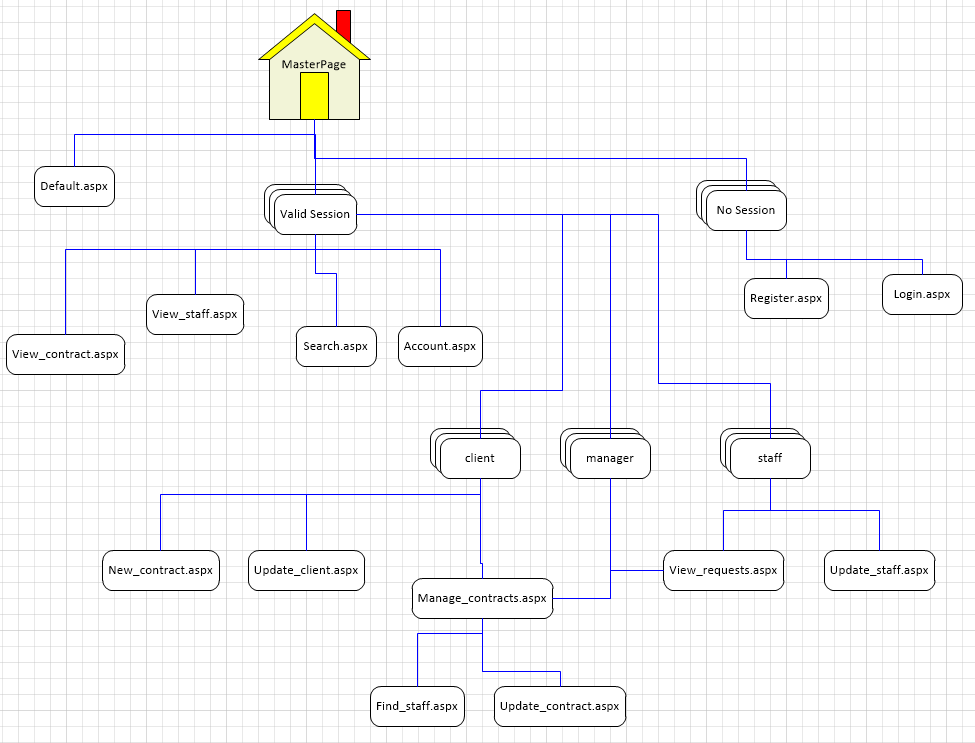
## Data Description







## Data Flow



## Program Architecture

**User Login:** This is where a user enters their login credentials. Then the login module checks the entered credentials against the database and allows the user to enter the requested area if their credentials are valid. If the credentials are invalid the user will receive an error message.

**Client Area:** The client area allows a client to enter a staff request and retrieve a staff request. All requests will be validated and checked against the database. The Contract Manager can login to the client area and retrieve contract information, validate staffing requests, and close staffing requests.

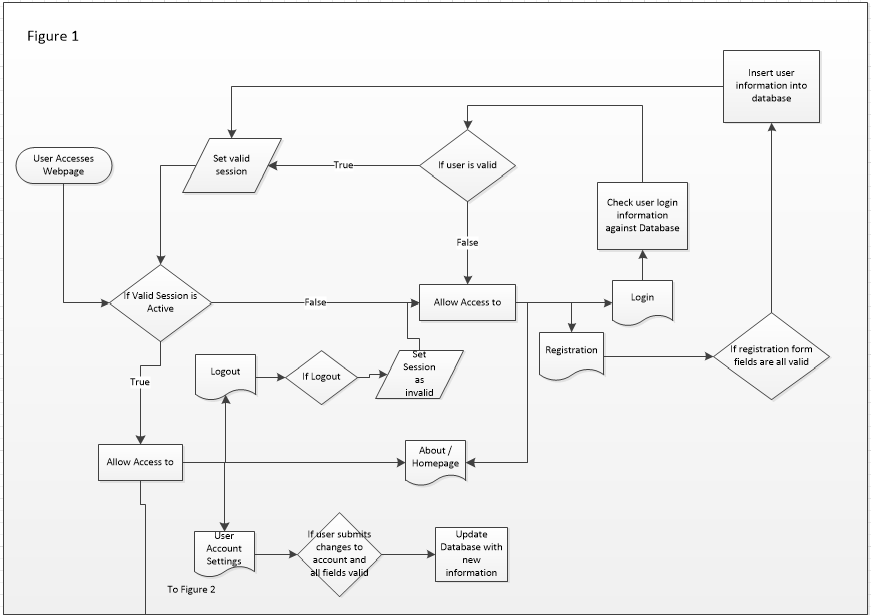
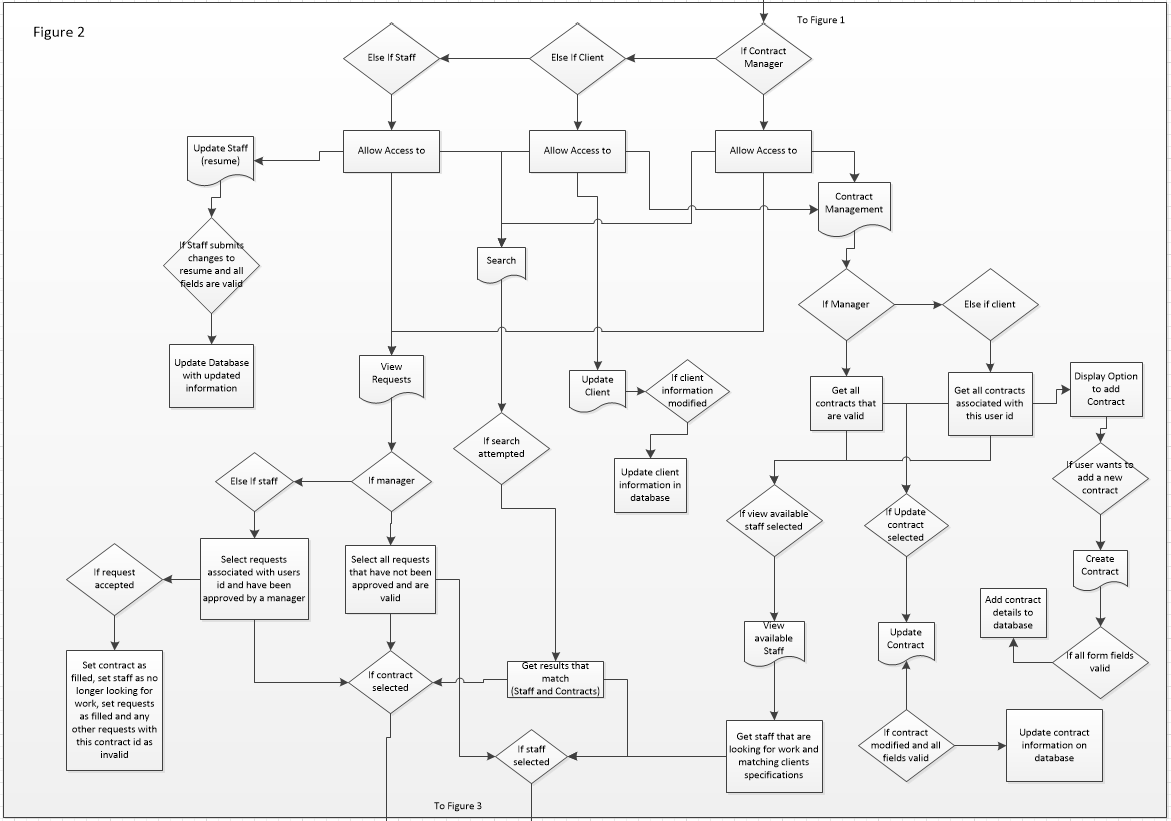
**Staff Area:** The staff area allows a staff member to update their personal information, resume, pictures, etc. The Contract Manager is also allowed to view, edit, and update staff information. All requests are validated and updated within the database.

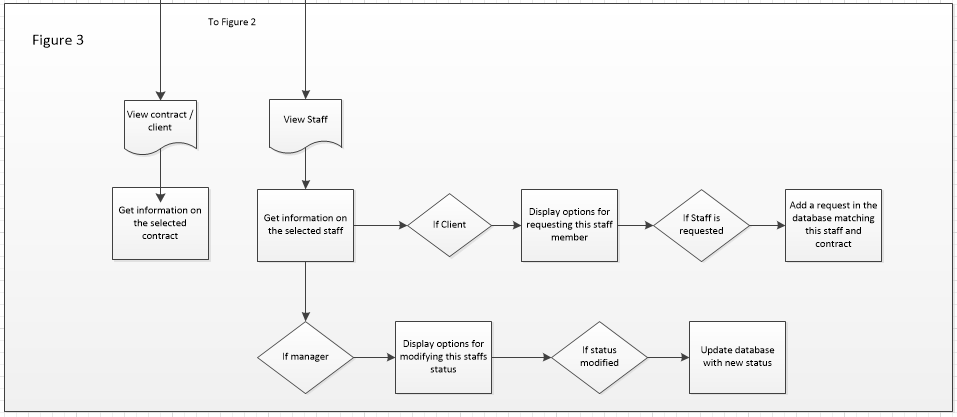
**Logoff Area:** The Client and Staff areas allow the users to log out.

All data retrieved and entered is added to or pulled from the database.

The data flow determines how the database tables will logically pass data between them.

The client, staff, and contract manager all have access to the TPS site through the defined flow diagram, determining which tables and data have access to each other.



## Component Interfaces



|  |  |
| --- | --- |
| **Page Title-Name** | **Description** |
| Registration Page | Allows the user to choose to fill out the new staff member registration form and new user login information and then submit the completed information, which will result in the user being automatically sent to the Verify Staff Information Page, or use the supplied link to return back to the Home Page. |
| Login Page | Allows the staff member, client or contract manager the option to login via a predetermined username and password that is authorized by role. |
| Main Page | Once logged in, this page gives the user a list of authorized areas they can visit.  For staff members their choices include the Home Page and the Update Staff Member Information Page, View Contracts, Find Staff and Search.  For clients, their choices include the Home Page, Find Staff, View Staff, and Search.  For the contract manager, their choices include all the above (both the staff and client sections) as well as links to Update Client, Update Staff, View Request, New Contract, Manage Contract, Update Contract and Update Staff. |
| Update Staff Page | Allows the user (staff member and/or contract manager) to search for specific staff members and then displays that specific record/file in order to update/edit/delete their personal staff information, resume, availability status, and/or photos. |
| User Registration Page | Allows theuser (client) to fill out the new staff request order form. Once completed the user has option to either “Submit” the form, which will result in the system providing an automated response stating that the contract manager will validate their request within 24 hours of receipt, or to “Cancel” the order. |
| View Staff Page | Allows theuser (client or contract manager) to search for and view specific staff request. Once submitted, all staff request information along with a field that states whether the staff request is valid, invalid, unable to fill, or filled is displayed.  The View Staff Page also displays links back to the, Main Page, New Staff Page, Search Page, and Login Page for ease of use. |
| Search Staff Page | Allows theuser (client or contract manager) to search for and view a list of potential candidates based on experience, education, salary, and location. Then be able to choose up to three potential candidates, and submit the selected potential candidates to the contract manager.  The Search Staff Information Page also displays links back to the Main Page, New Staff Page, View Staff Page, and Login Page for ease of use. |
| Update Staff Page | Allows the contract manager to state whether the staff request is valid, invalid, unable to fill, or filled.  The Update Staff Page also displays links back to the Main Page, Login Page, Update Staff Page, New Staff Page, View Staff Page, View Request Page, and Search Page displayed for ease of use. |
| View Requests Page | Allows the contract manager to view the list/chart of selected potential staff by staff request number, displaying staff members along with the Field of work, Company Name, and Pay Range.  In addition, the View Request Page also displays links back to the Main Page, Login Page, Update Staff Page, New Staff Page, View Staff Page, View Contract Page, and Search Page are displayed for ease of use. |
| New Contract Page | Allows the contract manager to enter new client contract information into the database. This includes information such as Field of work, Company Name, Staff ID, Contract ID, Pay Range, Contract Date, Experience, Status Education and Job Description. Once the contract manager selects the submit button the contract information will be saved to the database.  In addition, the New Contract Page also displays links back to the Main Page, Login Page, Manage Contract Page, Update Contract Page, View Contract Page, Update Staff Page, New Staff Page, View Staff Page, View Contract Page, and Search Page are displayed for ease of use. |
| View Contract Page | Allows the contract manager to view selected contracts based upon the Contract ID and or Company Name.  In addition, the New Contract Page also displays links back to the Main Page, Login Page, Manage Contract Page, Update Contract Page, View Contract Page, Update Staff Page, New Staff Page, View Staff Page, View Contract Page, and Search Page are displayed for ease of use. |
| Update Contract Page | Allows the contract manager to update selected contracts based upon the search by Contract ID, Company Name. You will be able to edit the contract.  In addition, the New Contract Page also displays links back to the Main Page, Login Page, Manage Contract Page, Update Contract Page, View Contract Page, Update Staff Page, New Staff Page, View Staff Page, View Contract Page, and Search Page are displayed for ease of use. |
| Manage Contract Page | Allows the contract manager to update, edit or delete contracts based upon a search by Contract ID or Company Name and Staff ID.  In addition, the New Contract Page also displays links back to the Main Page, Login Page, Manage Contract Page, Update Contract Page, View Contract Page, Update Staff Page, New Staff Page, View Staff Page, View Contract Page, and Search Page are displayed for ease of use. |

| Event | Description/Summary | |
| --- | --- |
| **Event Attributes** | **Event Name and Description** |
| *submit( )* | Called when adding the value of a variable for all members of the group specified by this function's argument. |
| *search( )* | Called when searching a variable from all members of the group specified by this function's argument. |
| *update( )* | Called when replacing the value of a variable for all members of the group specified by this function's argument. |
| *cancel( )* | Called when cancelling a variable from all members of the group specified by this function's argument. |
| *delete( )* | Called when deleting a variable from all members of the group specified by this function's argument. |

# Detailed Design

This section will define and describe the design details for each of the components in the application.

## *Default.aspx*

### Processing Description

Default.aspx

Home button clicked

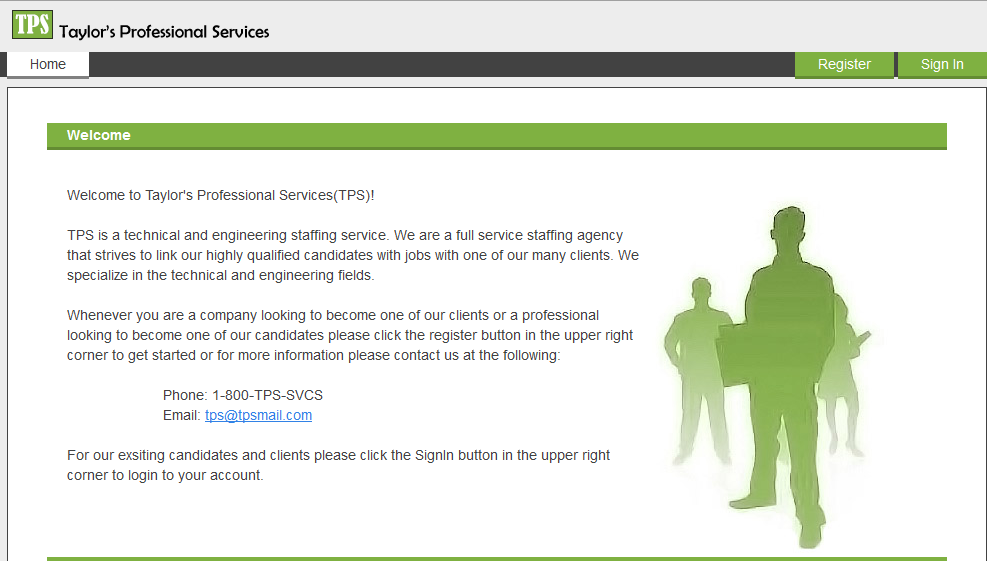
login.aspx

Sign In button clicked

login.aspx

Sign In button clicked

### Interface Description



Register button goes to the register page

Home button goes to default page

Sign In button goes to login page

### Pseudocode

Register button clicked redirect to register.aspx

Sign In button clicked redirect to login.aspx

Home button clicked redirect to default.aspx

## *Login.aspx*

### Processing Description

Login button clicked

Invalid login error displayed

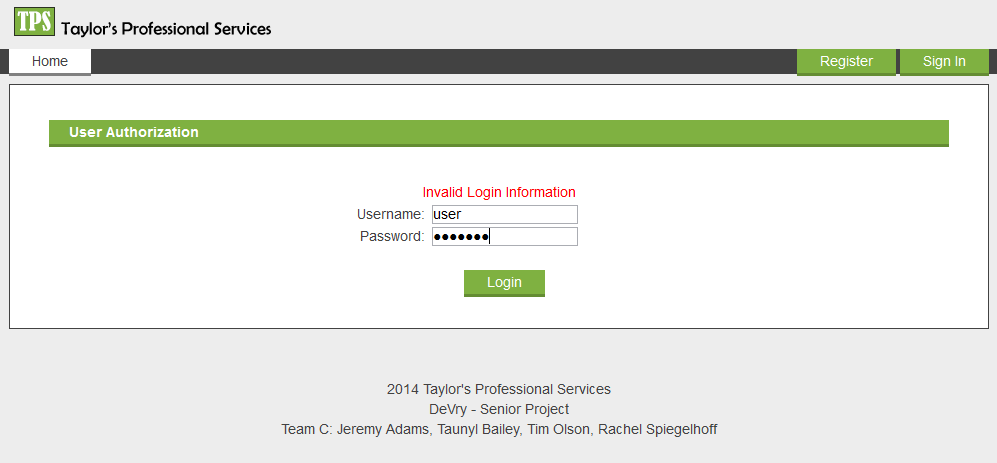
User logged in

User enters username & password

Incorrect login info entered

Correct login info entered

### Interface Description



User Input Fields

Login button

Error Label

### Pseudocode

On page load

hide error label

When login Button clicked

if username and password fields are not empty then

Get password and salt for that username from DB

if username found then

Encrypt inputted password with the users salt

if password matches password in db then

Set valid Session data

else

Set Error Label Text and display

else

Set error label text and display

else

Set error label text and display

### Modules Uses

HashPass( password, salt)

encrypt password and salt using SHA1

return encrypted password

GetConnectionString()

return connection string to DB

## *Register.aspx*

### Processing Description

Submit button clicked

Form validation results displayed

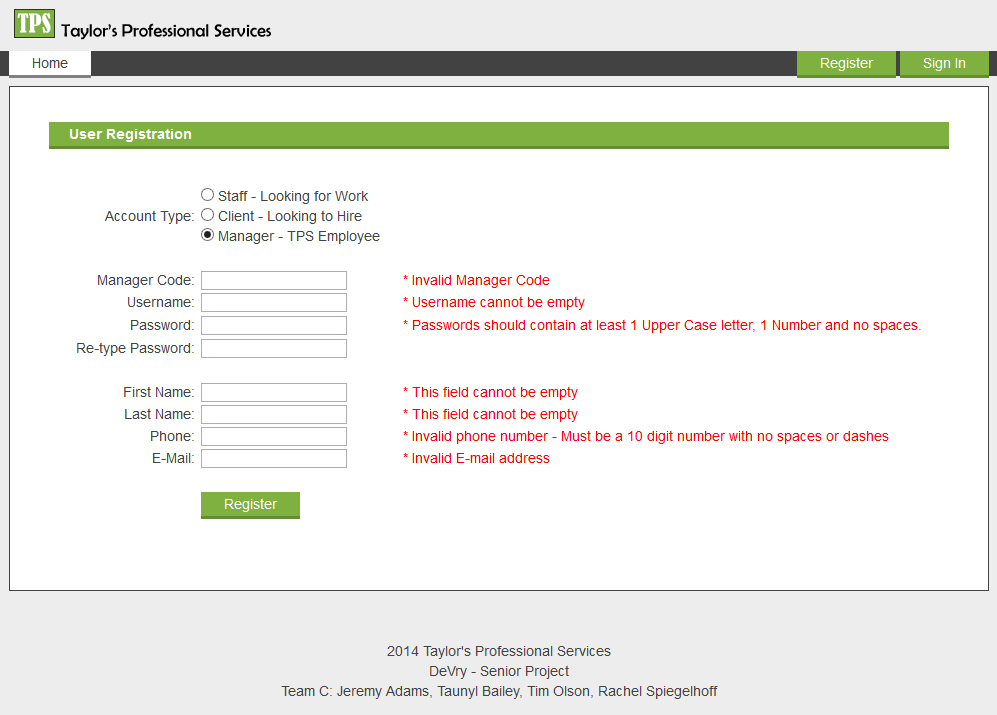
User is registered

User inputs fields

Missing info

All Info entered

### Interface Description



Form Validation Results

Attempts to submit form

User input fields

Only visible if account type is Manager

### Pseudocode

On Page load

if session is not empty then

redirect to homepage

else

hide manager code fields if manager radio button not selected

when register button clicked

if all fields are valid then

create random salt from current time

call encrypt function with password text field and salt as arguments

set account = 3

set account name = “”

if manager radio checked then set account = 1

else if client radio checked then set account = 2

INSERT INTO [user] username, encrypted password, generated salt, account, fname, lname, phone and email

If data successfully inserted into db then

Select userid from db where username = inputted username

if account = 3 then

Insert into staff userid

set valid session data

redirect to update\_staff

else if account = 2

Insert into client userid

set valid session data

redirect to update\_client

else // manager

set valid session data

redirect to view\_requests

else

display error message

else

display error message

### Modules Uses

GetConnectionString()

return connection string to DB

HashPass( password, salt)

encrypt password and salt using SHA1

return encrypted password

RadioAccount\_CheckedChanged()

if manager radio checked then display manager code field

else hide manager code field

checkManagerCode()

if manager radio checked and manager code != cis470

IsValid = false

else

isValid = true;

validateUsername()

if username length > 0

check database if name already exists

if username already exists

IsValid = false

set validator Text showing name is not available

else

IsValid = true

else

IsValid = false

set validator text showing that this field cannot be empty

checkPasswordStrength()

if password length < 6 || (no uppercase letter || no lowercase letters || no numbers) then

IsValid = false

set validator text showing that password is too weak

else

isValid = true

PasswordMatchValidator

if retyped password does not match validated password then

IsValid = false

display error message

FirstNameValidator()

if first name length < 1 then isValid = false

LastNameValidator()

if last name length < 1 then isValid = false

PhoneValidator()

if phone length < 10 then isValid = false

EmailValidator()

if !IsValidEmail(inputed email) then isValid = false

bool IsValidEmail(email)

if email is valid return true

else return false

## *Account.aspx*

### Processing Description

Update button clicked

Form validation results displayed

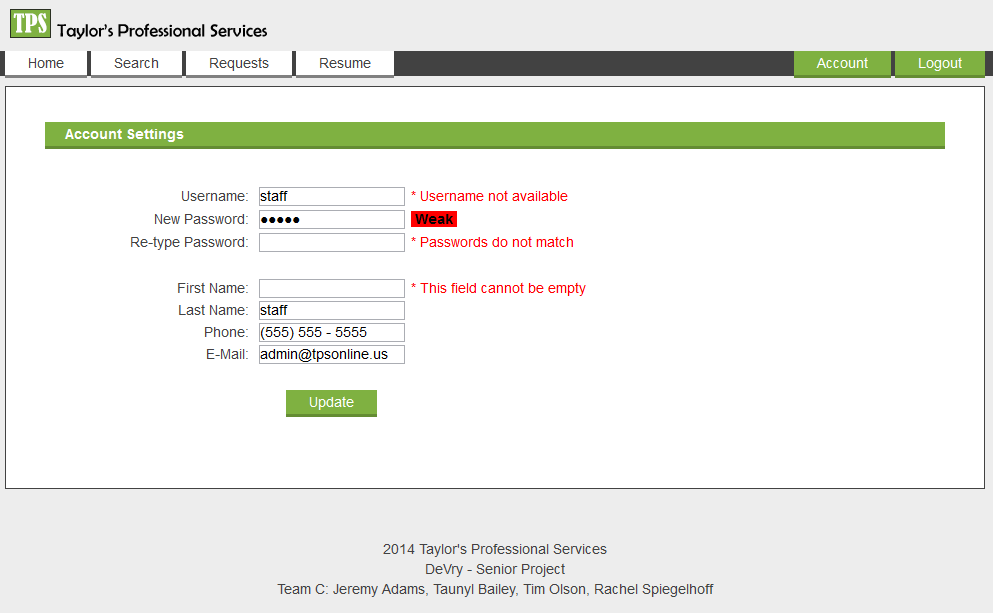
Account updated

User enters account info

Weak passwords or passwords do not match or info entered incorrectly

All info entered correctly

### Interface Description



User Modifiable fields

Validation Results

Update Button

JavaScript Interactive Password Strength Checker

### Pseudocode

On Page Load

if Session is empty

redirect to login

else

if !IsPostBack

SELECT \* From [user] where userid = @userid

if data found then

set text field values to match database

else

destroy session

redirect to login

when Update Button clicked

if IsValid

if user attempting to change password then

generate new salt from current time

encrypt new password and salt

Update [user] with new information

if update successful

reset session with new information

display success message

else

display error message

### Modules Uses

GetConnectionString()

return connection string to DB

HashPass( password, salt)

encrypt password and salt using SHA1

return encrypted password

validateUsername()

if username length > 0

check database if name already exists

if username already exists

IsValid = false

set validator Text showing name is not available

else

IsValid = true

else

IsValid = false

set validator text showing that this field cannot be empty

checkPasswordStrength()

if password length < 6 || (no uppercase letter || no lowercase letters || no numbers) then

IsValid = false

set validator text showing that password is too weak

else

isValid = true

PasswordMatchValidator

if retyped password does not match validated password then

IsValid = false

display error message

FirstNameValidator()

if first name length < 1 then isValid = false

LastNameValidator()

if last name length < 1 then isValid = false

PhoneValidator()

if phone length < 10 then isValid = false

EmailValidator()

if !IsValidEmail(inputed email) then isValid = false

bool IsValidEmail(email)

if email is valid return true

else return false

## *View\_Staff.aspx*

### Processing Description

Data Found

Redirect to login

Display error message

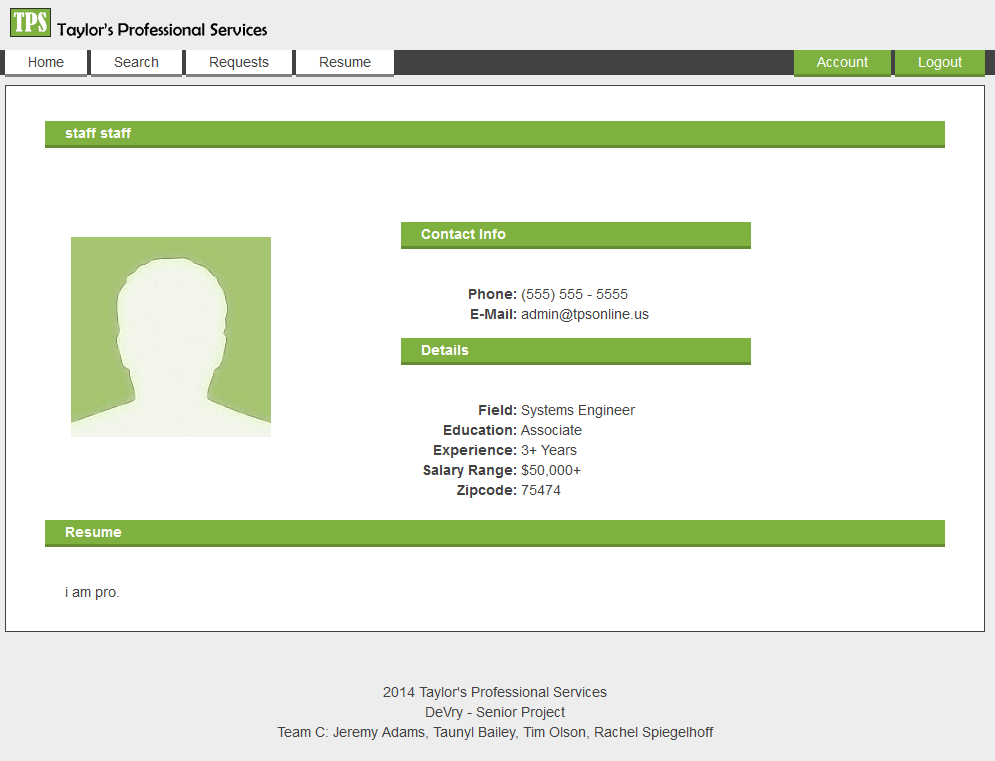
Page Loads

Display data

No data found

### Interface Description

Staff member’s name



Current profile picture

Other Staff information

Contact Info

Resume

### Pseudocode

on Page Load

if session != empty then

if Get[“id”] != empty then

SELECT [user]\*, staff.\*, WHERE userid = Get[“id”] and account = 3

if data found then

display staff information

if Session type == manager then

display option to activate/deactivate account

elseif Session type == client then

if client has valid contracts then

display option to request this staff

else

display invalid request message

else

display invalid request message

else

redirect to login

### Modules Uses

GetConnectionString()

return connection string to DB

Button\_Client\_Request\_Click()

Select \* from [request] where userid = Get[“id”] and contractid = selectedContractid

if results are returned then

display staff already requested message

else

Get number of request made for this contract

if requests > 2

display max requests already made message

else

Insert new request into database for this user and contract

if request inserted into db successful

display success message

else

display failed message

btn\_setActive()

Update [staff] set status = 1 where userid = get[id]

refresh page to display updated results

btn\_SetInactive

Update [staff] set status = 0 where userid = get[id]

refresh page to display updated results

## *Update\_Staff.aspx*

### Processing Description

Redirect to account.aspx

Clicks uploads

Opens windows explorer

Account info edit button clicked

User chooses file to upload

Browse button clicked

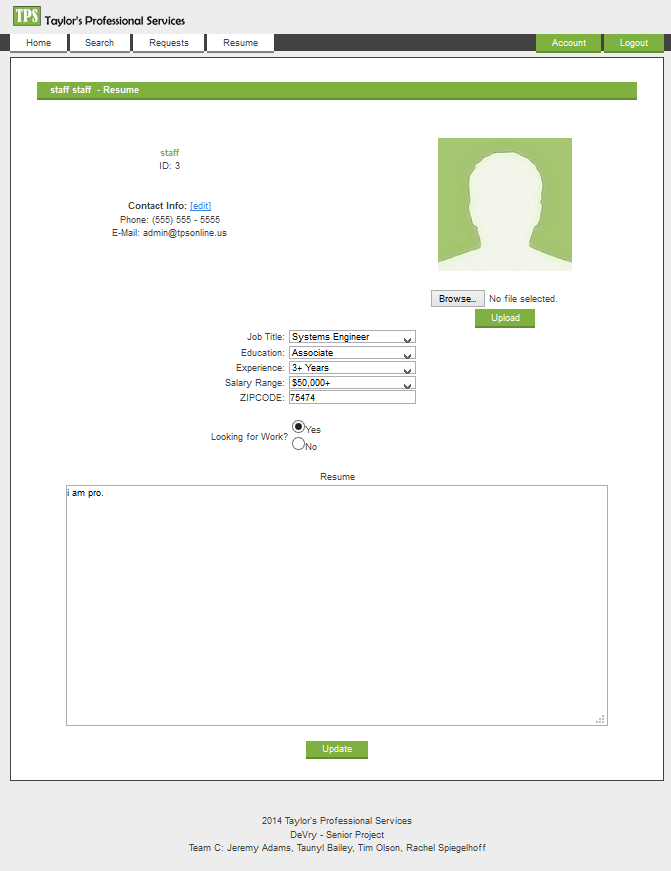
Image uploaded

All info entered

Update button clicked

Info uploaded to database

### Interface Description



Redirect user to account.aspx

Attempts to submit information to database

Multiline text field were user can type up a resume

Toggle profile visibility

Information drop down options

Will attempt to upload selected image

Image select field

Current Profile Picture

Account information

### Pseudocode

On Page Load

if session == empty then

redirect to login

else if session type != staff then

redirect to homepage

else

if !IsPostBack then

select \* from staff where userid = session[userid]

if results found then

set form fields with these values

else

destroy session

redirect to login

### Modules Uses

GetConnectionString()

return connection string to DB

BtnUpload\_clicked()

if file field != empty then

get file size

if filesize == 0

display no image uploaded message

return

get file type

if filetype != .jpg, .png, .gif, or .bmp then

display invalid format message

return

while filename exists on server

append filename with counter

save file to server

try to create bmp image with file and save a resized image

if unable to create bmp with file then

display error message

else

display success message

update staff set pic = uploaded image

if old profile picture != Default and exists on server

delete old image

Delete original uploaded file from server

## *Update\_Client.aspx*

### Processing Description

Update button clicked

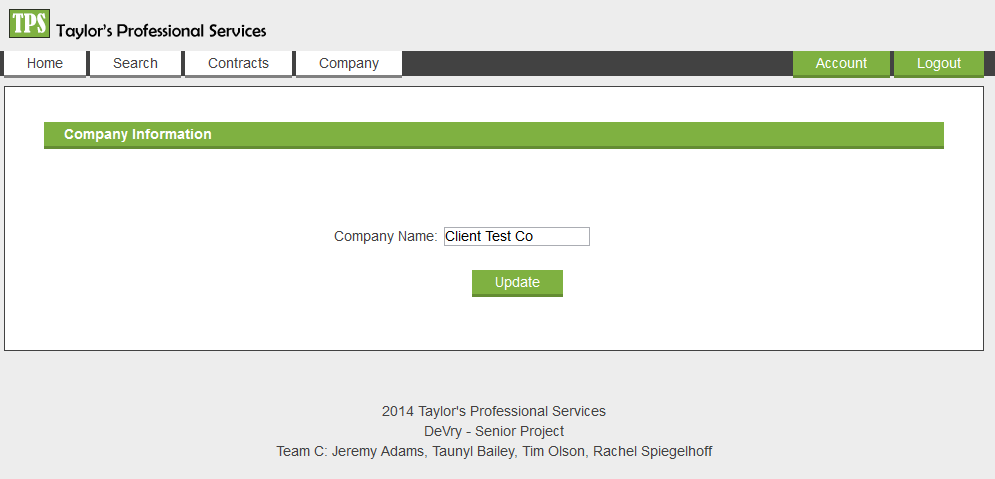
Info updated in database

User updates company info

Info present

No info entered

### Interface Description



User modifiable text field

Submit Button

### Pseudocode

On Page load

If session != null && session type == client then

if !IsPostBack

select \* from [client] where userid = session[userid]

if data found then

set form fields with this data

else

destroy session

redirect to homepage

else if session != empty

redirect to homepage

else

redirect to login

### Modules Uses

GetConnectionString()

return connection string to DB

Button\_Update\_Click()

if IsValid then

update [client] with textfield values where userid = session[userid]

if update successful then

display success message

else

display error message

## *New\_Contract.aspx*

### Processing Description

This module is used to create new contracts for various companies the client is working for (e.g. Company Name, Contract ID, Pay Rate, etc) and store this information in the database table.

Information updated to database.

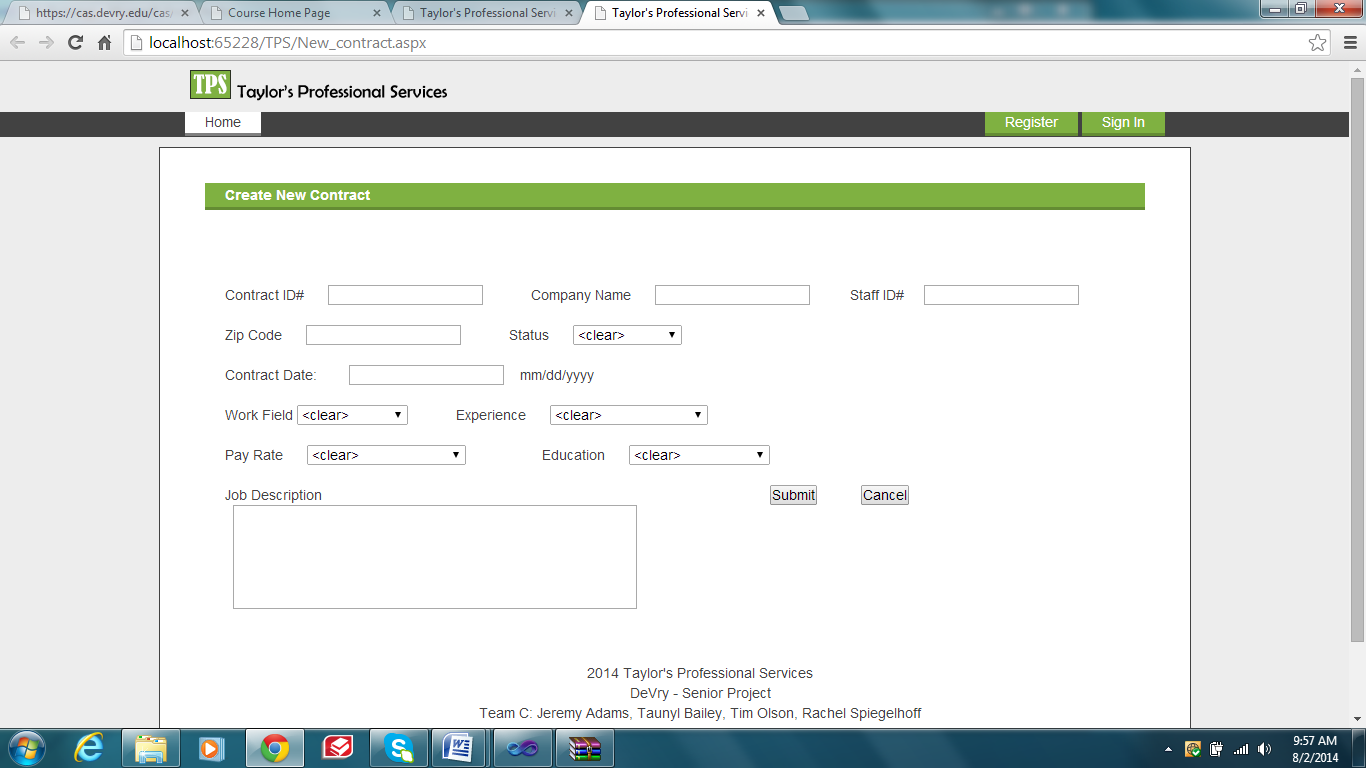
Missing fields/invalid information

All required fields are filled out.

Submit button is clicked.

Contract Manager enters information into fields.

### Interface Description



Enter Valid Zip Code, Status, and Contract Date

Enter contract Pay Rate and Education

Enter current Work Field, and Experience

Enter Contract ID, Company Name and Staff ID

Located just below the header banner containing the company name and logo, is a hyper link that allows the user the choice to go back to the “Home Page”. The “Create New Contract” page contains text boxes for the company name, contract id, staff id, zip code, and date. In addition, the “Create New Contract” also includes drop down lists for the status choices, pay range, education, experience, a work field type choices. Located at the bottom of the form is a “Submit” button and a “Clear” button. The footer contains the team member information”.

### Pseudocode

protected void btnSubmit\_Click(object sender, EventArgs e)

// Get each value from the data entry fields

// Get values from the data entry fields in session state

string businessName = txtCname.Text.Trim();

string contractID = txtContractid.Text.Trim();

string staffid = txtUserid.Text.Trim();

string zipcode = txtZipcode.Text.Trim();

string street = txtContractdate.Text.Trim();

string job = txtDescription.Text.Trim();

string contractdate = txtContractdate.Text.Trim();

// Remove the previous yellow

txtCname.BackColor = System.Drawing.Color.White;

txtContractid.BackColor = System.Drawing.Color.White;

txtUserid.BackColor = System.Drawing.Color.White;

txtZipcode.BackColor = System.Drawing.Color.White;

txtDescription.BackColor = System.Drawing.Color.White;

txtContractdate.BackColor = System.Drawing.Color.White;

//Add validation codes for CompanyName, Contract ID, Staff ID and Contract Date -- Checks for empty or filled with spaces

if (string.IsNullOrEmpty(businessName) && string.IsNullOrEmpty(contractID) && string.IsNullOrEmpty(staffid))

txtContractid.BackColor = System.Drawing.Color.Yellow;

txtCname.BackColor = System.Drawing.Color.Yellow;

txtUserid.BackColor = System.Drawing.Color.Yellow;

lblError.ForeColor = System.Drawing.Color.Red;

lblError.Text = "Company Name may not be empty. Contract Number may not be empty. Staff ID may not be empty.";

return;

if (string.IsNullOrEmpty(businessName) && string.IsNullOrEmpty(contractID))

txtContractid.BackColor = System.Drawing.Color.Yellow;

txtCname.BackColor = System.Drawing.Color.Yellow;

lblError.ForeColor = System.Drawing.Color.Red;

lblError.Text = "Company Name may not be empty. Contract ID may not be empty.";

return;

if (string.IsNullOrEmpty(staffid) && string.IsNullOrEmpty(zipcode))

txtUserid.BackColor = System.Drawing.Color.Yellow;

txtZipcode.BackColor = System.Drawing.Color.Yellow;

lblError.ForeColor = System.Drawing.Color.Red;

lblError.Text = "Staff ID may not be empty. Zipcode may not be empty.";

return;

if (string.IsNullOrEmpty(job) && string.IsNullOrEmpty(contractdate))

txtDescription.BackColor = System.Drawing.Color.Yellow;

txtContractdate.BackColor = System.Drawing.Color.Yellow;

lblError.ForeColor = System.Drawing.Color.Red;

lblError.Text = "Job description may not be empty. Contract date may not be empty";

return;

if (string.IsNullOrEmpty(businessName))

txtCname.BackColor = System.Drawing.Color.Yellow;

lblError.ForeColor = System.Drawing.Color.Red;

lblError.Text = "Company Name may not be empty.";

return;

if (string.IsNullOrEmpty(contractID))

txtContractid.BackColor = System.Drawing.Color.Yellow;

lblError.ForeColor = System.Drawing.Color.Red;

lblError.Text = "Contact Number may not be empty.";

return;

if (string.IsNullOrEmpty(zipcode))

txtZipcode.BackColor = System.Drawing.Color.Yellow;

lblError.ForeColor = System.Drawing.Color.Red;

lblError.Text = "Zipcode may not be empty";

return;

if (string.IsNullOrEmpty(contractdate))

txtContractdate.BackColor = System.Drawing.Color.Yellow;

lblError.ForeColor = System.Drawing.Color.Red;

lblError.Text = "Contract Date may not be empty.";

return;

### Modules Uses

This module is used to save new contract information for companies that are interested in registering as a client with the firm.

## *View\_Contract.aspx*

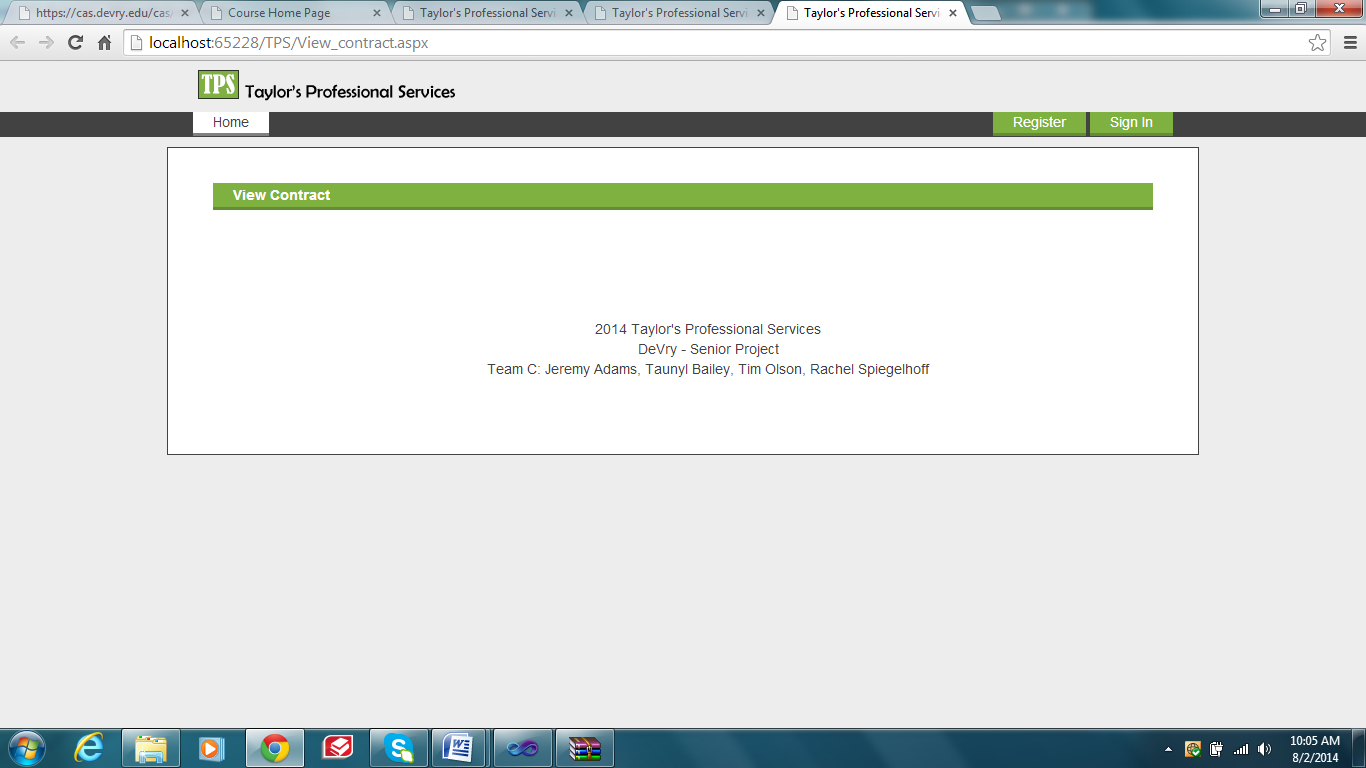
### Processing Description

This method will allow a user (read only) to view all of the contracts and client information in the database.

View Contract button clicked

Displays all contracts with read only access.

### Interface Description



After login, you will select View Contract from the menu and all contracts will show here

After valid login, a user can select View Contracts from the menu, and it will show a list of all contracts in the system.

### Pseudocode

If view contract clicked

Retrieve contract info from database

### Modules Uses

This module calls upon the database to fill all contracts in the database for all clients.

## *Manage\_contracts.aspx*

### Processing Description

Submit button pressed.

Valid information provided in a field.

Contract manager enters Contract ID, Company Name or Staff ID

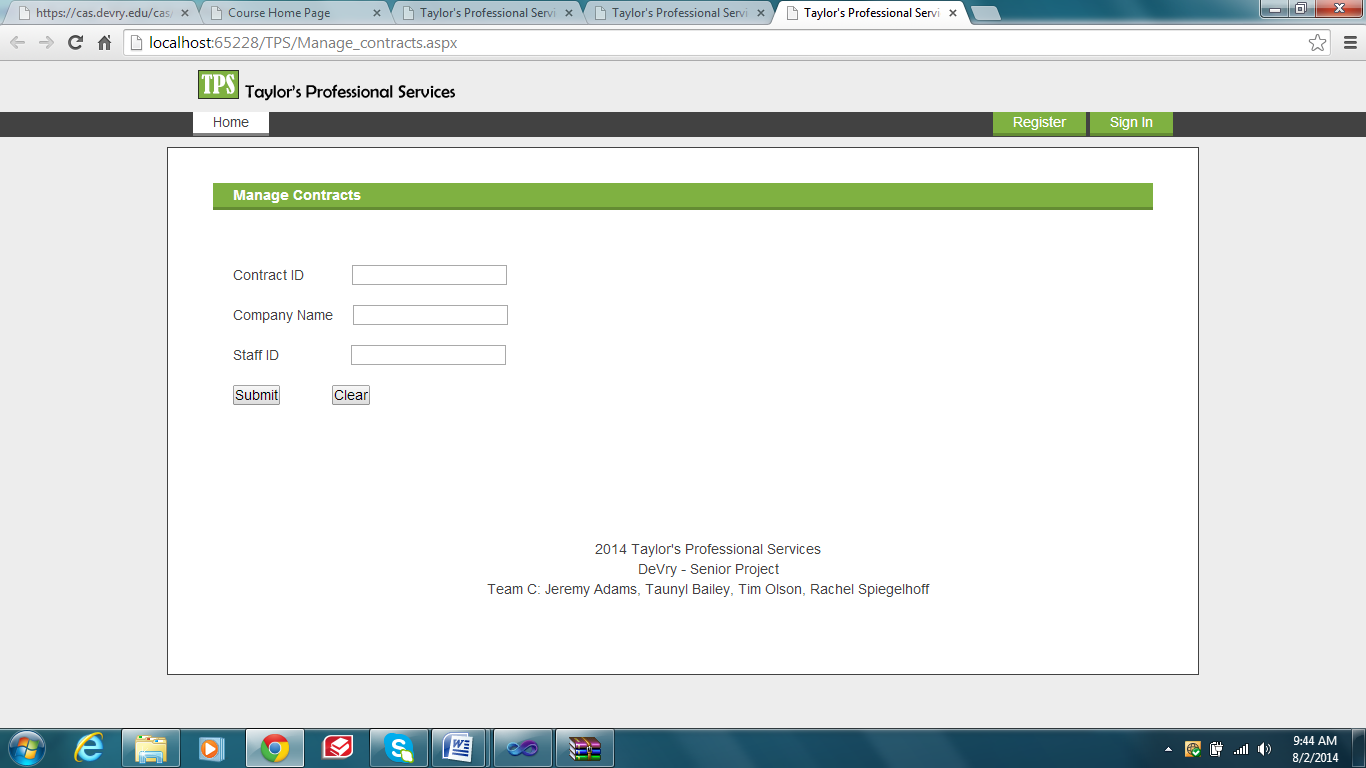
Invalid or No data entered

Information updated to database

Options appear to Edit, or Delete Contract

This method allows an administrator to edit or delete company contract information from the database.

### Interface Description



Selected contract information will display here

Search by Company Name

Search by Staff/User ID

Search by Contract ID

Located on this page is the option to Manage Contracts. This page contains text boxes for the contract id, company name and staff id. Once you select the submit button it will place in the empty area the contract and company information you were searching for. Once the information shows in the grid you will have an option to update, edit or delete a contract.

### Pseudocode

// Get values from the data entry fields in session state

string businessName = txtCname.Text.Trim();

string contractID = txtContractid.Text.Trim();

string staffid = txtUserid.Text.Trim();

// Remove the previous yellow

txtCname.BackColor = System.Drawing.Color.White;

txtContractid.BackColor = System.Drawing.Color.White;

txtUserid.BackColor = System.Drawing.Color.White;

//Add validation codes for CompanyName, Contract ID, Staff ID and Contract Date -- Checks for empty or filled with spaces

if (string.IsNullOrEmpty(businessName) && string.IsNullOrEmpty(contractID) && string.IsNullOrEmpty(staffid))

txtContractid.BackColor = System.Drawing.Color.Yellow;

txtCname.BackColor = System.Drawing.Color.Yellow;

txtUserid.BackColor = System.Drawing.Color.Yellow;

lblError.ForeColor = System.Drawing.Color.Red;

lblError.Text = "Company Name may not be empty. Contract Number may not be empty. Staff ID may not be empty";

return;

if (string.IsNullOrEmpty(businessName))

txtCname.BackColor = System.Drawing.Color.Yellow;

lblError.ForeColor = System.Drawing.Color.Red;

lblError.Text = "Company Name may not be empty.";

return;

if (string.IsNullOrEmpty(contractID))

txtContractid.BackColor = System.Drawing.Color.Yellow;

lblError.ForeColor = System.Drawing.Color.Red;

lblError.Text = "Contract Number may not be empty.";

return;

if (string.IsNullOrEmpty(staffid))

txtUserid.BackColor = System.Drawing.Color.Yellow;

lblError.ForeColor = System.Drawing.Color.Red;

lblError.Text = "Staff ID may not be empty.";

return;

### Modules Uses

This module will search the database for the information entered on the page, then display it in the area below the options. Once the company information is displayed you will be able to edit, update or delete the client from the database.

## *Update\_contract.aspx*

### Processing Description

This method allows a user (with proper credentials) to update contract and company information providing edit and update rights.

Invalid information entered

Contract information with Edit options

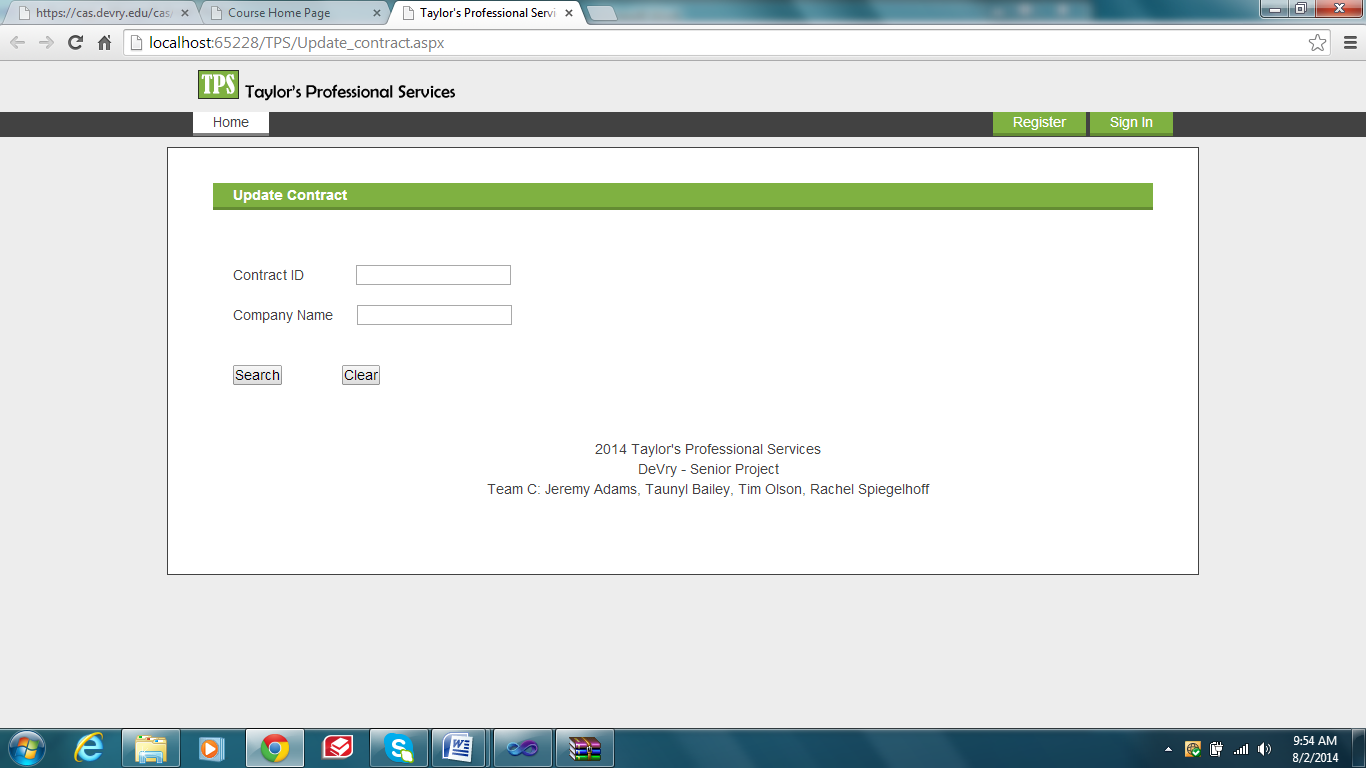
Information saved to database.

Valid information entered.

Submit button is pressed.

Contract manager enters Contract ID or Company Name

### Interface Description



Information will display here.

Search by Contract ID

Search by Company Name

Search by Contract ID

Search by Contract ID

Search by Contract ID

Located on this page is the option to Update Contracts. This page contains text boxes for the contract id, and company name . Once you select the submit button it will place in the empty area the contract and company information you were searching for. Once the information shows in the grid you will have an option to edit and update a contract.

### Pseudocode

// Get values from the data entry fields in session state

string businessName = txtCname.Text.Trim();

string contractID = txtContractid.Text.Trim();

// Remove the previous yellow

txtCname.BackColor = System.Drawing.Color.White;

txtContractid.BackColor = System.Drawing.Color.White;

//Add validation codes for CompanyName, Contract ID, Staff ID and Contract Date -- Checks for empty or filled with spaces

if (string.IsNullOrEmpty(businessName) && string.IsNullOrEmpty(contractID))

txtContractid.BackColor = System.Drawing.Color.Yellow;

txtCname.BackColor = System.Drawing.Color.Yellow;

lblError.ForeColor = System.Drawing.Color.Red;

lblError.Text = "Company Name may not be empty. Contract Number may not be empty.";

return;

if (string.IsNullOrEmpty(businessName))

txtCname.BackColor = System.Drawing.Color.Yellow;

lblError.ForeColor = System.Drawing.Color.Red;

lblError.Text = "Company Name may not be empty.";

return;

if (string.IsNullOrEmpty(contractID))

txtContractid.BackColor = System.Drawing.Color.Yellow;

lblError.ForeColor = System.Drawing.Color.Red;

lblError.Text = "Contract Number may not be empty.";

return;

### Modules Uses

This module will search the database for the information entered on the page, then display it in the area below the options. It will then allow you to edit or update company information and save it to the database.

## *Find\_staff.aspx*

### Processing Description

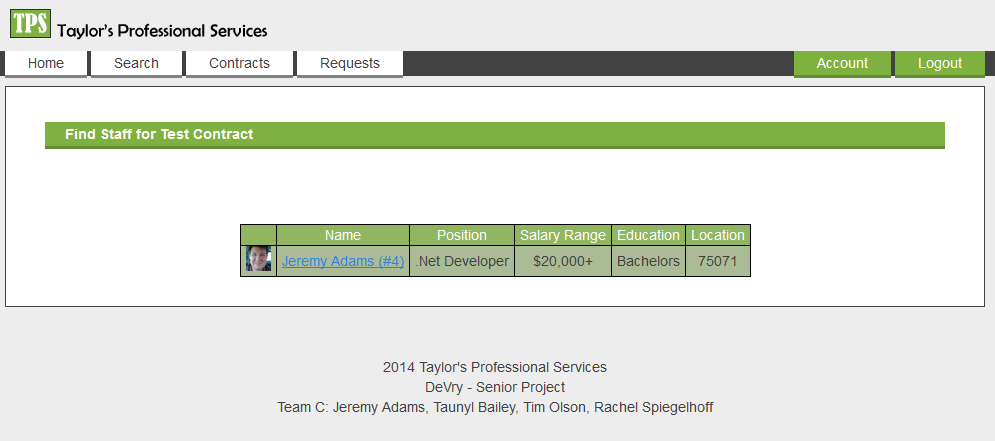
Find Staff that will match the requirements

GET[“id”] != null

Pull Contract requirements from db

Redirect user if Get[“id”] == null

### Interface Description



Redirect user to view staff information where they can make requests

Name of Contract

Staff matching contract requirements

### Pseudocode

On Page Load

if Session != null then

if Session == client || Session == manager then

if GET[‘id’] != null then

try to convert GET[“id”] to int

catch return to Manage\_contracts.aspx

if session type is client then

Select \* From contract where userid = session and contractid = Get[“id”]

else // if manager

Select \* from contract where contractid = get[“id”]

if results are found then

Select user.\*, staff.\* where values match contract requirements

Display results in table

else redirect user

else redirect user

else redirect user

else redirect user

### Modules Uses

GetConnectionString()

return connection string to DB

## *View\_requests.aspx*

### Processing Description

Redirect to View\_Staff.aspx

Staff ID clicked

Redirect to View\_Contract.aspx

Contract ID clicked

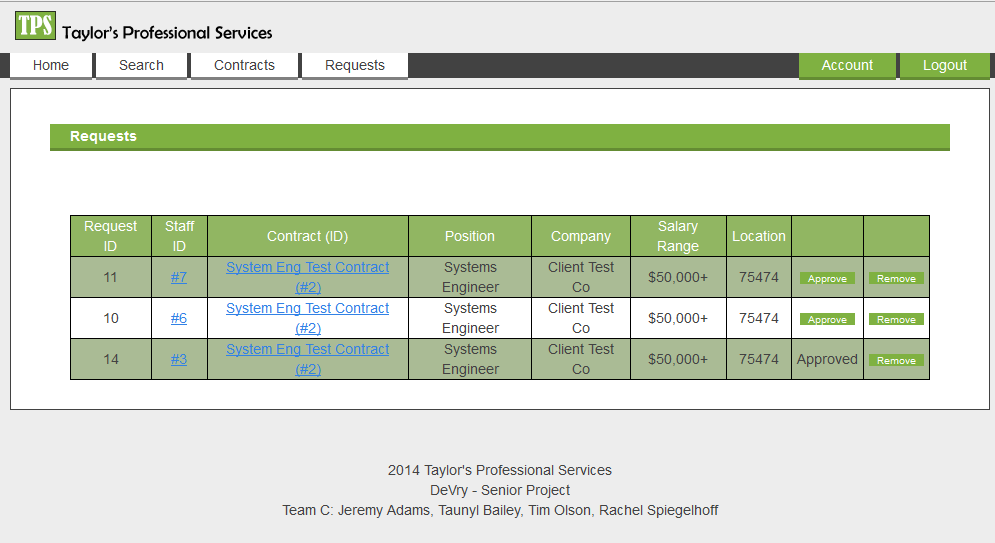
Remove button clicked

Approve button clicked

Contract removed from database

Contract approved

### Interface Description



Accept or Approve a Contract request

Remove request from DB

General Contract Information

Redirect user to view contract information

Redirect user to view staff

### Pseudocode

On Page Load

if session != null and session type != client then

if session == staff then

select \* request that have been approved or accepted

else // manager

select \* requests that have not been denied

set counter = 0

while db.rows exists

if counter = 0

insert new table row with column titles into requesttable

create new table row

create a cell for each db.value and insert into row

insert new row into requesttable

else

redirect to homepage

### Modules Uses

GetConnectionString()

return connection string to DB

Button\_StaffAccept\_Click()

update request set approval = 2 were userid = staffId and contractid = selectedContractId

if update successful then

display success message

delete any request where userid <> Session[userid] and contractid = selectedContractId

Update staff set status = 0 where userid = session[userid]

Update contract set status = 2 where contractid = selectedContractId

reload page

Btn\_Admin\_Approve()

Update request set approval = 1 were userid = selectedStaffId and contractId = SelectedContractId

reload page

Btn\_RemoveRequest()

Delete from request where userid = selectedUserid and contractid = selectedContractId

reload page

**Document Work Log:**

*To assist in assessing the contributions made by the individual team members, the team must complete the table below:*

|  |  |  |
| --- | --- | --- |
| ***Section*** | ***Team Member - Primary*** | ***Team Member - Secondary*** |
| 1. *Scope* |  |  |
| * 1. *System Description* | *T.Olson* | *R.Spiegelhoff* |
| * 1. *Major Software Functions* | *R.Spiegelhoff* | *J. Adams* |
| * 1. *Database Description* | *R.Spiegelhoff* | *J. Adams* |
| * 1. *Database Constraints and Limitations* | *R.Spiegelhoff* |  |
| 1. *Design Description* | *T.Bailey* |  |
| * 1. *Data Description* | *T.Bailey* | *J. Adams* |
| *Schema* | *T.Bailey* | *T. Olson* |
| *Table and Relationships – Data Dictionary* | *T.Bailey* | *J. Adams* |
| * 1. *Data Flow* | *T.Bailey* |  |
| * 1. *Program Architecture* | *T.Bailey* |  |
| * 1. *Component Interfaces* | *T.Bailey* |  |
| 1. *Detailed Design Section -* ***One section (3.1 – 3.x) for each individual component or module*** | | |
| *Detailed Processing Description* | *R.Spiegelhoff* | *T.Bailey* |
| *Interface Description – with UI* | *T. Olson* | *J. Adams* |
| *Pseudo code, Flowcharts or Storyboards* | *J. Adams* | *T.Bailey* |
| *Module Uses* | *J. Adams* | *T. Olson* |