2015

Database for a Spanish to English Services Company

Vlad Predovic

FALL 2015

Database serving a Spanish to English services company

- Employees: EID, Name, PW, phone number, admin(binary)
- Clients: CID, Client Name, password, Email, phone number, address
- Work: WID, Date requested, Client Name, Date Started, Completion Date, Hrs. Worked
- Job: Job type, Date started, DID
- Service: Service type, Location
- Documentation(files): DID, Description, Media Type, CID, Date received

Introduction:

The database project will support a company that provides services for bridging the gap between English and Spanish speakers. This could be something as simple as translating a document or personal like communicating between a sick patient and their doctor and even developing learning material for non-English speakers. The company provides both jobs and services to its clients, and needs a way to keep track of this. Additionally, long-term clients will have accounts so they can automatically request additional items. The company needs a way to keep track of the many documents being worked on, be able to add and remove users, and keep track of hours worked on a job for all the employees.

Detailed Application Requirements:

The first implementation is a way for employees to see the work being done in table form. This will require some form of login/session-protected page with a table drawing joined values from the database. The table will portray jobs by creation date and have details like who is the 'lead' working the job, links to a file attached if applicable, and the status of the job/service to be completed. Additionally the owners of the company want to be able to adjust values themselves such as the time worked on the job, changing current parameters, and adding additional jobs. These user-oriented applications can be done through the implementation of form queries in php.

Another requirement that was decided upon with my sponsors was a way for clients to view the jobs they have requested, those they have completed, and be able to request more work. The issue here is that the company does not want to receive emails from regular clients each time a new job or service is needed. Regulars should have a way to submit work they want done and view their current history. Accounts for each long-term client will be added requiring that they be added to the database as a 'client' object. Once logged on a load query and some php code will bring up a table showing the client their history with a form option to add another request.

The company wants to be able to upload and keep track of information. This company will be dealing with documents a lot of the time due to the translating services that are provided. These need to be archived and connected to the rest of the database. The user will have a way to upload a file in relation to each job. Once this is complete, the document will automatically be related to a client, a work date, and a 'lead' employee through the implementation of the relational database. However, one of the limitations is the storage capacity of the servers. A work-around in consideration is using an existing system such as the Google Drive or Microsoft's One Drive to upload files and then providing the URL link to these to be saved into the database. To users a file link would

be displayed for the current job that would lead them to where the file that is being worked on is located.

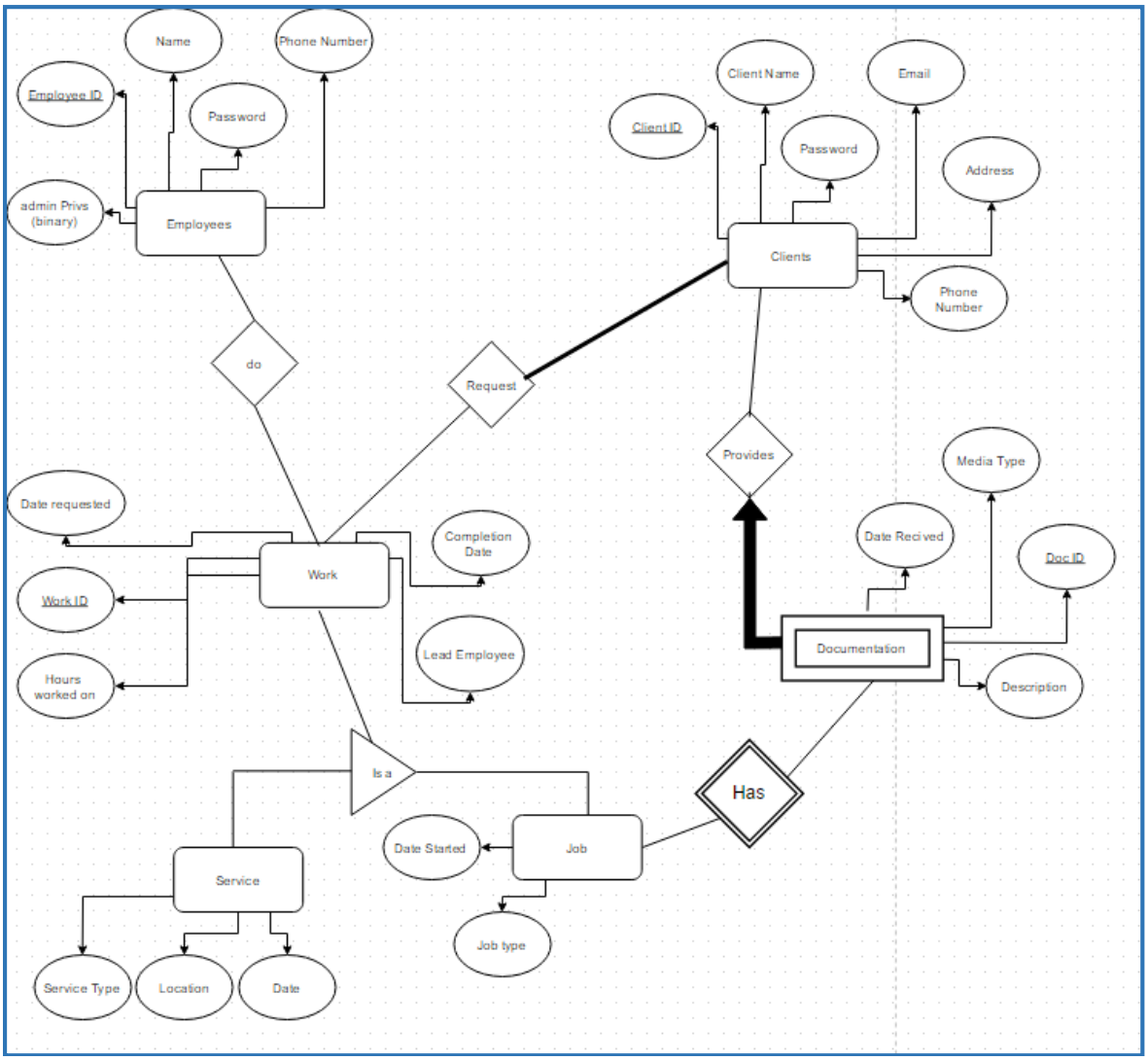
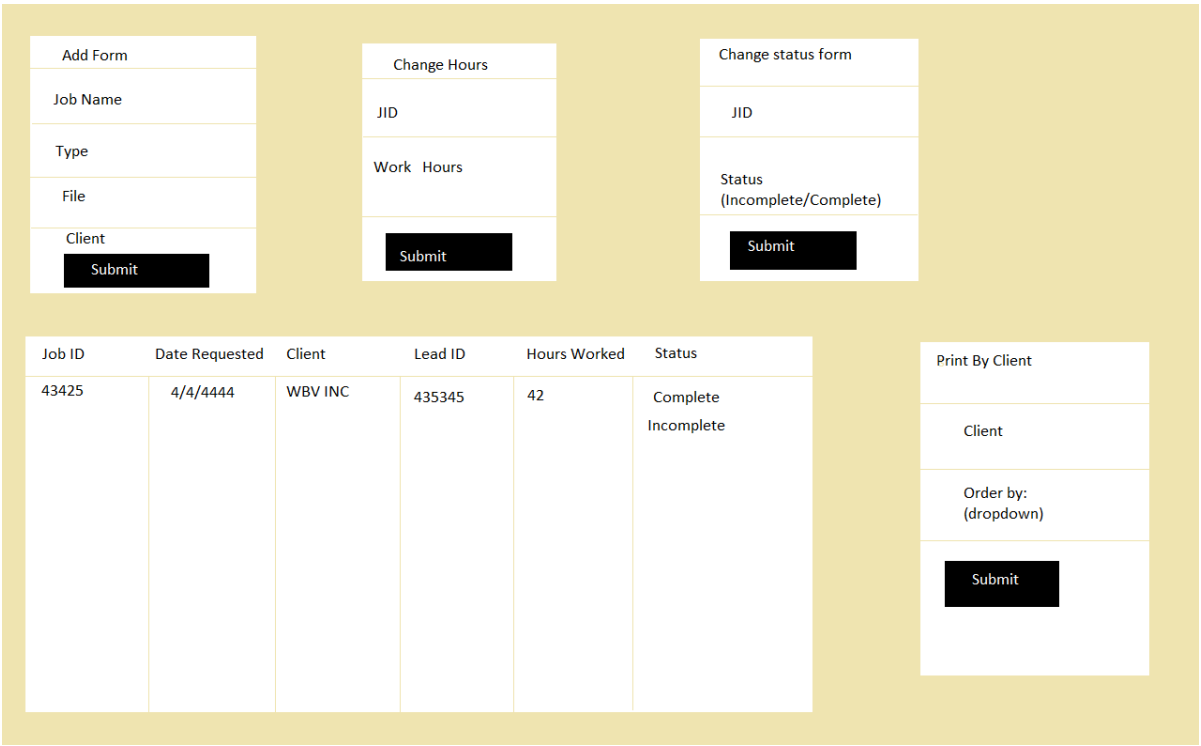


Table to Show Jobs in Work:

Work ID	Date Started	Lead	Client	File	Work Status
3235	5/4/2212	Vlad Pred	BBB LC	www.zzz.com	Incomplete

These are the couple of the forms Administrators and clients will be able to use to enter their information. On the left you can add a job along with a start date and a file link. This can be used by clients to enter jobs on their page. On the right an employee can enter values to alter their job info. The client has also requested the ability to query all requests/jobs by a certain employer and that these appear on a separate printable page.

Below is a rough sketch of what the Administration page might look like for an employee:



Tables, Entities, and Relations:

Employees (*EID: Integer*, Name: char (20), password: char (20), PhoneNumber: char (10), adminPrivs: Boolean)
Candidate Keys: EID, PhoneNumber
Work_On (*EID: integer*, *WID: integer*, HrsWorked: integer)
Candidate Keys: EID, WID
Foreign Keys: EID, WID
Work (*WID: Integer*, DateRequested: Date, CompletionDate: Date, LeadID: Integer (10), WorkType:
Char (6), CID: Integer)
Candidate Keys: WID
Foreign Keys: CID
Service (*WID: Integer*, ServiceType: Char (10), ServiceLocation: Char (50), ServiceDate: Date)
Candidate Keys: WID
Job (*WID: Integer*, DateStarted: Date, JobType: Char (10), DocID: Integer)
Candidate Keys: WID
Foreign Keys: DocID
Documentation (*DocID: Integer*, DateRecieved: Date, MediaType: Char (20), Description: Char 50))
Candidate Keys: DocID, Description
Clients (*CID: Integer*, ClientName: Char (25), Cpassword: Char (20), Email: Char (15), Address: Char (50), Phone: Char (10))
Candidate Keys: CID, ClientName, Email

1. CREATE TABLE EMPLOYEES (EID INTEGER, EmpName CHAR(20), Password CHAR(20), PhoneNumber CHAR(10), adminPrivs TINYINT, PRIMARY KEY(EID))

2. CREATE TABLE Clients (CID INTEGER, ClientName CHAR(25), Cpassword CHAR(20), Email CHAR(15), Address CHAR(50), Phone CHAR(10), PRIMARY KEY (CID))

3. CREATE TABLE Work (WID INTEGER, DateRequested DATE, CompletionDate DATE, LeadEID INTEGER (10), WorkType CHAR(6), CID INTEGER, PRIMARY KEY (WID), FOREIGN KEY (CID) REFERENCES Clients)

4. CREATE TABLE Work_On (EID INTEGER, WID INTEGER, HrsWorked INTEGER, PRIMARY KEY (EID, WID), FOREIGN KEY(EID) REFERENCES Employees, FOREIGN KEY(WID) REFERENCES Work)

5. CREATE TABLE Service (WID INTEGER, ServiceType CHAR(10), ServiceLocation CHAR(50), ServiceDate DATE, PRIMARY KEY (WID), FOREIGN KEY (WID) REFERENCES Work)

6. CREATE TABLE Documentation (DocID INTEGER, DateRecieved DATE, MediaType CHAR(20), Description CHAR(50), PRIMARY KEY (DocID))

7. CREATE TABLE Job (WID INTEGER, JobType CHAR(10), DateStarted DATE, PRIMARY KEY (WID), DocID INTEGER, FOREIGN KEY (WID) REFERENCES Work, FOREIGN KEY DocID REFERENCES Documentation)

Relational Algebra for user-driven questions

What service dates are currently scheduled by Client 'PtMedical'?

$\pi_{DateRequested}((\sigma_{ClientName}=\text{"PtMedical"}Client) \bowtie Service)$ ~This is valid because service is a 'Work'

Provide the ID of documentation available for jobs started in the last 30 days?

$\pi DID((\sigma DateStarted \leq \text{"10/08/2015"}Jobs) \bowtie Documentation)$

What employees are currently leading a project?

$\pi_{EmpName}((\sigma_{LeadEID}\ Work) \bowtie Employees)$

A listing of the amount of hours Sally Morris worked on jobs requested by PTMedical?

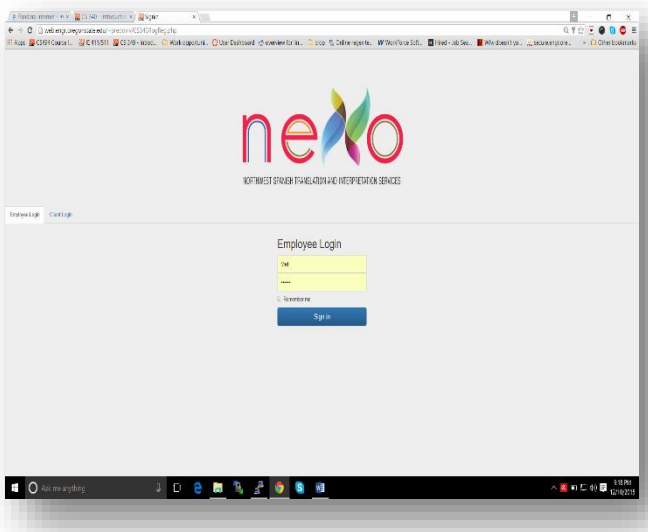
$p(R1, \pi_{WID}((\sigma_{ClientName}=\text{"PtMedical"}Client) \bowtie Work))$

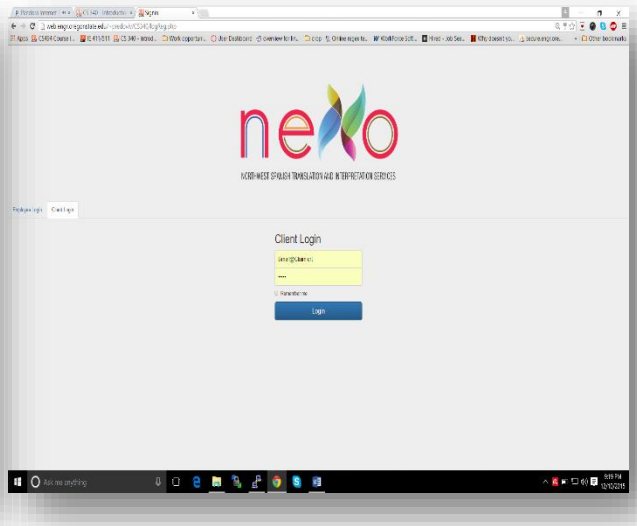
$p(R2, R1 \bowtie Works\_On \bowtie (\sigma_{EmpName}=\text{"Sally Morris"}Employees))$

$\pi_{HrsWorked}R2$

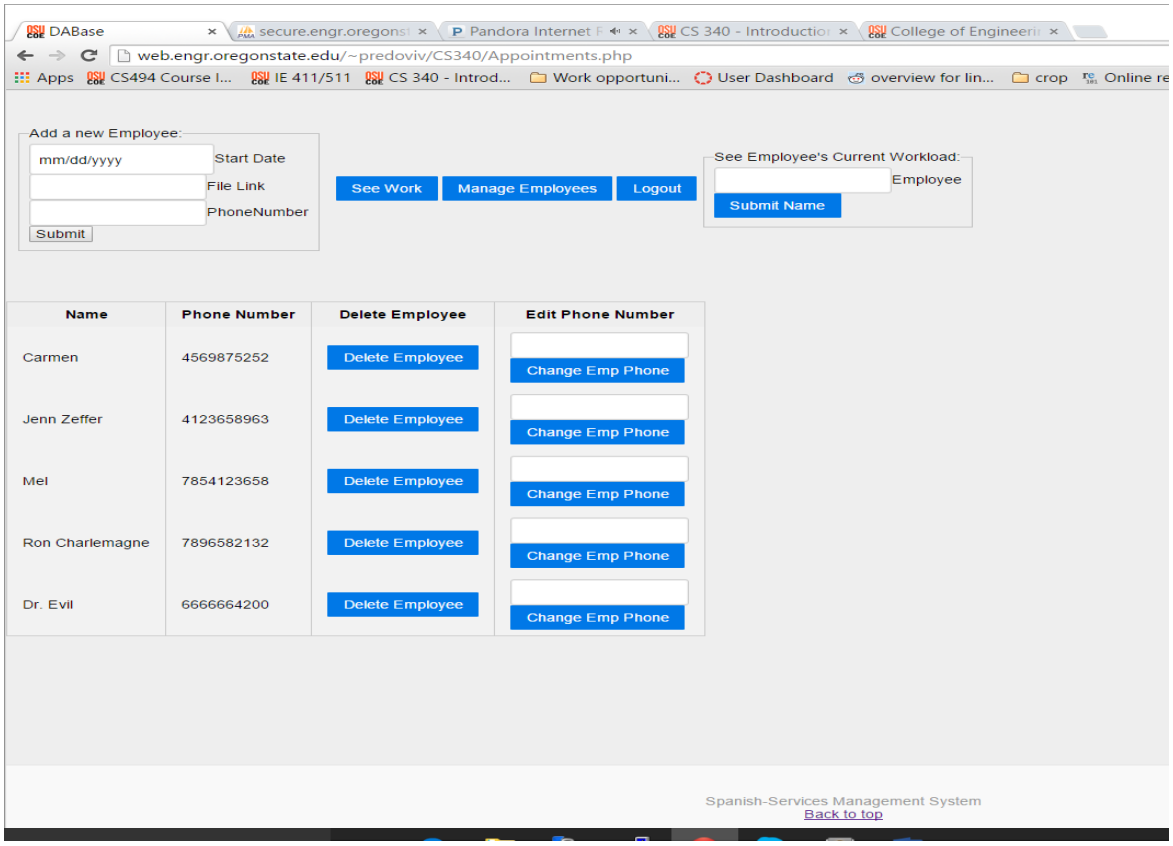
Table and Data Manipulation

For this part of the project I created a two-section database. The client specified the ability to login as a client as well as an employee. Below I aligned features from logging in as an employee and the difference with logging in as a client:





The page featured below shows the basic layout of the employee login page.
You can do things such as add an employee, see the work completed, and remove current employees.
The admin employee can also see any of the current employee's workloads and edit an employee's phone number,



Code used for developing the page above:

Delete Query to remove employees:

```
84
85
86  if(isset($_POST["delEmp"]))
87  {//Still need to make sure this works.
88    $delID = $_POST["delEmp"];
89    $DeleteEMP = $finalDB ->query("DELETE * FROM EMPLOYEES WHERE CID = '$delID'");
90  }
```

HTML code to delete the Employee exists within a function to create a button for each employee and to alter the phone number of the employee.

```
}

function displayEmps($showEmps)
{
   echo "<div id = 'box-formd'><table id='data' class='pure-table' border='1'><tr><th>  Name  </th>
   <th> Phone Number </th><th> Delete Employee </th><th> Edit Phone number_format(number) </th></tr>";
    // output data of each row
   if ($showEmps->num_rows > 0) {
     while($row = $showEmps->fetch_assoc())
     {
       echo "<tr class='pure-table-odd'>
            <td id='target' class='normals' >".$row["Name"]."</td>
            <td >".$row["PhoneNumber"]."</td>
           <td>
                <form class='pure-form' action='Appointments.php' method='post'>
                <button type='submit' class='pure-button pure-button-primary' name='delEmp'
                value=".$row["EID"].">Delete Employee</button>
                </form>
          </td>
           <td>
                <form class='pure-form' action='Appointments.php' method='post'>
                <input size='16' type='text' name='newNum'><br>
                <button type='submit' class='pure-button pure-button-primary' name='updateEmpPhoneNumber'
                value=".$row["EID"].">Change Emp Phone</button>
                </form>
          </td>
      </tr>";
     }
     echo "</table></div>";
     } else {
     echo "0 results";
     }
}
```

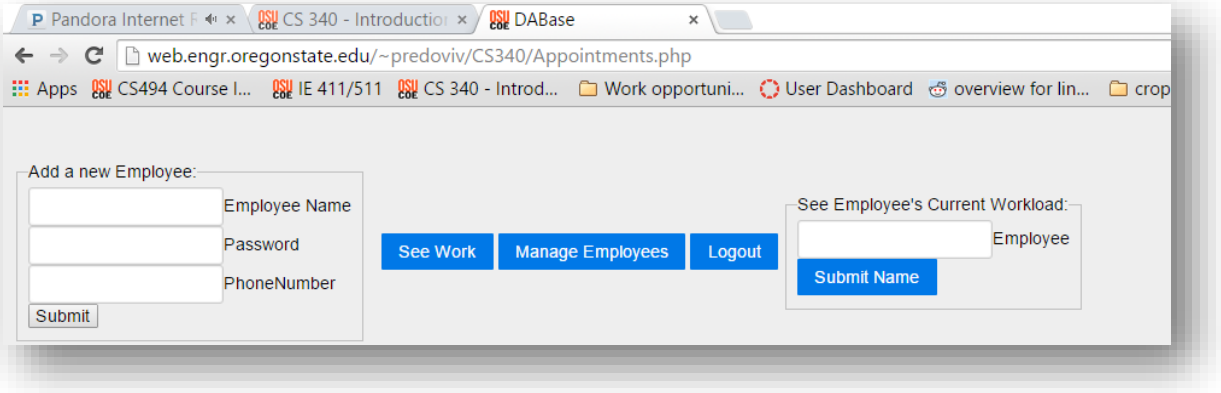
Code to add a new Employee:

```
43  if(isset($_POST["addEmp"]))
44  {
45
46    $addDoc = $finalDB->prepare("INSERT INTO EMPLOYEES (Name, Password, PhoneNumber, adminPrivs) VALUES (?, ?, ?, ?)");
47    $addDoc->bind_param("ssi", $empName, $empPass, $empPhNum, $empAdmin);
48    $empName=$_POST["empName"];
49    $empPhNum=$_POST["empPhoneNumber"];
50    $empPass=$_POST["empPass"];
51    $empAdmin=0;
52  //https://drive.google.com/file/d/0Byfufr0utaZsVXFWQ0VSODAzcGs/view?usp=sharing
53    $addDoc->execute();
54    if ($addDoc == 0)
55      die(mysqli_error($finalDB));
56    else
57      echo "Succesfully added new employee";
58    $addDoc->close();
59  }
60
61
62  /* create a prepared statement */
```

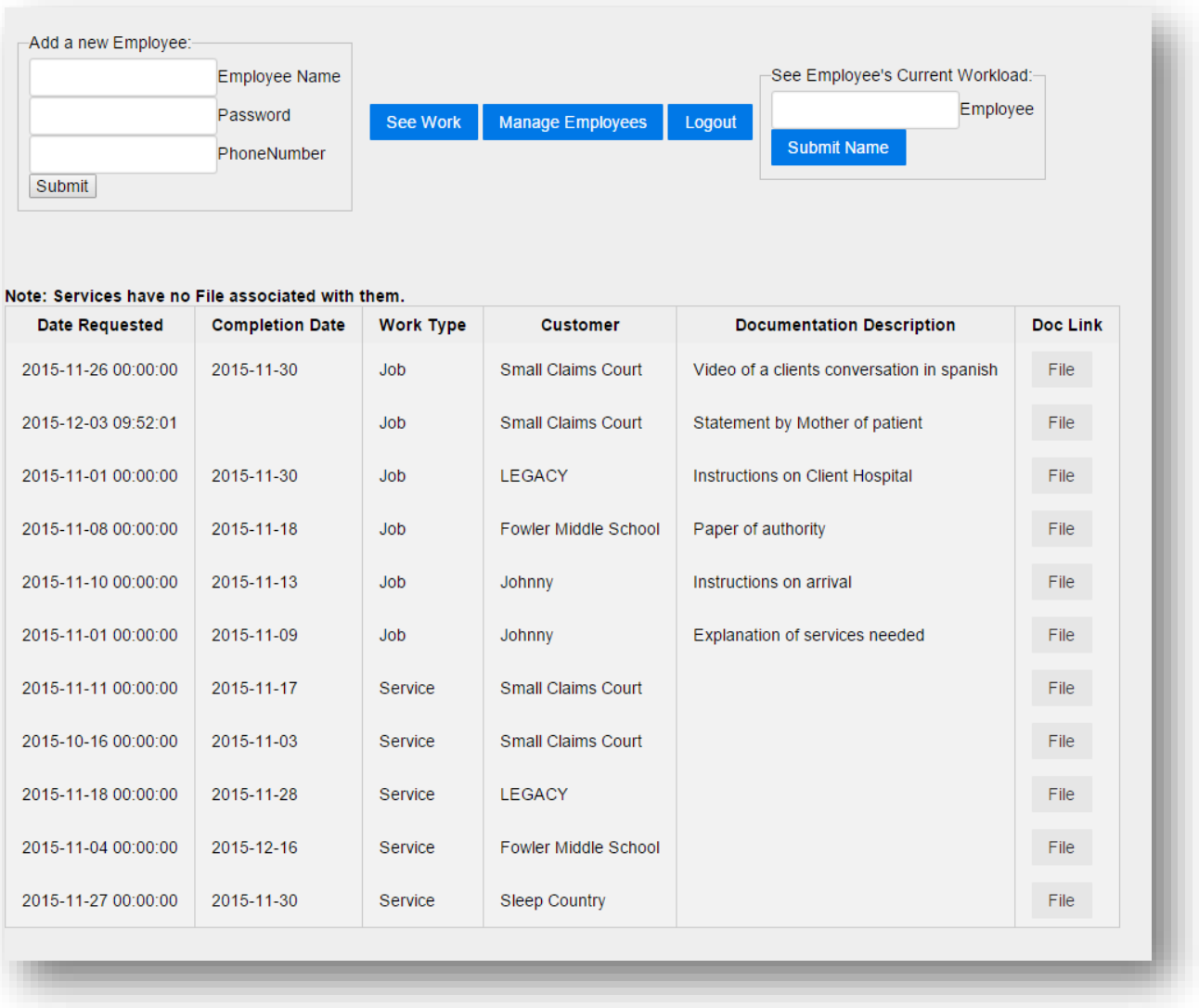
Code to Update an Employee Phone Number or Delete given employee:

```
if(isset($_POST["delEmp"]))
{//Still need to make sure this works.
  $delID = $_POST["delEmp"];
  $DeleteEMP = $finalDB ->query("DELETE * FROM EMPLOYEES WHERE EID = '$delID'");
}

if(isset($_POST["updateEmpPhoneNumber"]))
{
$empID = $_POST["delEmp"];
$newNum = $_POST["newPhoneNumber"];
"UPDATE MyGuests SET lastname='Doe' WHERE id=2";
$DeleteEMP = $finalDB ->query("UPDATE EMPLOYEES SET PhoneNumber='$newNum' WHERE EID = '$delID'");

}
```

The figure outlined below below represents the command menu and possible options from the employee admistrators login screen. As you can see, for now there are not many options. One of the critical issues regarding my design is that my clients are not tech-savvy, in fact they are very bad with computers.



As a result, simplyfing the system as much as possible froma users perspective was one of the highest priorities. Pressing the See Work button allows one to all work items that have been processed. It also allows access to all associated documentation, shows the related customer, and explains what type of job it was. This is one of the most important views as it presents a story of the workflow.

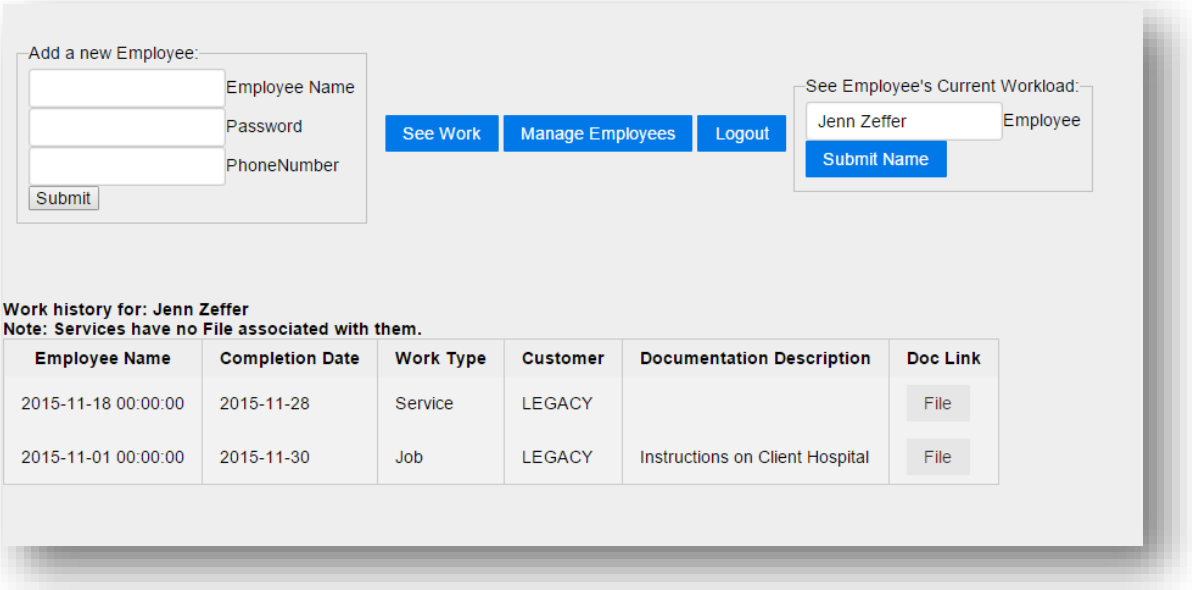


Date Requested	Completion Date	Work Type	Customer	Documentation Description	Doc Link
2015-11-26 00:00:00	2015-11-30	Job	Small Claims Court	Video of a clients conversation in spanish	File
2015-12-03 09:52:01		Job	Small Claims Court	Statement by Mother of patient	File
2015-11-01 00:00:00	2015-11-30	Job	LEGACY	Instructions on Client Hospital	File
2015-11-08 00:00:00	2015-11-18	Job	Fowler Middle School	Paper of authority	File
2015-11-10 00:00:00	2015-11-13	Job	Johnny	Instructions on arrival	File
2015-11-01 00:00:00	2015-11-09	Job	Johnny	Explanation of services needed	File
2015-11-11 00:00:00	2015-11-17	Service	Small Claims Court		File
2015-10-16 00:00:00	2015-11-03	Service	Small Claims Court		File
2015-11-18 00:00:00	2015-11-28	Service	LEGACY		File
2015-11-04 00:00:00	2015-12-16	Service	Fowler Middle School		File
2015-11-27 00:00:00	2015-11-30	Service	Sleep Country		File

Accomplishing this type of view was fairly tricky due to to the complex nature of the database. Below is the query used to access it. Of note is that this query is extremely inneficient. An improvement currently set to be rolled out is to alter this query to only extract the exact data we need, as opposed to selecting everything as it is now doing.

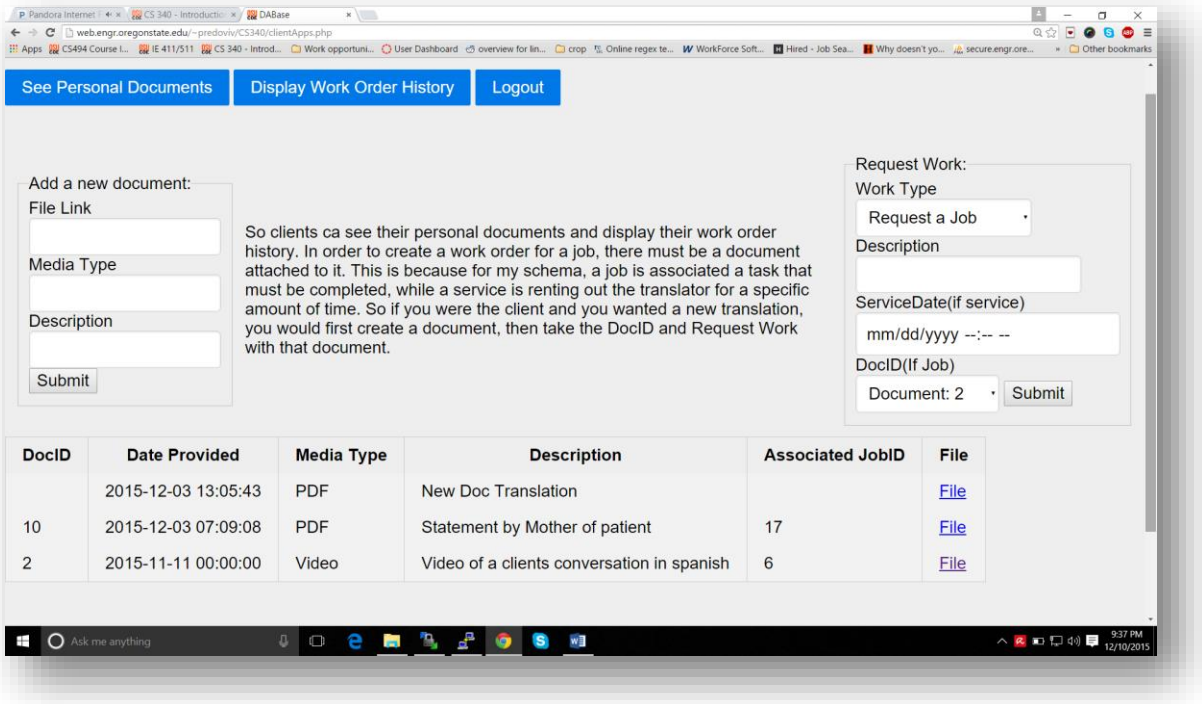
```
//Query to fill table with values only from this client
$employee = $_SESSION['empID'];
$showProjects = $finalDB ->query("SELECT * FROM Work
                                  INNER JOIN Clients on Work.CID = Clients.CID
                                  LEFT JOIN Job on Work.WID = Job.WID
                                  INNER JOIN Documentation on Job.DocID = Documentation.DocID
                                  UNION
                                  SELECT * FROM Work
                                  INNER JOIN Clients on Work.CID = Clients.CID
                                  LEFT JOIN Job on Work.WID = Job.WID
                                  LEFT JOIN Documentation on Job.DocID = Documentation.DocID");
```

Finally, typing in a users name into the last field, will show the administrator a work of log for a given employee as shown below for Jenn Zeffer.

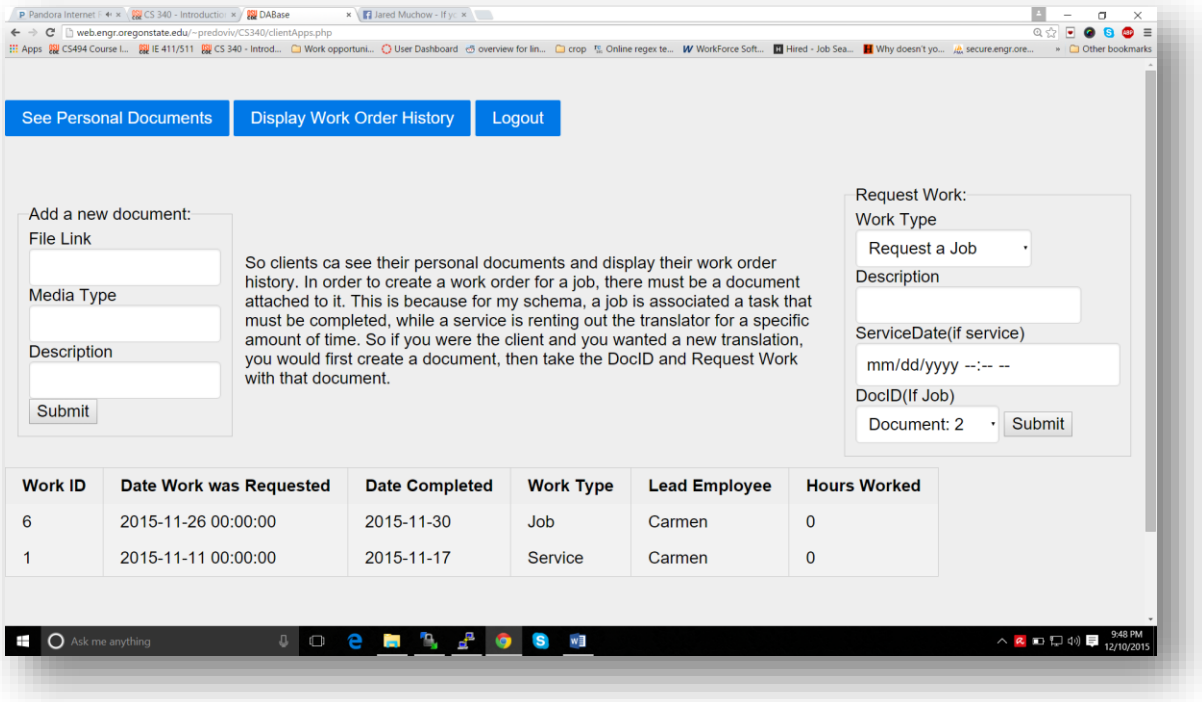


However, this is only half the story as one of the requirements for this project was that the client be also able to access a site where they could monitor the current work being done and request new translations or interpretations from the company. Below is a view of the main page when logging in as a client.

In the center of the page is a quick explanation of the page's uses. The example below highlights what occurs if the button 'See Personal Documents' is pressed. It allows a client company to see a list of documents they have provided to be translated. If the Document is not connected to a requested job, the ID will not appear.
Work can be requested using the form on the right hand side. Adding documents can be done with the form on the left hand side of the screen.



The button ‘Display Work Order History’ provides the client with critical information. It shows when work was requested, what type of work it was, who is the lead employee on the case, the amount of hours put towards the job, and the date completed if the job has been completed.The forms and dispalys work well in unision, providing the client with control and visibility over the completion of work that they require.



A note on security:

My application does not allow one to view any of the pages unless you are logged in correctly. This is for two reasons the first of which is to maintain the confidentiality of employees and clients. The second reason is that the system uses the login info of the user to cater the website as a personal view, only querying for data regarding that user.

Below I have highlighted some sample users for logging in and testing the website.

Accesing my Project:

For Employee Perspective:
Name = Mel
Password = Gibson

For Client Perspective:
Email = Small@Claim.crt
Password = SMALL

Link: http://web.engr.oregonstate.edu/~predoviv/CS340/logReg.php

Explanation of Scripts:
My files are organized as followed:

- *Appointments.php:* File consisting of all forms and scripts used to modify, delete, add, or display data relevant to an employee administrator.
- *clientApps.php:* File consisting of all forms and scripts used to modify, delete, add, or display data relevant to a client of the company.
- *Logout.php:* A page which serves solely to destroy all session data and redirects the current view to logReg.php
- *logReg.php:* Login page in which a user can access the system either as a company administrator or as a client.
-