Creating an Application Management System in HTML, SQL, and PHP

Senior Project

In partial fulfillment of the requirements for
The Esther G. Maynor Honors College
University of North Carolina at Pembroke

By

Jonathan Culbreth
Department of Mathematics and Computer Science
April 15th, 2021

Jonathan Culbreth	5/3/2021
Jonathan Culbreth Honors College Scholar	Date
Xin Zhang	5/3/2021
Cynthia Xin Zhang, Ph.D. Faculty Mentor	Date
Joshua Kalin Busman, Ph.D. Senior Project Coordinator	Date

Table of Contents

Acknowledgements

The assistance and guidance provided by both Dr. Busman and Dr. Zhang was greatly appreciated. I would also like to thank my father, John Culbreth, for his highly valued advice and experience regarding my approach and design of the project.

Abstract

Many applications today are browser-based, and require only internet access for utilization. For this project, I developed a web-based application in HTML, SQL, and PHP. The goal of the project was to develop a system for creating new job listings and applying to said listings in one environment, with additional emphasis placed on software security measures with minimal impact. The program and its various elements were programmed over the course of three weeks throughout March. Ultimately, a few elements were eliminated from the project due to time constraints, though a functioning system with a variety of features was still produced as a result, and I learned a lot about both managing and creating databases as well as programming in PHP. Furthermore, with some modifications and additions, the system could be extended into a fully functioning application management system. This paper provides background information on the techniques and software used and proceeds to detail the development process in chronological order, from start to finish, including the issues and mitigations that ensued, as well as the changes that would be required in order to find use with this system beyond an experimental learning process.

Creating an Application Management System in HTML, SQL and PHP

1. Introduction

Many modern software applications are hosted online in one form or another. The cloud computing technology provides an excellent opportunity for productivity, as it allows users to access content and documents across almost any device or operating system from any location with internet access. With this in mind, I decided for my project that I would build a piece of online software of some sort that could be easily adaptable for real-world use, with specific emphasis on cybersecurity functionality. For this, I settled on creating an online job application management system that would offer users the option to create job listings and apply to those, and carried out the process of programming the system over the course of a month. I began with my own local virtual server by configuring an install of the XAMPP open-source web server management toolkit and programmed the rest of the project in HTML, CSS, PHP, and SQL. This tested system is scalable and can be migrated to any virtual machine on a cloud system.

2. Background Information

The software system is, in its entirety, stored locally on my personal Windows 10 Professional PC. Initially, I had planned to host the website online after the project was sufficiently developed, but decided against it. This was due to the fact that, during development, I was constantly adjusting and testing elements of the program, and having it stored and hosted locally made such work easier. The XAMPP toolkit provided simple

and useful configuration as well for the SQL databases through the PHP MyAdmin management software.

Generally, the system consists of two parts – the SQL database and the various files for the webpages themselves. The SQL database and server files are stored in separate locations under XAMPP's file system, and are connected when required in the website's files. The program itself was written in three programming languages. The main is HTML, or "hyper-text markup language" – this is the basis of the entire system. In addition to HTML, parts are programmed in "CSS," or "cascading style sheets" – this was used to determine the visual design and layout of the GUI, or graphical user interface. The third language used is PHP, or "hypertext preprocessor," which was used to submit and retrieve data from the SQL databases. SQL was used for its database functionality, though some SQL code used was applied in the PHP MyAdmin software in order to modify the database by performing some tasks, such as dropping or adding tables while others were in the form of embedded SQL throughtout my PHP programs.

3. Preparation

Before beginning the programming phases of the project, I completed some initial preparations regarding the functionality and interface design. First, I decided early on that I would like to stick with a darker interface color palette with specific highlights. Initially, the highlight colors were a darker shade of blue, though during development I would change this to gold, as I felt it would stand out visually more, look better, and would feature the general color theme of UNCP. Additionally, beyond personal preference, the darker color scheme itself poses two key advantages: first, it is less likely to cause eyestrain with use, and second, on many modern displays, darker

colors require less power usage, which can save battery on mobile devices such as laptops and cellphones.

Next, I planned out some of the pages the system would require for functionality. This included the login page, dashboard, profile page, and company page.

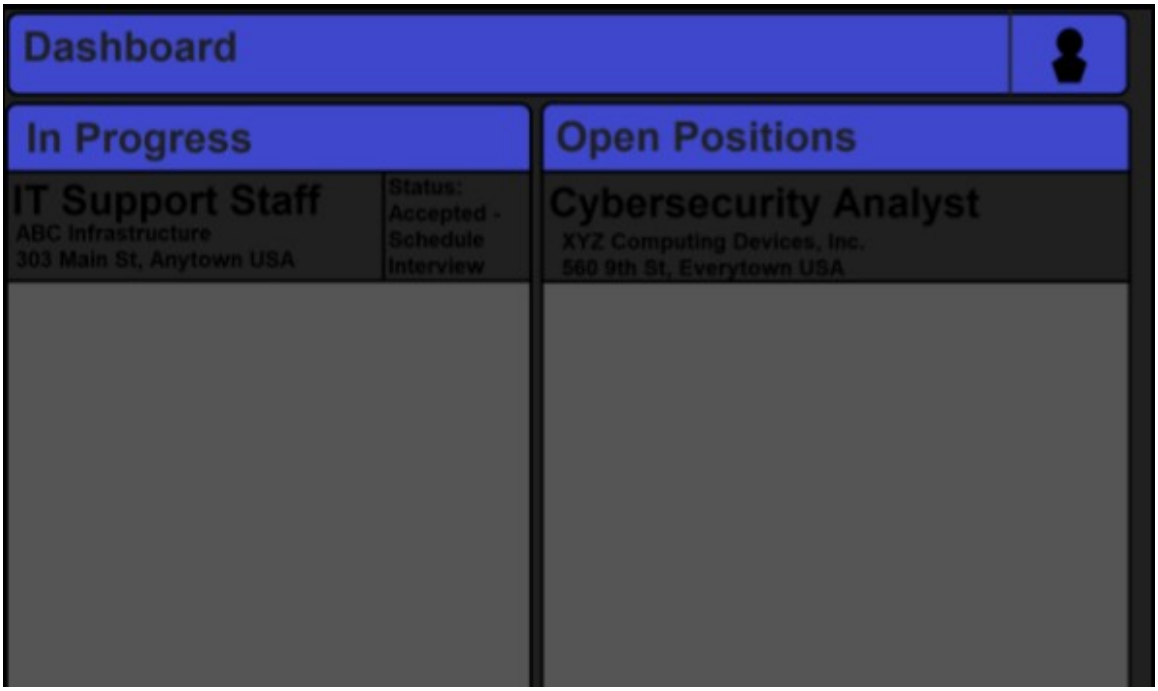

Figure 1: Prototype dashboard design

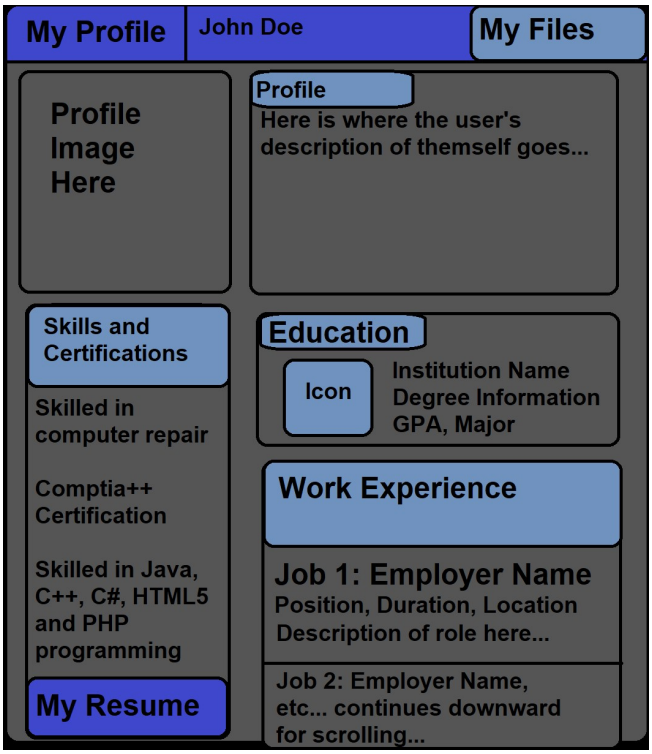

Fig. 2: Prototype profile design

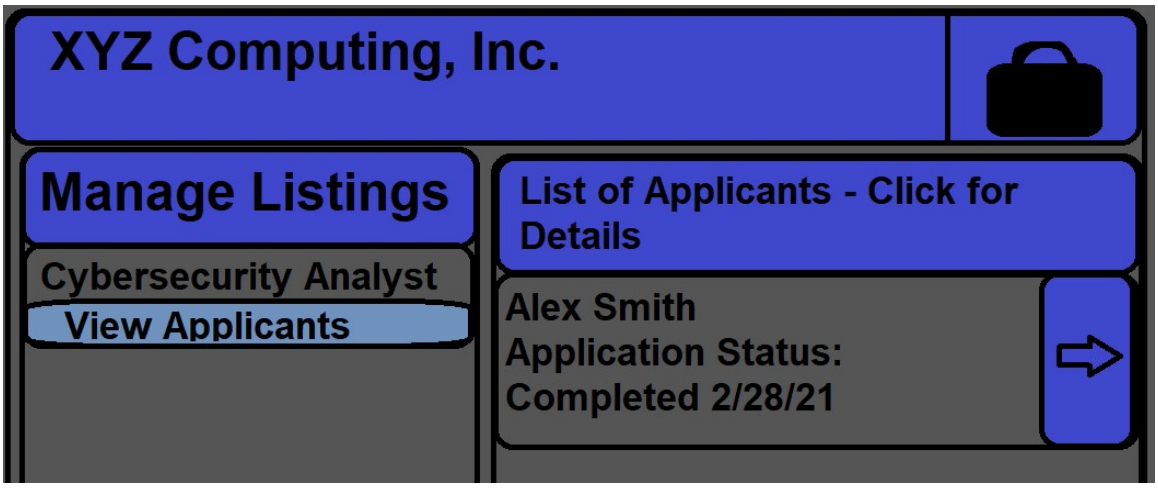

Fig. 3: Prototype company page design

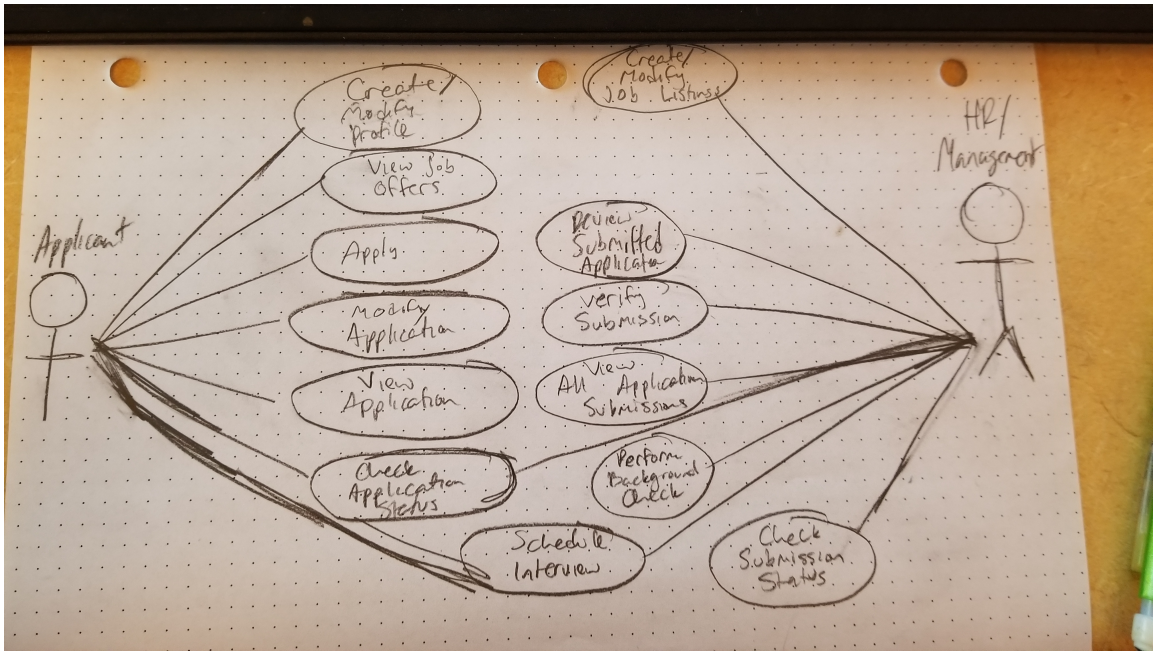

Fig. 4: Use Case Diagram

For my use case diagram, I used representations of two groups - the applicants and human resources/management for the employers.Applicants have access to the following:create/modify user profile, view job offers, submit applications, modify applications, view applications, check application status, and scheduling an interview. HR/Management would have access to creating and modifying job listings, viewing the submitted applications for a specified listing, verifying the eligibility and legitimacy of applications, viewing all applications, performing background checks, viewing the status of all current submitted applications, and scheduling an interview with a prospective employee.

4. First Week

For the first week of development, I began by building the initial login and registration functions and settled on the basic design for the interface as far as layout and color scheme. I decided early on that I would like to stick with a darker interface color palette with gold highlights. Beyond personal preference, the darker color scheme itself poses two key advantages: first, it is less likely to cause eyestrain with use, and second, on many modern displays, darker colors require less power usage, which can save battery on mobile devices such as laptops and cellphones.

First, I started by creating the database itself used for the project as the Database Adaministrator (DBA). To do so, I created it in PHP MyAdmin with the name "ams" and created the first table of that database for users, and labelled it "accounts." During the first week of development, the accounts database contained the username, password, email address, and an ID number for that user. The ID number was automatically assigned and incremented by the database upon registration, and the username, password, and email address were created by the user during registration. Furthermore, the password is automatically hashed with SHA-1 encryption upon creation, for security. Additionally, the system performs a check to ensure that the username is not already in use, and during registration, all SQL statements are prepared to prevent an SQL injection, which is a type of vulnerability in which by entering certain characters an unauthorized party could gain access to the database.

Upon logging in, a session was created and assigned a name based on the user's username. From there, each user had three options – they could view the home page, view their profile page, or log out. The log out option was very basic, and would

simply close the user's session and redirect them to the login page. The home and profile pages, however, were very basic prototypes at this point, and displayed minimal placeholder information. The home page simply displayed a welcome back message, which displayed the session name. The profile page displayed their username and email.

During this stage, I also experimented with the programming required for email verification, but decided against implementing that code as doing so would require I also configure and create a webmail server. With additional time, that feature could have been implemented, but was not feasible as the configuration, implementation, and testing would have required significant additional time.

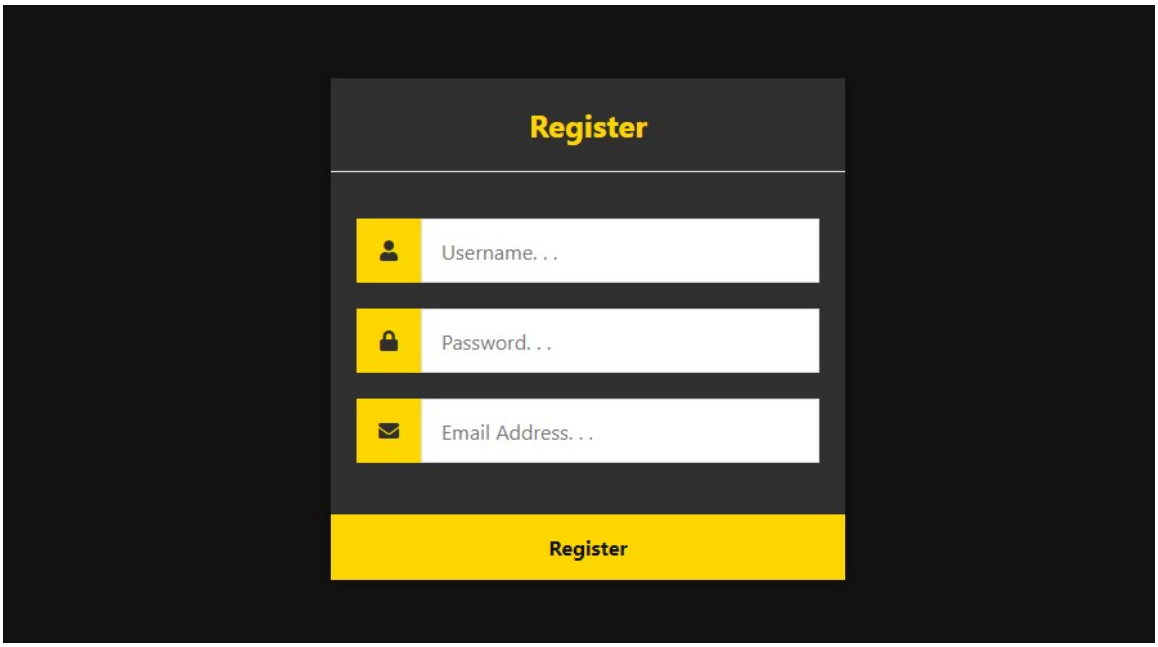

Figure 1: The registration page during the first week of development

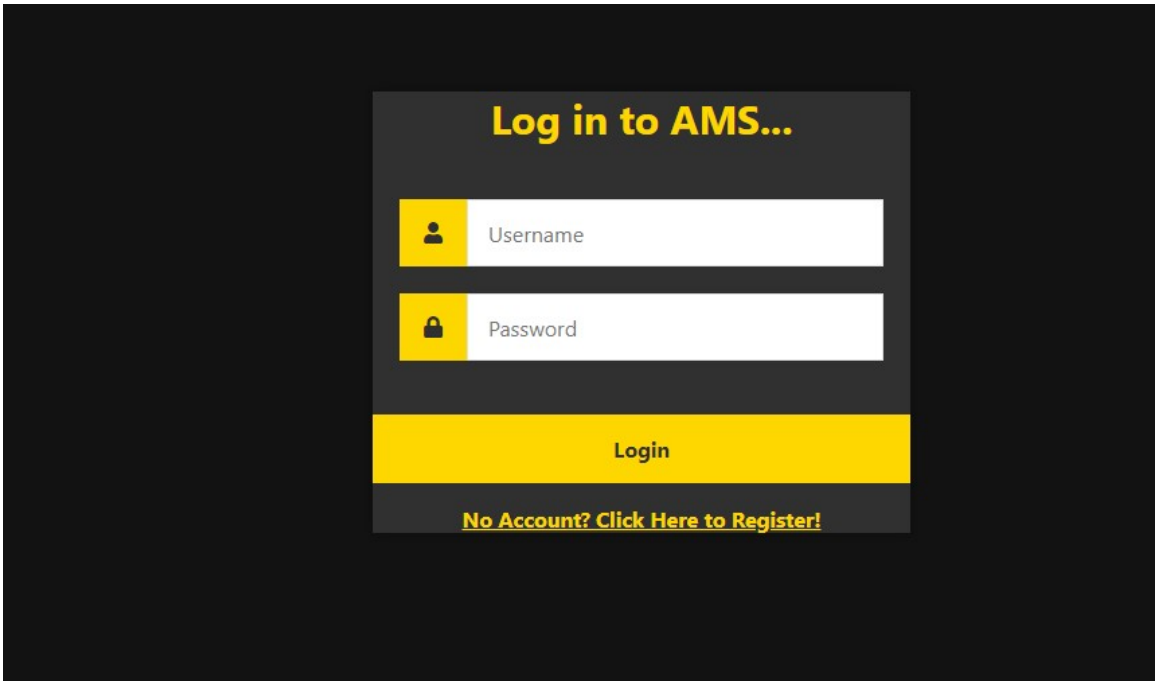

Fig. 2: The login page during the first week of development

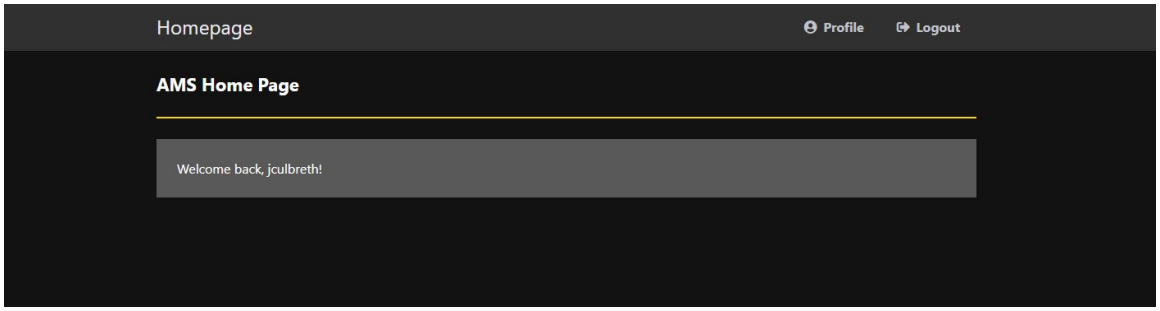

Fig. 3: The prototype “home page” during the first week of development

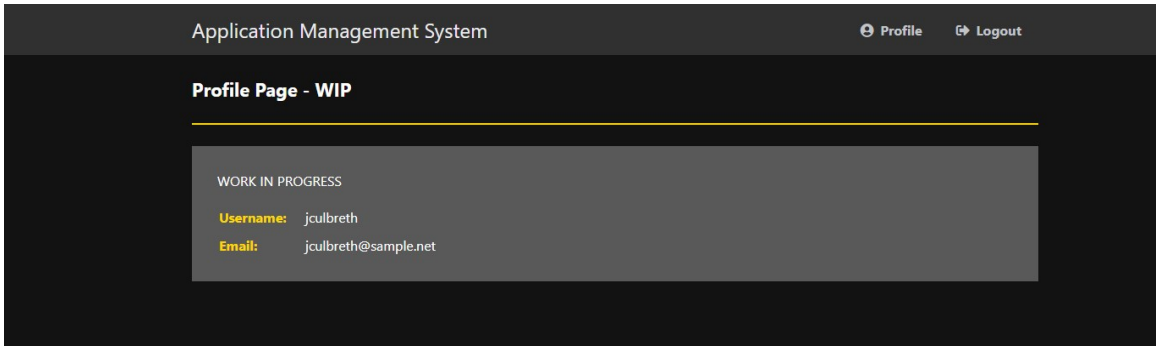

Fig. 4: The prototype profile page during the first week of development

5. Second Week

The second week was spent primarily working on the file upload infrastructure for the project, and further refining the overall design of the interface. Interface alterations made were relatively minor, and predominantly consisted of adding additional iconography when necessary for the various webpages in the navigation bar.

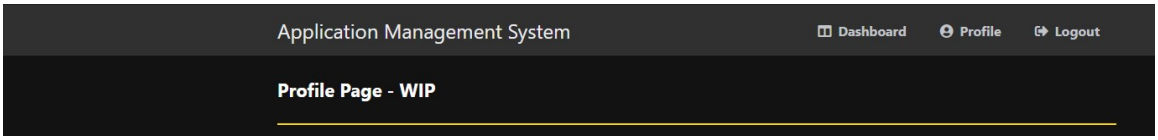

Fig. 5: The “Home page” was renamed “Dashboard” and made accessible from the navigation bar.

The file upload systems required the most work during this phase of development. I implemented two file upload systems – a document upload system, intended for user

resume submissions, and an image upload system, for user profile photos. The resume system was limited to .pdf, .doc, .docx, and even .jpg and .png image types, and resume file uploads are handled by a file named “upload.php” which uses PHP to upload the images to the /uploads_resumes/ folder in the program’s files. For profile photos, only .jpg files are allowed, and these are uploaded to /uploads_imgs/.

During the implementation process for the file upload systems, I ran into a number of issues. The first three were resolved quickly and without issue, and of these the first two were the result of my own error during development. Firstly, I discovered the system would not create the uploads folders automatically during development, which required me to create the folders myself. Second, I set the file upload limit to what I believed to have been around 100mb, and quickly discovered it to have been in actuality around 1mb. This was because I believed that the fileSize function counted in kilobytes, while it in reality counts in bytes. Additionally, 100mb is massive for a resume, so I reduced this to the much more reasonable limit of 50mb, or 52,428,800 bytes, which is likely still a lot larger than necessary. Realistically, I believe this could be restricted to around 5 or 6 megabytes.

The third issue I discovered was perhaps one of the first major hurdles I encountered in terms of time and effort to resolve. Initially, I had configured the user profile page so that the user would be able to select and submit both a profile image and resume from their personal files. However, despite having two selection boxes for the files and one submit button, clicking the “submit" button would only submit one of the files. I worked on this for a considerable length of time and attempted to resolve it by double checking my code and attempting several techniques, including combining the

upload files' scripts into one overall upload.php files and creating two separate buttons to post the individual files. Eventually, I decided to add the resume submission button to the dashboard (though this would change later when I realized the formatting had broken) and left the profile image selection on the user's profile page.

Two larger issues remained, however, that I was unable to resolve until later in development. First, the formatting for the dashboard page was broken with my attempts to display user images and the buttons for submitting said images, and the puzzle of linking specific file uploads to specific users was a larger solution that I was never truly able to resolve, though I was able to work around it. Finally, the profile images and resumes were not named correctly. Resumes were given randomly generated names, which made identification difficult, and profile images were simply uploaded with the name "profile" followed by the file type.

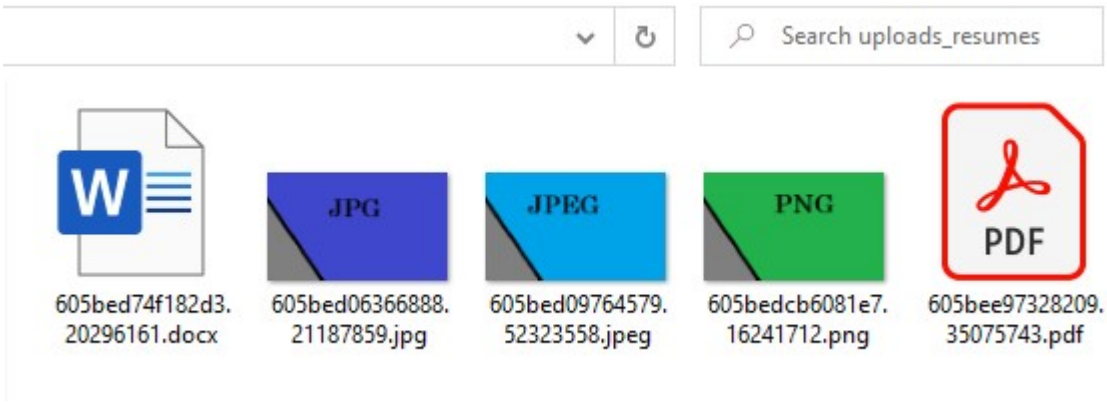

Fig. 6: The "uploads-resumes" folder demonstrating both the file support and the name issue

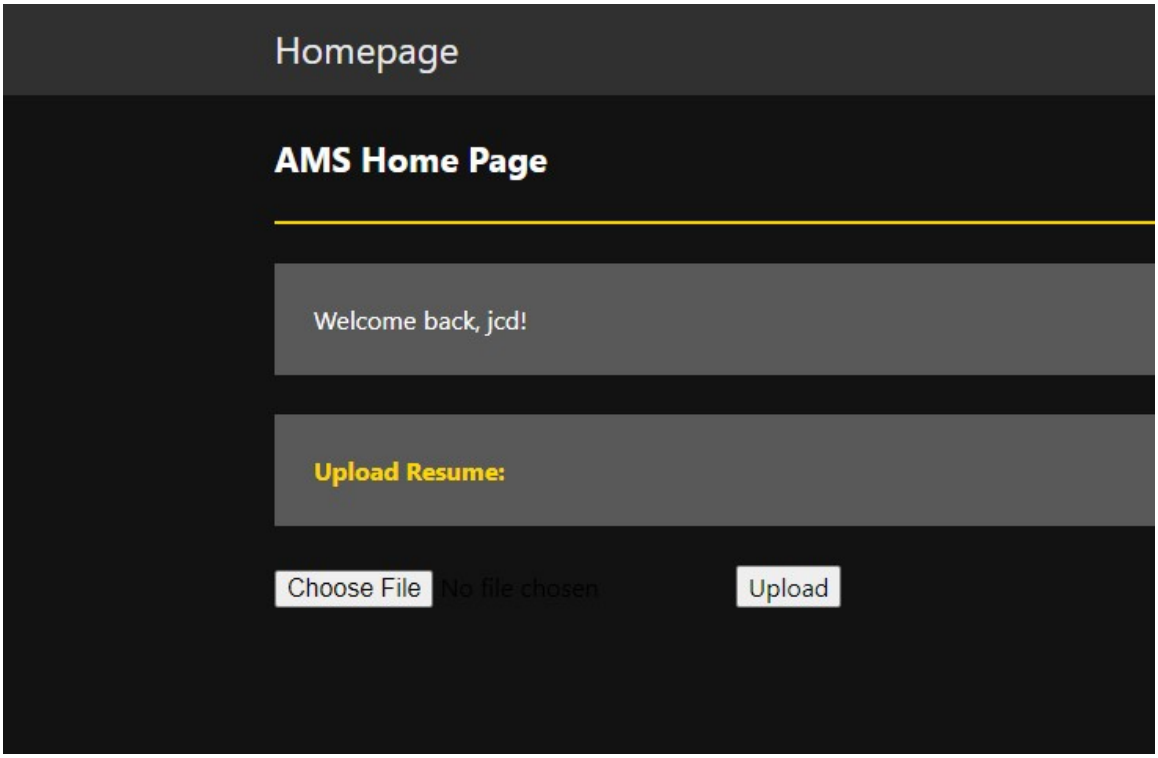

Fig. 7: The “Dashboard” page formatting broken after attempting to add the Resume upload buttons

6. Third Week

The third week was the most difficult stage of development. Much of this week was spent attempting to implement the planned different user classes feature in order to restrict the job listing creation and application processes to the appropriate user groups, as well as the required planning and implementation of the application process itself. In addition to that, numerous other issues were discovered, some of which were resolved.

First, to cover some of the positive progress during this segment, I performed a complete overhaul of the registration system. I felt that the profiles featuring only a username, email, and password was highly limited, and expanded the database's accounts table to include the first name, last name, and phone number, as well, and modified the registration page appropriately to support entry for that additional information. Modifying the registration page to support the additional input occurred in two phases: first, I modified the "register.php" file to support the additional textboxes required for input and to post their submissions to the appropriate database sections. Second, I added the required textboxes to the registration page, and added appropriate icons to maintain the interface design.

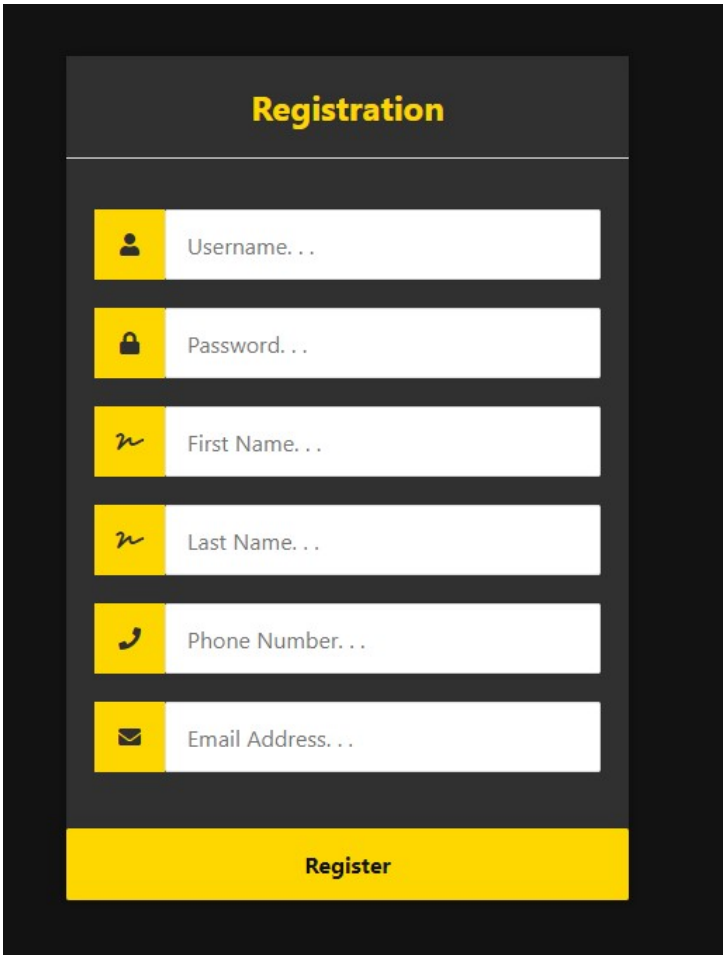

Fig. 8: The redesigned registration page

Next, I focused on the user profile page. I added the additional information to the page, and also displayed the user ID number. This is achieved by outputting the user's content from their appropriate entry in the accounts table of the database, and I also modified the CSS code for this page to change the layout and formatting of the content so that the database content would stay as white text, while the titles for the information and some instructions are in gold colored, bold text. Additionally, I fixed the profile image output through a simple workaround: first, the user must upload the image with the name formatted as

"profileimg_ID.jpg", where ID is their assigned ID number. The user is instructed as to what their ID number is on their profile page, and instructed as to exactly how their profile should be named. This workaround was achieved by "dummying out" the code used to generate the randomly assigned names by turning that code into comments in the files, and by having the system check the uploads_pimgs folder for an appropriately named file that matches the user's ID number.

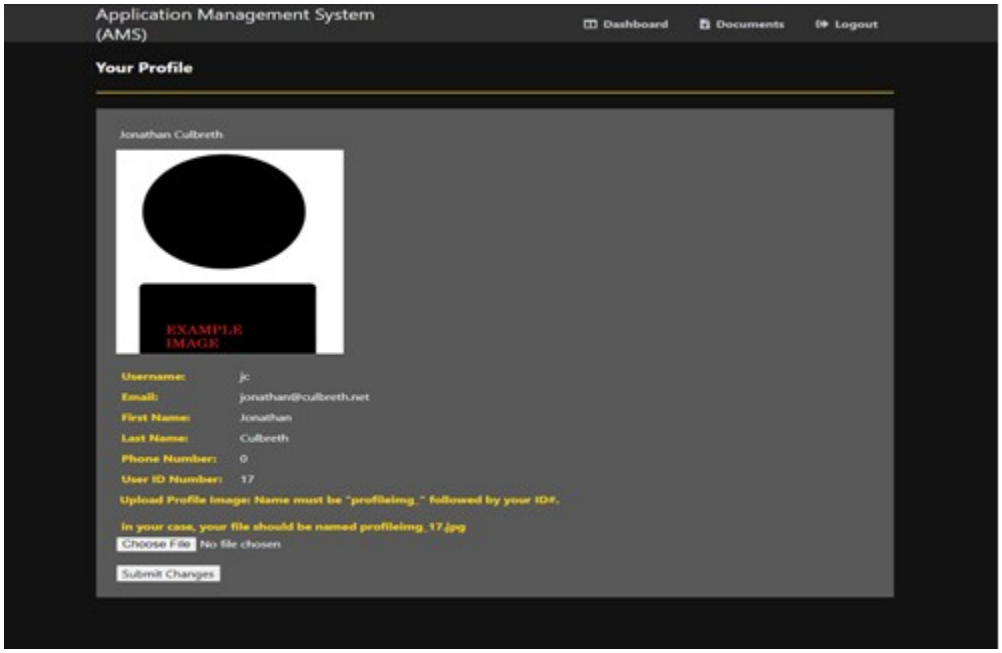

Fig. 9: Example profile page with animage displayed. Additionally visible is the "Documents" icon

Additionally, I added a "Your Documents" page, which was navigated to via a "Documents" icon in the navigation bar at the top of the screen. To fix the broken formatting on the dashboard, I copied the old version of the "Profile" page and modified

it so that it provided an upload option for a resume file, though the system in its current implementation only supports basic file uploads.

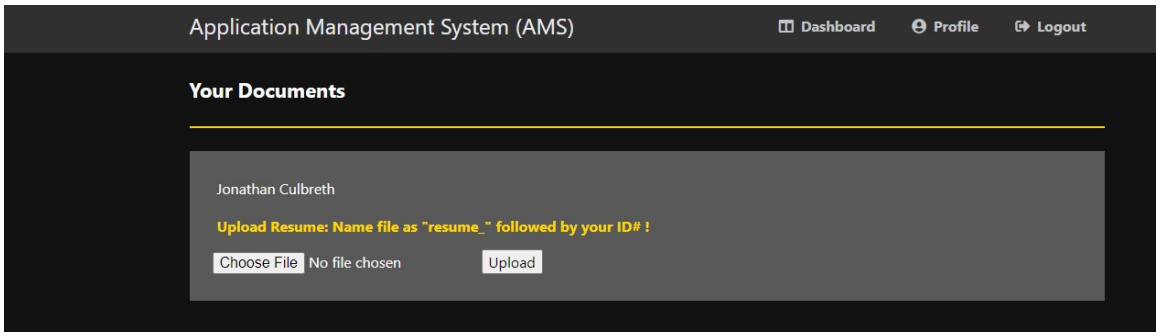

Fig. 10: File upload option for documents page

After these were implemented, most of the time spent during the third week was focused on the job listing and application system itself. Eventually, the application system was scrapped at the recommendation of my mentor, Dr. Zhang, as it was simply unfeasible. I began by creating a "positions" table in my database, which initially contained only the status, name, and description of the job listings, along with another automatically incremented and assigned ID number. From there, I eventually decided to display the informations from these listings in the form of a table on the dashboard page. This was achieved by having the system go through a simple "while" loop to print the contents of each entry of the positions table into a preformatted table on the dashboard webpage until all entries had been printed.

After implementing the basic output stage, however, I decided that I wanted to program an option for users to create new job listings, as I realized that would have been a fairly simple enough process, as I could repurpose some of my other code in order to do so. Since the job applications and registration features would be performed in roughly the

same way (that is, posted to the database via PHP), I realized that I could essentially duplicate the process used for the registration process, rename it to “joblisting.php”, and modify it in order to post the appropriate information in the correct columns in the positions table. At this point, I also expanded the positions table to include contact number and contact email for the position, as well as the requirements for the position.

With a method determined, I worked on the job listing creation page, which I also based on the old prototype version of the dashboard page. I added the appropriate text entry boxes, and the submission button. Afterwards, I added an appropriate button to the navigation bar for the other webpages in the system.

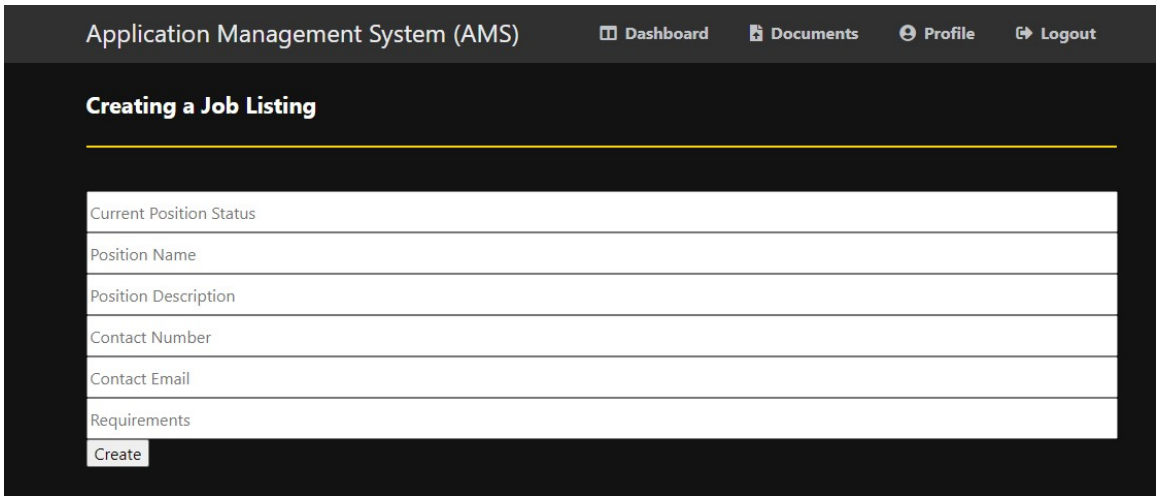

Fig. 10: Creating a Job Listing menu

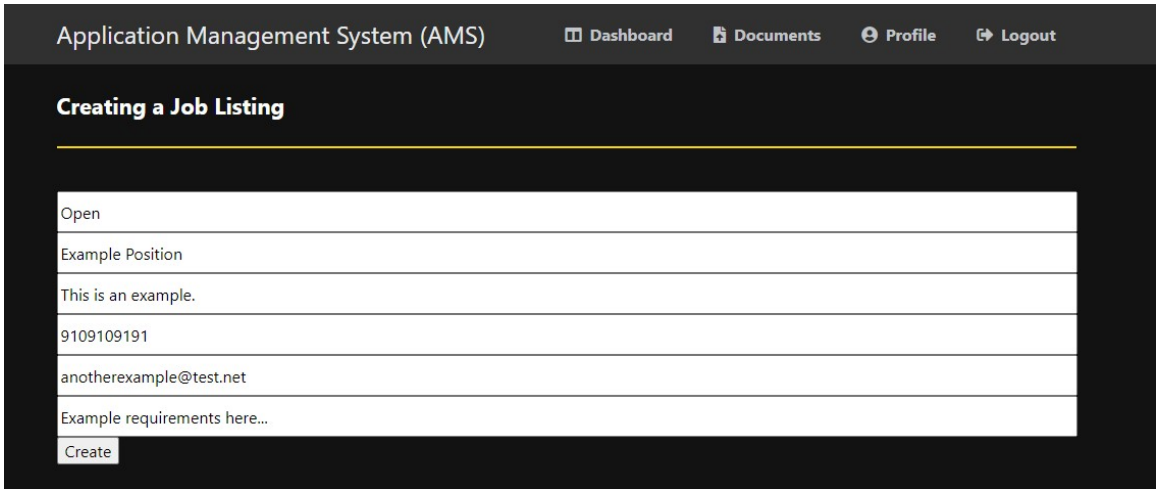

Fig. 11: Sample entry in the Creating a Job Listing menu

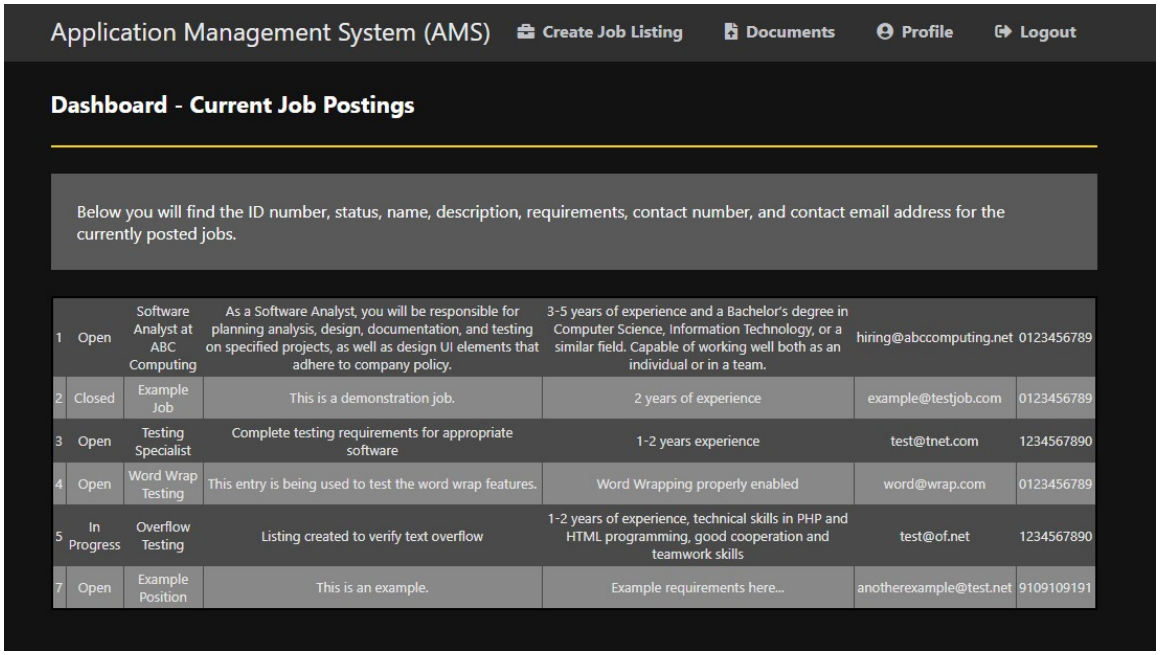

Fig. 12: Sample entry after successful post to the database, as well as the output of the positions database table

7. Final Touches

For the final touches on the system, I worked on cleaning up the system's files by removing unnecessary code and database tables. During my attempts at creating the user classification system, for example, I had created three tables that would later prove to be no longer necessary, and as a result I removed those from my database. Next, I deleted several of the files used over the course of development to test the file upload systems, as well as some of the unnecessary files that were built for previously planned features, such as those originally planned for the email verification system. Finally, I reviewed each remaining file for my webpage to ensure there were no typos in my code or remaining unnecessary sections of code. At that point, I also removed code that had been previously "dummied" out, such as those previously used for generating file names.

8. User Testing

After creating the system, I organized for six participants to test the system and respond to a brief survey after being asked to access the system. For this, their tasks were to register, check their profile, browse job listings, and create their own listing. The questionnaire was given afterwards and featured five questions that asked users to evaluate the system by selecting one of the following options: Very Bad, Bad, Neutral/No Opinion, Good, Very Good. Three additional questions provide "Yes" or "No" answers, and one additional free response style question was included.

The questions were focused on the ease of access for the interface, the color scheme, the organization of the interface, the quality of information provided for the

job listings, and the quality of features included for the user profiles. Generally, the results of user testing revealed that the lack of an application system itself, the lack of information available for profiles and companies, and the lack of an ability to view other user profiles were large issues. An additional finding is that one participant recommended the "Dark mode" style UI instead be optional, as they believe that lighter interfaces are easier to read and stated that they believe systems that offer the option for both are overall best.

9. Conclusions and Future Work

I successfully completed the design and implementation of a job application system with very basic backbone features. Originally, I planned for far more advanced features, but due to the time constraints and the fact that I worked alone on this project, I was unable to implement the additional features due to the time limitation. For the application to truly be useful and efficient in its final form, several additional advanced features would need to be implemented. At a minimum, it would need to be hosted online, it would likely need different user classes so that only certain users could create job listings, it would need the application functionality I had originally planned, and the job listings would need additional information that could be provided through the form of profile pages for organizations. The user testing confirmed many of these beliefs while also highlighting some of the system's best features and revealing some additional downsides that it featured that could be further refined in the future. Overall, however, I

feel that I have learned a lot from this project, particularly regarding PHP and SQL programming as well as MyPHP and SQL database management.

Appendix

Source Code

For this segment, I will provide the source code for my project's files. The name will be in italics with a brief description, followed by the source code itself.

Style.css – This file defines the layout and design for the webpages

```
* {
	box-sizing: border-box;
	font-family: -apple-system, BlinkMacSystemFont, "segoeui", roboto, oxygen, ubuntu, cantarell, "fira sans", "droid sans", "helveticaneue", Arial, sans-serif;
	font-size: 16px;
	-webkit-font-smoothing: antialiased;
	-moz-osx-font-smoothing: grayscale;
}

body {
	background-color: #121212;
}
.login {
	width: 400px;
	background-color: #303030;
	box-shadow: 0 0 9px 0 rgba(0, 0, 0, 0.3);
	margin: 100px auto;
}
.login h1 {
	text-align: center;
	color: #FFD700;
	font-size: 30px;
	padding: 20px 5px 20px 0:
	border-bottom: 1px solid #dee0e4;
}

.login form {
	display: flex;
	flex-wrap: wrap;
	justify-content: center;
	padding-top: 20px;
}
.login form label {
	display: flex;
	justify-content: center;
	align-items: center;
	width: 50px;
```

```
        height: 50px;
        background-color: #FFD700;
        color: #303030;
}
.login form input[type="password"], .login form input[type="text"] {
        width: 310px;
        height: 50px;
        border: 1px solid #dee0e4;
        margin-bottom: 20px;
        padding: 0 15px;
}

.login form input[type="submit"] {
        width: 100%;
        padding: 15px;
        margin-top: 20px;
        background-color: #FFD700;
        border: 0;
        cursor: pointer;
        font-weight: bold;
        color: #303030;
        transition: background-color 0.2s;
}

.login form input[type="submit"]: hover {
        background-color: #303030;
        transition: background-color 0.2s;
}

/* register button */
.login ul {
  list-style-type: none;
  margin: 0;
  padding: 0;
  overflow: hidden;
  background-color: #FFD700;
}

.login ul {
  display: table;
  margin: 0 auto;
}

.login li a {
  display: block;
  padding: 100px 20px;
```

```
}


/*Styles for Dashboard */
.navtop {
	background-color: #303030;
	height: 60px;
	width: 100%;
	border: 0;
}
.navtop div {
	display: flex;
	margin: 0 auto;
	width: 1000px;
	height: 100%;
}
.navtop div h1, .navtop div a {
	display: inline-flex;
	align-items: center;
}
.navtop div h1 {
	flex: 1;
	font-size: 24px;
	padding: 0;
	margin: 0;
	color: #eaebed;
	font-weight: normal;
}
.navtop div a {
	padding: 0 20px;
	text-decoration: none;
	color: #c1c4c8;
	font-weight: bold;
}
.navtop div a i {
	padding: 2px 8px 0 0;
}
.navtop div a:hover {
	color: #eaebed;
}
body.loggedin {
	background-color: #121212;

}
.content {
```

```
        width: 1000px;
        height: 10000px;
        margin: 0 auto;
        background-color: #121212;

}
.content h2 {
        margin: 0;
        padding: 25px 0;
        font-size: 22px;
        border-bottom: 2px solid #e0e0e3;
        color: #fff;
        border-color: #FFD700;
}
.content> p, .content > div {
        box-shadow: 0 0 5px 0 rgba(0, 0, 0, 0.1);
        margin: 25px 0;
        padding: 25px;
        background-color: #595959;
        color: #fff;
}
.content> p table td, .content > div table td {
        padding: 5px;
}

.content> p table, .content > div table {
                box-sizing: border-box;
        font-family: -apple-system, BlinkMacSystemFont, "segoeui", roboto, oxygen,
ubuntu, cantarell, "fira sans", "droid sans", "helveticaneue", Arial, sans-serif;
        font-size: 16px;
        -webkit-font-smoothing: antialiased;
        -moz-osx-font-smoothing: grayscale;
}
.content> p table td:first-child, .content > div table td:first-child {
        font-weight: bold;
        color: #FFD700;
        padding-right: 15px;
}
.content> div p {
        padding: 5px;
        margin: 0 0 10px 0;
}


/* Styles for registration page */
* {
```

```
        box-sizing: border-box;
        font-family: -apple-system, BlinkMacSystemFont, "segoeui", roboto, oxygen,
ubuntu, cantarell, "fira sans", "droid sans", "helveticaneue", Arial, sans-serif;
        font-size: 16px;
        -webkit-font-smoothing: antialiased;
        -moz-osx-font-smoothing: grayscale;
}
body {
        background-color: #121212;
        margin: 0;
}
.register {
        width: 400px;
        background-color: #303030;
        box-shadow: 0 0 9px 0 rgba(0, 0, 0, 0.3);
        margin: 100px auto;
}
.register h1 {
        text-align: center;
        color: #FFD700;
        font-size: 24px;
        padding: 20px 0 20px 0;
        border-bottom: 1px solid #dee0e4;
}
.register form {
        display: flex;
        flex-wrap: wrap;
        justify-content: center;
        padding-top: 20px;
}
.register form label {
        display: flex;
        justify-content: center;
        align-items: center;
        width: 50px;
        height: 50px;
        background-color: #FFD700;
        color: #303030;
}
.register form input[type="password"], .register form input[type="text"], .register
form input[type="email"] {
        width: 310px;
        height: 50px;
        border: 1px solid #dee0e4;
        margin-bottom: 20px;
        padding: 0 15px;
```

```
}
.register form input[type="submit"] {
	width: 100%;
	padding: 15px;
	margin-top: 20px;
	background-color: #FFD700;
	border: 0;
	cursor: pointer;
	font-weight: bold;
	color: #121212;
	transition: background-color 0.2s;
}
.register form input[type="submit"]:hover {
	background-color: #FFD700;
	transition: background-color 0.2s;
}
```

Authenticate.php – This file is used for login

```
<?php
session_start();
$DATABASE_HOST = 'localhost';
$DATABASE_USER = 'root';
$DATABASE_PASS = '';
$DATABASE_NAME = 'ams';

// attempts connection with above info
$con = mysqli_connect($DATABASE_HOST, $DATABASE_USER, $DATABASE_PASS,
$DATABASE_NAME);

if ( mysqli_connect_errno() ) {
	// displays error if connection error occurs
	exit('Failed to connect to MySQL: ' . mysqli_connect_error());
}

// check is data from login form exists and was sent
if ( !isset($_POST['username'], $_POST['password']) ) {
	exit('Please fill both the username and password fields!');
}

// Prepare our SQL, which prevents SQLi
if ($stmt = $con->prepare('SELECT id, password FROM accounts WHERE username
= ?')) {
	// Bind parameters (s = string, i = int, etc) - username is a string so "s" is
used
```

```
        $stmt->bind_param('s', $_POST['username']);
        $stmt->execute();
        // Store the result to check if the account exists
        $stmt->store_result();

if ($stmt->num_rows> 0) {
        $stmt->bind_result($id, $password);
        $stmt->fetch();
        // Account exists, verify password

        if (password_verify($_POST['password'], $password)) {
                // Verification successful, user is now logged in
                // Create sessions
                session_regenerate_id();
                $_SESSION['loggedin'] = TRUE;
                $_SESSION['name'] = $_POST['username'];
                $_SESSION['id'] = $id;
                header('Location: home.php');


//incorrect username/password use same message as a simple security measure
        } else {
                // Incorrect password
                echo 'Incorrect username and/or password!';
        }
} else {
        // Incorrect username
        echo 'Incorrect username and/or password!';
}
        $stmt->close();
}
?>
```

index.html – This is the file for the "login" page itself

```
<!DOCTYPE html>

<html>

        <head>

        <meta charset="utf-8">

        <title>Application Management System: Login</title>

        <link href="style.css" rel="stylesheet" type="text/css">
```

```
        <link rel="stylesheet"
href="https://use.fontawesome.com/releases/v5.7.1/css/all.css">


        <style>
        p {text-align: center;}
        </style>


        </head>
        <body>
        <div class="login">
                <h1>Log in to AMS...</h1>


                                <form action="authenticate.php" method="post">
                                <label for="username">
                                        <i class="fas fa-user"></i>
                                </label>
        <input type="text" name="username" placeholder="Username" id="username"
required>
                                <label for="password">
                                        <i class="fas fa-lock"></i>
                                </label>
                                <input type="password" name="password"
placeholder="Password" id="password" required>
```

```
                    <input type="submit" value="Login">
                </form>

<p><b><a href="http://localhost/SrProjRev2/register.html" style="color:#ffD700"; text-
decoration="none" text-align="center" >No Account? Click Here to
Register!</a></p></b>
                </div>
        </body>
</html>
```

register.html – This is the registration page

```
<!DOCTYPE html>
<html>

<head>
<meta charset="utf-8">
<title> Registration</title>
<link href="style.css" rel="stylesheet" type="text/css">
<link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.7.1/css/all.css">
</head>

<body>
```

```
<div class="register">

<h1>Registration</h1>


<form action="register.php" method="post" autocomplete="off">


<label for="username">

<i class="fas fa-user"></i>

</label>

<input type="text" name="username" placeholder="Username. . . " id="username"

required>



<label for="password">

<i class="fas fa-lock"></i>

</label>

<input type="password" name="password" placeholder="Password. . . " id="password"

required>


<label for="fname">

<i class="fas fa-signature"></i>

</label>

<input type="text" name="fname" placeholder="First Name. . . " id="fname" required>
```

```
<label for="lname">
<i class="fas fa-signature"></i>
</label>
<input type="text" name="lname" placeholder="Last Name. . . " id="lname" required>



<label for="phone">
<i class="fas fa-phone"></i>
</label>
<input type="text" name="phone" placeholder="Phone Number. . . " id="phone"
required>



<label for="email">
<i class="fas fa-envelope"></i>
</label>
                                    <input type="email" name="email" placeholder="Email
Address. . . " id="email" required>
                                    <input type="submit" value="Register">
</form>



</div>
```

```
</body>
</html>
```

register.php – This is used to submit the information from the registration page to the database

```
<?php
//connection information for database
$DATABASE_HOST = 'localhost';
$DATABASE_USER = 'root';
$DATABASE_PASS = '';
$DATABASE_NAME = 'ams';

//attempts connection
$con = mysqli_connect($DATABASE_HOST, $DATABASE_USER,
$DATABASE_PASS, $DATABASE_NAME);


if (mysqli_connect_errno()) {
	//stops script and displays error if error occurs
	exit('Failed to connect to MySQL: ' . mysqli_connect_error());
}
```

```
if (!isset($_POST['username'], $_POST['password'], $_POST['email'])) {
	exit('All information is required!');
}
// Make sure the submitted registration values are not empty.
if (empty($_POST['username']) || empty($_POST['password']) ||
empty($_POST['email'])) {
	// One or more values are empty.
	exit('All information is required!');
}
//check if username uses invalid characters
if (preg_match('/^[a-zA-Z0-9]+$/', $_POST['username']) == 0) {
exit('Username is not valid!');
}
//check if email is valid
if (!filter_var($_POST['email'], FILTER_VALIDATE_EMAIL)) {
	exit('Email is not valid!');
}
//Checks if username is already claimed
if ($stmt = $con->prepare('SELECT id, password FROM accounts WHERE username =
?')) {
	// Bind parameters, hash the password using PHP password_hash
	$stmt->bind_param('s', $_POST['username']);
```

```
        $stmt->execute();
        $stmt->store_result();
        // Store the result to check if the account exists in the database.
        if ($stmt->num_rows> 0) {
                // Username already exists
                echo 'Username exists, please choose another!';
        } else {
                // Username is not claimed - create account
if ($stmt = $con->prepare('INSERT INTO accounts (username, password, fname, lname,
phone, email) VALUES (?, ?, ?, ?, ?, ?)')) {
        // password is hashed and verified upon login
        $password = password_hash($_POST['password'], PASSWORD_DEFAULT);
        $stmt->bind_param('ssssis', $_POST['username'], $password, $_POST['fname'],
$_POST['lname'], $_POST['phone'], $_POST['email']);
        $stmt->execute();
                $stmt->close();
        echo 'You have successfully registered, you can now login!';
}
}}
$con->close();


header('Location: index.html');
exit();
```

```
?>
```

home.php – This is the dashboard page, and provides the user with the job listings

```
<?php
 //session information
session_start();
if (!isset($_SESSION['loggedin'])) {
        header('Location: index.html');
        exit;
}
?>

<!DOCTYPE html>
<html>



<head>
<style>
table {
  font-family: "Arial Black", Gadget, sans-serif;
  border: 2px solid #000000;
  background-color: #4A4A4A;
```

```
  width: 100%;
  height: 200px;
  text-align: center;
  border-collapse: collapse;
}
table td, table th {
  border: 1px solid #4A4A4A;
  padding: 3px 2px;
}
table tbody td {
  font-size: 13px;
  color: #E6E6E6;
}
table tr:nth-child(even) {
  background: #888888;
}
table thead {
  background: #000000;
  border-bottom: 3px solid #000000;
}
table theadth {
  font-size: 15px;
  font-weight: bold;
```

```
  color: #E6E6E6;
  text-align: center;
  border-left: 2px solid #4A4A4A;
}
table theadth:first-child {
  border-left: none;
}

table tfoot {
  font-size: 12px;
  font-weight: bold;
  color: #E6E6E6;
  background: #000000;
  background: -moz-linear-gradient(top, #404040 0%, #191919 66%, #000000 100%);
  background: -webkit-linear-gradient(top, #404040 0%, #191919 66%, #000000 100%);
  background: linear-gradient(to bottom, #404040 0%, #191919 66%, #000000 100%);
  border-top: 1px solid #4A4A4A;
}
table tfoot td {
  font-size: 12px;
}
</style>
```

```
<meta charset="utf-8">
		<title>Application Management System - Dashboard</title>
		<link href="style.css" rel="stylesheet" type="text/css">
		<link rel="stylesheet"
href="https://use.fontawesome.com/releases/v5.7.1/css/all.css">
		</head>
		<body class="loggedin">
		<nav class="navtop">
		<div>
		<h1> Application Management System (AMS)</h1>
		<a href="create.html"><i class="fas fa-briefcase"></i>Create Job
Listing</a>
		<a href="docs.php"><i class="fas fa-file-
upload"></i></i>Documents</a>
		<a href="profile.php"><i class="fas fa-user-circle"></i>Profile</a>
		<a href="logout.php"><i class="fas fa-sign-out-alt"></i>Logout</a>

		</div>
	</nav>
	<div class="content">
	<h2> Dashboard - Current Job Postings</h2>
```

```
<p> Below you will find the ID number, status, name, description, requirements, contact
number, and contact email address for the currently posted jobs.</p>
<?php
$DATABASE_HOST = 'localhost';
$DATABASE_USER = 'root';
$DATABASE_PASS = '';
$DATABASE_NAME = 'ams';
$con = mysqli_connect($DATABASE_HOST, $DATABASE_USER,
$DATABASE_PASS, $DATABASE_NAME);

$query = mysqli_query($con, "SELECT * FROM positions");

echo "<table>"; //opens HTML table
while($row = mysqli_fetch_array($query)){   //Creates a loop to loop through results
echo "<tr><td>" . $row['pId'] . "</td><td>" . $row["pStatus"] . "</td><td>" .
$row['pName'] . "</td><td>" . $row['pDescription'] . "</td><td>" . $row['pReq'] .
"</td><td>" . $row['pEmail'] . "</td><td>" . $row['pNum'] . "</td></tr>";
//$row['index'] the index here is a field name
}

echo "</table>"; //closes HTML table
        ?>
```

```
        </div>
        </body>
        </html>
```

profile.php – This is the profile page

```
<?php
//Starts session
session_start();
include_once 'upload_img.php'; //this is for the profile image to appear.

// Logged in? If no, redirect to login page...
if (!isset($_SESSION['loggedin'])) {
        header('Location: index.html');
        exit;
}

//db connection
$DATABASE_HOST = 'localhost';
$DATABASE_USER = 'root';
$DATABASE_PASS = '';
$DATABASE_NAME = 'ams';
```

```
$con = mysqli_connect($DATABASE_HOST, $DATABASE_USER,
$DATABASE_PASS, $DATABASE_NAME);
if (mysqli_connect_errno()) {
        exit('Failed to connect to MySQL: ' . mysqli_connect_error());
}
// Pulls user data from database
$stmt = $con->prepare('SELECT fname, lname, phone, password, email FROM accounts
WHERE id = ?');



//Uses account ID# to pull info
$stmt->bind_param('i', $_SESSION['id']); //bases session info on account info
$stmt->execute();
$stmt->bind_result($fname, $lname, $phone, $password, $email);
$stmt->fetch();
$stmt->close();
?>



<!DOCTYPE html>
<html>
        <head>
                <meta charset="utf-8">
```

```
        <title>Application Management System - Profile Page</title>
        <link href="style.css" rel="stylesheet" type="text/css">
        <link rel="stylesheet"
href="https://use.fontawesome.com/releases/v5.7.1/css/all.css">
    </head>
    <body class="loggedin">
        <nav class="navtop">
            <div>
                <h1>Application Management System (AMS)</h1>
                <a href="create.html"><i class="fas fa-
briefcase"></i>Create Job Listing</a>
                <a href="home.php"><i class="fas fa-
columns"></i>Dashboard</a>
                <a href="docs.php"><i class="fas fa-file-
upload"></i></i>Documents</a>
                <a href="logout.php"><i class="fas fa-sign-out-
alt"></i>Logout</a>
            </div>
        </nav>
        <div class="content">
            <h2>Your Profile</h2>
            <div>
```

```
                    <p><?=$fname ?><?=$lname?></p>

                                <table>
                                <td align-right>
                                <?php
//profile image recovery
$sql = "SELECT * FROM accounts";
$result = mysqli_query($conn, $sql);
if (mysqli_num_rows($result) > 0) {
        while ($row = mysqli_fetch_assoc($result)) {
                $id = $row['id'];
                $sqlImg = "SELECT * FROM profileimg WHERE userid='$id'";
                $resultImg = mysqli_query($conn, $sqlImg);
                while ($rowImg = mysqli_fetch_assoc($resultImg)) {
                        echo "<div class='user-container'>";
                        if ($rowImg['status'] == 0) {
                                echo
"<imgsrc='uploads_pimgs/profileimg_".$id.".jpg?'".mt_rand().">";
                        } else {
```

```
                    echo "imgsrc='uploads_pimgs/default.jpg'>";
                }
                echo "<p>".$row['username']."<p>";
                echo "</div>";
}}
}
?></td>

                    <tr>
                    <imgsrc='uploads_pimgs/profileimg_<?=$id?>.jpg'
style="width:30%; height:30%;">
                    </tr>

                    <tr>
                        <td>Username:</td>
                        <td style=" color: white";
><?=$_SESSION['name']?></td>
                    </tr>

                    <tr>
                        <td>Email:</td>
                        <td style=" color: white";
><?=$email?></td>
```

```
                </tr>

                <tr>
                <td> First Name:</td>
                <td style="color:white";><?=$fname?></td>
                </tr>

                <tr>
                <td> Last Name:</td>
                <td style="color:white";><?=$lname?></td>
                </tr>

                <tr>
                <td> Phone Number: </td>
                <td style="color:white";><?=$phone?></td>
                </tr>

                <tr>
                <td> User ID Number: </td>
                <td style="color:white";><?=$id?></td>
                </tr>

            </table>
```

```
<p style=" color: #FFD700"; ><b> Upload Profile Image: Name must be "profileimg_"
followed by your ID#.
<table>
<tr>
<td> in your case, your file should be named profileimg_<?=$id?>.jpg</td>
</tr>
</table>

<form action="upload_img.php" method="POST" enctype="multipart/form-data">
<input type="file" name="file">
<br>
<br>
<button type="submit" name="submit">Submit Changes</button>
                        </div>
                </div>
        </body>
</html>
```

upload.php – This is the code used to upload the resumes.

```
<?php
        if (isset($_POST['submit'])) {
```

```
$file = $_FILES['file'];

//assigns values
$fileName = $_FILES['file']['name'];
$fileTmpName = $_FILES['file']['tmp_name'];
$fileSize = $_FILES['file']['size'];
$fileError = $_FILES['file']['error'];
$fileType = $_FILES['file']['type'];

$fileExt = explode('.', $fileName);
$fileActualExt = strtolower(end($fileExt));
$allowed = array('jpg', 'jpeg', 'png', 'pdf', 'docx', 'doc'); //allowed types of files

if (in_array($fileActualExt, $allowed)) {
    if ($fileError === 0) {
    if ($fileSize< 52428800) { //file size limit is 50mb, filesize is in bytes
        $fileDestination = 'uploads_resumes/'.$fileName; //stores files in
folder under localhost
        move_uploaded_file($fileTmpName, $fileDestination); //see above
- moves files from temp location to permanent location
```

```
                        header("Location: home.php?uploadsuccess"); //changes address
name is transfer succeeds, no other effect
                } else {
                        echo "Your file exceeds the maximum file size limitation!"; // file
>50mb
                }


                } else {
                        echo "File upload error!"; //generic error
                        }
        } else {
                echo "Invalid file type!"; //file is unauthorized type, e.g. bmp, tiff, ppt/x
        }
}
```

dbh.php – This file is used to establish a database connection for image uploads. This was implemented primarily as an experiment late into development and could likely have been reused elsewhere.

```
<?php

$conn = mysqli_connect("localhost", "root", "", "ams");
```

Upload_img.php – This file is used to upload the profile image users can provide in their profile page.

```
<?php
include_once 'dbh.php';


        if (isset($_POST['submit'])) {
        $file = $_FILES['file'];

        //assigns values
        $fileName = $_FILES['file']['name'];
        $fileTmpName = $_FILES['file']['tmp_name'];
        $fileSize = $_FILES['file']['size'];
```

```
        $fileError = $_FILES['file']['error'];
        $fileType = $_FILES['file']['type'];

        $fileExt = explode('.', $fileName);
        $fileActualExt = strtolower(end($fileExt));

        $allowed = array('jpg',); //allowed types of files - in this case only jpg

        //file submission content
        if (in_array($fileActualExt, $allowed)) {
                if ($fileError === 0) {
                if ($fileSize< 30428800) { //file size limit is in bytes
                        $fileDestination = 'uploads_pimgs/'.$fileName; //stores files in
folder under localhost
                        move_uploaded_file($fileTmpName, $fileDestination); //see above
- moves files from temp location to permanent location
                        $sql = "UPDATE profileimg SET status=0 WHERE userid =
'$id';";
                        $result = mysqli_query($conn, $sql);

                        header("Location: profile.php?uploadsuccess"); //changes address
name is transfer succeeds, no other effect
                } else {
```

```
                        echo "Your file exceeds the maximum file size limitation!"; // file
>50mb
                }


                } else {
                        echo "File upload error!"; //generic error
                        }
        } else {
                echo "Invalid file type!"; //file is unauthorized type, e.g. bmp, tiff, ppt/x
        }
}
```

docs.php– This is the document upload page

```
<?php
//Starts session
session_start();
include_once 'upload_img.php'; //this is for the profile image to appear.

// checks if user is logged in, otherwise redirects to login page
if (!isset($_SESSION['loggedin'])) {
        header('Location: index.html');
```

```
        exit;
}

//db connection
$DATABASE_HOST = 'localhost';
$DATABASE_USER = 'root';
$DATABASE_PASS = '';
$DATABASE_NAME = 'ams';
$con = mysqli_connect($DATABASE_HOST, $DATABASE_USER,
$DATABASE_PASS, $DATABASE_NAME);
if (mysqli_connect_errno()) {
        exit('Failed to connect to MySQL: ' . mysqli_connect_error());
}
// Pulls user data from database
$stmt = $con->prepare('SELECT fname, lname, phone, password, email FROM accounts
WHERE id = ?');


//Uses account ID# to pull info
$stmt->bind_param('i', $_SESSION['id']); //bases session info on account info
$stmt->execute();
$stmt->bind_result($fname, $lname, $phone, $password, $email);
$stmt->fetch();
```

```
$stmt->close();
?>



<!DOCTYPE html>
<html>
	<head>
		<meta charset="utf-8">
		<title>Application Management System - Profile Page</title>
		<link href="style.css" rel="stylesheet" type="text/css">
		<link rel="stylesheet"
href="https://use.fontawesome.com/releases/v5.7.1/css/all.css">
	</head>
	<body class="loggedin">
		<nav class="navtop">
			<div>
				<h1>Application Management System (AMS)</h1>
				<a href="create.html"><i class="fas fa-
briefcase"></i>Create Job Listing</a>
				<a href="home.php"><i class="fas fa-
columns"></i>Dashboard</a>
				<a href="profile.php"><i class="fas fa-user-
circle"></i>Profile</a>
```

```
                    <a href="logout.php"><i class="fas fa-sign-out-
alt"></i>Logout</a>
                </div>
            </nav>
            <div class="content">
                <h2>Your Documents</h2>
                <div>

                <p><?=$fname ?><?=$lname?></p>

                <p style=" color: #FFD700"; ><b> Upload Resume: Name file as
"resume_" followed by your ID# !</b>
</p>

<table>
<tr>

<form action="upload.php" method="POST" enctype="multipart/form-data">
<input type="file" name="file">

</tr>
<button type="submit" name="submit">Upload</button>
```

create.html – This is the page where users can create new job listings.

```
<?php
 //session information
session_start();
if (!isset($_SESSION['loggedin'])) {
        header('Location: index.html');
        exit;
}
?>



<!DOCTYPE html>
<html>
<head>
<style>
  input[type=text]{
      word-wrap: break-word;
      height: 40px;}
</style>

<meta charset="utf-8">
                <title>Application Management System - Dashboard</title>
```

```
                <link href="style.css" rel="stylesheet" type="text/css">
                <link rel="stylesheet"
href="https://use.fontawesome.com/releases/v5.7.1/css/all.css">
                </head>
                <body class="loggedin">
                <nav class="navtop">
                <div>
                <h1> Application Management System (AMS)</h1>
                <a href="home.php"><i class="fas fa-columns"></i>Dashboard</a>
                <a href="docs.php"><i class="fas fa-file-
upload"></i></i>Documents</a>
                <a href="profile.php"><i class="fas fa-user-circle"></i>Profile</a>
                <a href="logout.php"><i class="fas fa-sign-out-alt"></i>Logout</a>

                </div>
        </nav>
        <div class="content">
        <body>
        <h2> Creating a Job Listing</h2>
        <br>
<table>
<tr>
<form action="joblisting.php" method="post" autocomplete="off">
```

```
</tr>
<br>

<tr>
<input type="text" name="pStatus" placeholder="Current Position Status" size="137"
id="pStatus" required>
</tr>
<br>

<tr>
<input type="text" name="pName" placeholder="Position Name" size="137"
id="pName" required>
</tr>
<br>

<tr>
<input type="text" name="pDescription" placeholder="Position Description" size="137"
id="pDescription" required>
</tr>
<br>

<tr>
```

```
<input type="text" name="pNum" placeholder="Contact Number" size="137"
id="pNum" required>
</tr>
<br>

<tr>
<input type="text" name="pEmail" placeholder="Contact Email" size="137"
id="pEmail" required>
</tr>
<br>

<tr>
<input type="text" name="pReq" placeholder="Requirements" size="137" id="pReq"
required>
</tr>
<br>
                                    <input type="submit" value="Create">
</form>

</table>


        </div>
```

```
	</body>
	</html>
```

Joblisting.php – This file is used to post the job listing to the positions table of the database.

```
<?php
//connection information for database
$DATABASE_HOST = 'localhost';
$DATABASE_USER = 'root';
$DATABASE_PASS = '';
$DATABASE_NAME = 'ams';

//attempts connection
$con = mysqli_connect($DATABASE_HOST, $DATABASE_USER,
$DATABASE_PASS, $DATABASE_NAME);


if (mysqli_connect_errno()) {
	//stops script and displays error if error occurs
	exit('Failed to connect to MySQL: ' . mysqli_connect_error());
}
```

```
if ($stmt = $con->prepare('SELECT pId, pName FROM positions WHERE pName = ?'))
{
	$stmt->bind_param('s', $_POST['pName']);
	$stmt->execute();
	$stmt->store_result();
	// Store the result to check if the account exists in the database.
	if ($stmt->num_rows> 0) {
		// Username already exists
		echo 'Listing exists, please choose another!';
	} else {
		// Username is not claimed - create account
if ($stmt = $con->prepare('INSERT INTO positions (pStatus, pName, pDescription,
pNum, pEmail, pReq) VALUES (?, ?, ?, ?, ?, ?)')) {
	// password is hashed and verified upon login
	$stmt->bind_param('ssssss', $_POST['pStatus'], $_POST['pName'],
$_POST['pDescription'], $_POST['pNum'], $_POST['pEmail'], $_POST['pReq']);
	$stmt->execute();
		$stmt->close();
	echo 'Listing Created Successfully!';
}
}}
$con->close();
```

```
header('Location: home.php');
exit();
?>
```

References and Sources

51: Upload Files and Images to Website in PHP | PHP Tutorial | Learn PHP Programming | Image Upload

https://www.youtube.com/watch?v=JaRq73y5MJk (Dani Krossing)

52: How to upload profile images to users using PHP - PHP tutorial

https://www.youtube.com/watch?v=y4GxrIa7MiE(Dani Krossing)

Secure Login System with PHP and MySQL

https://codeshack.io/secure-login-system-php-mysql/ (David Adams)

W3Schools Online Web Tutorials

https://www.w3schools.com/

Additionally, I used https://fontawesome.com/ for icons to use in my CSS styles sheets.