

CRITERION C - DEVELOPMENT

Date commenced: Feb 1st, 2019

Date completed: Feb 19th, 2019

Contents

1.0 Structure	3
2.0 Account framework	4
2.1 New user creation	4
2.2 Login	7
2.3 makeShash()	9
2.4 Security checks	9
2.5 "Terminate other sessions" feature	10
2.6 Account deletion	10
3.0 Features	12
3.1 Job browser and filters	12
3.2 Job seeker dashboard	14
3.3 Employer dashboard	15
3.4 Upload handling	15
4.0 GUI	17
4.1 Adaptive layouts	17
4.2 Dynamic forms	20
Sources cited	21

Advanced techniques used

COMPLEXITY	SECTION(S)
Use of secure cryptography	2.1, 2.2, 2.3
Triple table SQL joins	3.2
Recursion	2.3, 2.6
Nested result filtering	3.1
Converting MySQL table column to an array using a while loop	3.1
Document uploading	3.4
Passing functions as arguments for a while loop	3.1, 3.2, 3.3
Data validation and error handling	2.1, 2.2, 3.4
File and directory handling	2.1, 2.6, 3.4
Complex CSS layouts	4.1

1.0 STRUCTURE

FOLDERS	
IA Website	User folders for file uploads
data	
users	Placeholders for profile pictures
placeholder.svg	
placeholder2.svg	Job browser for job seekers (includes sidebar.php and card.php)
browse.php	Job content cards, displayed on browse.php
card.php	List of countries for option selector inputs
countries.php	Job creation page for employers
createjob.php	Loads one of the two child pages below depending on account type
dashboard.php	Displays a clickable list of hosted jobs, "new job" option
dashboard_employer.php	Displays a clickable list of submitted applications
dashboard_job_seeker.php	Page to edit account details
details.php	Page to edit job details (employers only)
editjob.php	PHP methods used throughout the application
functions.php	Loads stylesheets, JQuery, provides title and viewport settings
header.php	Login page, only accessible if not already logged in
index.php	SQL query to initialize database and tables (for client only)
INIT.sql	List of job categories for easy modification by the client
job_categories.php	Navigation bar, displayed on all post-login pages
menubar.php	"Change password" page
newpass.php	Profile page, holds a settings menu with links (currently privacy.php and profile.php)
profile.php	Page with security options (logout, account deletion, password change etc)
security.php	Holds boilerplate code and performs security checks
session.php	Holds filters and search for the browse.php page
sidepanel.php	User creation page
signup.php	Main CSS stylesheet
style.css	Applicant review page (employers only)
viewapplicant.php	Displays job information and (employer only)
viewjob_employer.php	Displays job information and options (job seeker only)
viewjob_seeker.php	

The web app is divided into multiple pages, some of which are directly accessible by the user, while others hold routines that are loaded on demand.

2.0 ACCOUNT FRAMEWORK

2.1 New user creation

The image shows two side-by-side screenshots of a 'Sign Up' form. The left form is for 'Job Seeker' and the right is for 'Employer'. Both forms have fields for first name, last name, email, password, and repeat password, followed by a 'SIGN UP' button. The right form also has a 'comname' field and a tooltip for the email field.

HTML form data on signup.php page is sent to server using HTTP POST method (secure as long as the client uses SSL). A hook in the page will catch submitted form by using PHP `isset()` function¹ inside an `if` statement.

```
if (isset($_POST["newuser"]) && $_POST["newuser"] == '1') {  
    session_start();  
    makeAcc($_POST["firstname"], $_POST["lastname"], $_POST["comname"], $_POST["email"], $_POST["password"],  
        $_POST["repass"], $_POST["acctype"]);  
    login($_POST["email"], $_POST["password"]);  
    redirect('details.php?msg=new');  
}
```

¹ php.net/manual/function.isset.php

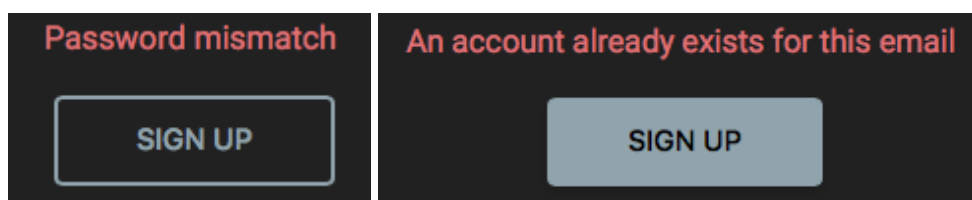
If true, `session_start()` is called to initialize a session² and enable the use of `$_SESSION` variables, used for frequently accessed data to avoid expensive mysqli queries. The form data is passed as arguments to the `makeAcc()` function, shown below.

Array `$mysqli` holds MySQL connection information, returned by the `mysqli()` class, and is needed to perform mysqli queries. Arguments of the `makeAcc()` function containing user input are sanitized using the `mysqli_real_escape_string()` function³, which escapes special characters that could be used to inject unauthorized SQL code.

```
function makeAcc($firstname, $lastname, $comname, $email, $password, $repass, $type) {
    $mysqli = new mysqli("127.0.0.1", "anon", "123", "SIT");

    // SQL injection mitigation
    $email = mysqli_real_escape_string($mysqli, $email);
    $password = mysqli_real_escape_string($mysqli, $password);
    $repass = mysqli_real_escape_string($mysqli, $repass);
    $firstname = mysqli_real_escape_string($mysqli, $firstname);
    $lastname = mysqli_real_escape_string($mysqli, $lastname);
    $comname = mysqli_real_escape_string($mysqli, $comname);
```

The function then performs post-factum data validation (primary email validation is performed on the client side by using the `<input type="email">` HTML tag) and outputs relevant error messages to the user:



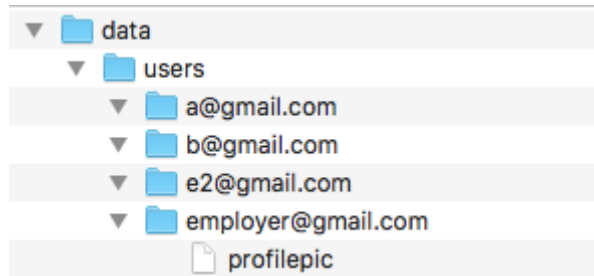
```
// Validate email
if (strpos($email, '@') == false OR strpos($email, '.') == false) {session_destroy();
    header("Location: signup.php?msg=badmail"); die();}
// Validate repeat password
if ($password != $repass) {session_destroy(); header("Location:
    signup.php?msg=mismatch"); die();}
```

```
// Return error if email already in use
$pass = mysqli_query($mysqli, "SELECT email FROM USERS WHERE email='{ $email }' limit 1");
if ($pass->num_rows > 0) {session_destroy(); header("Location: signup.php?msg=mailinuse"); die();}
```

² php.net/manual/function.session-start.php

³ php.net/manual/mysqli.real-escape-string.php

If the data is valid, the user's password is securely hashed using `password_hash()` function⁴ (the actual password is never stored in plaintext, which is an essential security measure). Data is added to the database using a `mysql` INSERT query based on the user's type, and a user specific directory is created for using the OS-agnostic `mkdir()` function⁵, to be used for file uploads (3.4).



```
$phash = password_hash($password, PASSWORD_DEFAULT); // Hash password

// Run database queries
mysql_query($mysqli,"INSERT INTO USERS (email, pword, type) VALUES ('{$email}', '{$phash}', '{$type}')");
if ($type == 0) {mysql_query($mysqli,"INSERT INTO JOB_SEEKERS (email, firstname, lastname) VALUES ('{$email}'
, '{$firstname}', '{$lastname}')");}
if ($type == 1) {mysql_query($mysqli,"INSERT INTO EMPLOYERS (email, comname) VALUES ('{$email}', '{$comname}'
)");}

mkdir('data/users/' . $email, 0777, true); // Create user dir
```

id	email	pword	type	shash
102	employer@gmail.com	\$2y\$10\$5BzF.74QvwI6ul2G.DdtAeD4MUoVyHanFrMNAT81tRRY9ZDJPO6om	1	2f877f553b568256a72b
103	a@gmail.com	\$2y\$10\$aLQ7PzOYKjTPAHzhwH13JOFTxAteDFYeZkOp6nmYwyhGBExz1F6j6	0	3d6b6cd897be7437c74d

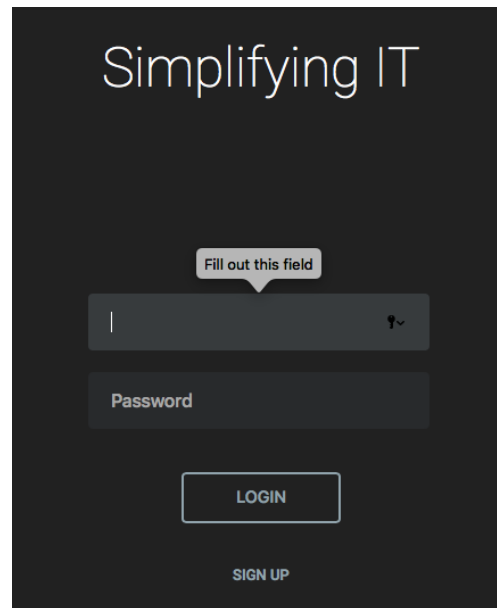
id	email	comname	phone	website	address	fax
24	employer@gmail.com	Company	NULL	NULL	NULL	NULL

id	email	firstname	lastname	gender	citizen_of	country	city	phone	website	about
26	a@gmail.com	Alex	P	0	NULL	NULL	NULL	NULL	NULL	NULL
27	b@gmail.com	Bob	Ross	0	NULL	NULL	NULL	NULL	NULL	NULL

⁴ php.net/manual/function.password-hash.php

⁵ php.net/manual/function.mkdir.php

2.2 Login



Similar to `signup.php` (2.1), `index.php` hook will catch a submitted form. The function `login()` verifies the user's submitted password against the hash found in the database using the `password_verify()` function⁶.

```
function login($email, $password) {
    $mysqli = new mysqli("127.0.0.1", "anon", "123", "SIT");

    // SQL injection mitigation
    $email = mysqli_real_escape_string($mysqli, $email);

    // Grab password hash from db
    $pass = mysqli_query($mysqli, "SELECT pword FROM USERS WHERE email='{ $email }' limit 1");
    $result = mysqli_fetch_array($pass);

    if ($pass->num_rows > 0) { // Check if account exists
        $phash = $result['pword']; // Verify password

        if (password_verify($password, $phash)) {
            initSession($email, $phash);
        }

        // Output error if password is wrong
        else {header("Location: index.php?msg=failed"); die();}
    }

    // Output error if email doesn't exist
    else {header("Location: index.php?msg=nomail"); die();}
}
```

⁶ php.net/manual/function.password-verify.php



Next, `initSession()` initializes the `$_SESSION` variables. The session hash is loaded from the database using the `SELECT` query ("limit 1" parameter reduces server workload when a single result is expected). If it is not found, it will be generated using the `makeShash()` function (2.3). Appropriate tables are selected based on the user's account type.

```
function initSession($email, $phash) {
    $mysqli = new mysqli("127.0.0.1", "anon", "123", "SIT");

    $_SESSION["email"] = $email;    // Set session variables
    $_SESSION["password"] = $phash;

    $work1 = mysqli_query($mysqli, "SELECT shash, type FROM USERS WHERE email='{ $email }' limit 1");
    $result1 = mysqli_fetch_array($work1);

    if ($result1['shash'] > 0) {$_SESSION["shash"] = $result1['shash'];}    // Assign session hash
    else { // Create new hash if null
        $_SESSION["shash"] = makeShash();
        mysqli_query($mysqli, "UPDATE USERS SET shash='{ $_SESSION["shash"] }' WHERE email='{ $email }'");
    }

    $_SESSION["type"] = $result1['type'];    // Init based on acc type
}
```

```
if ($_SESSION["type"] == '0') {
    $work3 = mysqli_query($mysqli, "SELECT firstname FROM JOB_SEEKERS WHERE email='{ $email }' limit 1");
    $result3 = mysqli_fetch_array($work3);

    $_SESSION["name"] = $result3['firstname'];
}

if ($_SESSION["type"] == '1') {
    $work4 = mysqli_query($mysqli, "SELECT comname FROM EMPLOYERS WHERE email='{ $email }' limit 1");
    $result4 = mysqli_fetch_array($work4);

    $_SESSION["name"] = $result4['comname'];
}
}
```


2.3 makeShash()

shash
2f877f553b568256a72b
3d6b6cd897be7437c74d
960b48ddf046647c7d4c
86df023fba5d68695008

`random_bytes()` function is called inside a `bin2hex()` function⁷, returning a hexadecimal string of length 10 and of 16^{10} or 1,099,511,627,776 possible combinations, enough for any amount of traffic. The function is executed recursively until a unique hash is found.

```
function makeShash() {
    $mysqli = new mysqli("127.0.0.1", "anon", "123", "SIT");

    $shash = bin2hex(random_bytes(10));

    // Check if hash is unique
    $work = mysqli_query($mysqli, "SELECT shash FROM USERS WHERE shash='$shash' limit 1");

    if ($work->num_rows > 0) {makeShash();} // If not unique, recalculate
    else {return $shash;}
}
```

2.4 Security checks

`Session.php` (shown below) is included on all post-login pages and validates session integrity by comparing session tokens against their values in the database. If validation fails, user is logged out.

```
<?php
session_start(); include 'functions.php';

// Perform session validity checks
$mysqli = new mysqli("127.0.0.1", "anon", "123", "SIT");
$pass = mysqli_query($mysqli, "SELECT pword, shash FROM USERS WHERE email='{$_SESSION["email"]}' limit 1");
$result = mysqli_fetch_array($pass);

// Logout if...
// ... password changed during session
if ($_SESSION["password"] != $result['pword']) {logout();}

// ... session token changed
if ($_SESSION["shash"] != $result['shash']) {logout();}

// ... no login found
if (is_null($_SESSION["email"])) {logout();}
?>
```

⁷ php.net/manual/function.random-bytes.php

Other checks are present on pages that are meant for a specific account type, such as `viewjob_employer.php` and `viewjob_seeker.php`...

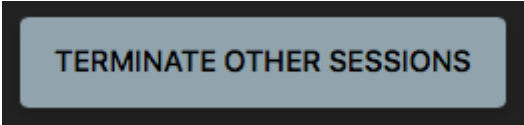
```
if ($_SESSION["type"] != '1') {redir("viewjob_seeker.php?id=".$_GET["id"]);}
```

```
if ($_SESSION["type"] == '1') {redir("viewjob_employer.php?id=".$_GET["id"]);}
```

... and explicitly using email session token in a query to make sure a job belongs to employer before letting them edit it.

```
$pass = mysqli_query($mysqli,"SELECT * FROM JOBS WHERE id='{$_GET["id"]}'  
AND company_email='{$_SESSION["email"]}' ORDER BY id DESC limit 1");
```

2.5 “Terminate other sessions” feature

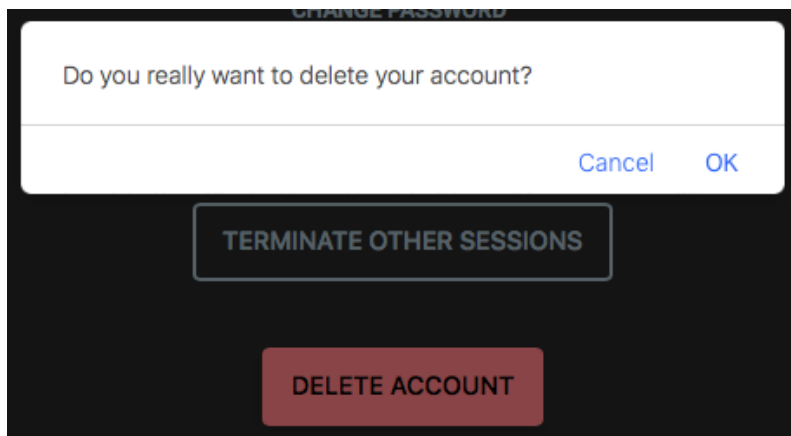


TERMINATE OTHER SESSIONS

When called, the `logoutOthers()` function generates a new hash for the current session using the `makeShash()` function (2.3) and pushes it to the `USERS` table. Due to `session.php` security checks (2.4), all sessions using an older hash will automatically be logged out.

```
function logoutOthers() {  
    $mysqli = new mysqli("127.0.0.1", "anon", "123", "SIT");  
    $_SESSION["shash"] = makeShash();  
    mysqli_query($mysqli,"UPDATE USERS SET shash='{$_SESSION["shash"]}' WHERE email='{$_SESSION["email"]}';");  
}
```

2.6 Account deletion



CHANGE PASSWORD

Do you really want to delete your account?

Cancel OK

TERMINATE OTHER SESSIONS

DELETE ACCOUNT

After the user proceeds with the JS `alert()` method message, `deleteAcc()` function is called, which runs a series of DELETE queries that erase any database entries related to the user's email.

```
function deleteAcc($email) {
    $mysqli = new mysqli("127.0.0.1", "anon", "123", "SIT");

    removeDirectory("data/users/" . $email);
    mysqli_query($mysqli, "DELETE FROM USERS WHERE email = '{$email}'");

    mysqli_query($mysqli, "DELETE FROM JOB_SEEKERS WHERE email = '{$email}'");
    mysqli_query($mysqli, "DELETE FROM JOB_APPLICATIONS WHERE applicant_email = '{$email}'");

    mysqli_query($mysqli, "DELETE FROM EMPLOYERS WHERE email = '{$email}'");
    mysqli_query($mysqli, "DELETE FROM JOBS WHERE company_email = '{$email}'");
    mysqli_query($mysqli, "DELETE FROM JOB_APPLICATIONS WHERE company_email = '{$email}'");
}
```

User's directory on server (created in section 2.1) is deleted using the `removeDirectory()` function, which is necessary because PHP does not allow deletion of non-empty directories, and was suggested by Christopher Smith on Stackoverflow⁸. The function takes the directory path as an argument and uses the `glob()` function⁹ to get an array of all entities under the directory. It uses the `foreach()`¹⁰ construct to traverse the array and the `is_dir()` function to determine if the traversed entity is a directory in itself. If true, it will execute recursively for that directory, and if false it will delete the traversed file using `unlink()`. After `foreach()` is finished executing, `rmdir()`¹¹ will remove the now empty directory.

```
function removeDirectory($path) {
    $files = glob($path . '/*');
    foreach ($files as $file) {
        is_dir($file) ? removeDirectory($file) : unlink($file);
    }
    rmdir($path);
}
```

3.0 FEATURES

⁸ stackoverflow.com/a/49444840

⁹ php.net/manual/function.glob.php

¹⁰ php.net/manual/control-structures.foreach.php

¹¹ php.net/manual/function.rmdir.php

3.1 Job browser and filters

The image shows a mobile application interface for a job browser. On the left is a dark-themed sidebar with filter options, and on the right is a light-themed main content area.

Filters Sidebar (Left):

- NAME CONTAINS...**
All
- CATEGORY**
Any
- COMPANY**
Any
- COUNTRY**
Any / Remote employment
- MAXIMUM REQUIRED YEARS OF EXPERIENCE**
Any

A circular arrow icon is at the bottom of the sidebar.

Main Content Area (Right):

- Header: "Welcome, Alex" and "Here's what we found for you"
- Job Listing Card:
 - Logo: "i6" inside a circular seal with text "COLEGIO DEL MUNDO • WORLD SCHOOL • ÉCOLE DU MONDE"
 - Title: "Sample Job at Company"
 - Text: "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Curabitur dictum ut sapien in scelerisque. Integer quis tempus lacus, at feugiat velit. Donec a fermentum metus. Maecenas quis lorem ac massa finibus euismod. Nam volutpat tellus sit amet tempor. In egestas lectus arcu, sed l congue in. Nam felis leo, imperdiet id dignissim a,"
 - More button: "MORE"

At the bottom of the screen is a navigation bar with three icons: a hamburger menu, a magnifying glass, and a user profile icon.

Filters form is submitted using HTTP GET method in order to let the user bookmark their search queries. Using the LIKE mysql comparator with wildcards around each filter string means that if a filter is null for a category, no filter is applied to it. This allows nested filters to behave exactly how the user would expect them to, using only one sql query. Jobs that are closed for applicants are always filtered out (using "WHERE is_open='1'").

```

if ($min_exp == '') {$min_exp = 99;} // User will filter exp by setting upper boundary so the default value
must be the maximum possible

// Main query
$pass = mysqli_query($mysqli,"SELECT id FROM JOBS WHERE is_open='1' AND name LIKE '%{$name}%' AND company_email
LIKE '%{$company}%' AND country LIKE '%{$country}%' AND city LIKE '%{$city}%' AND min_exp <= '{$min_exp}' AND
category LIKE '%{$category}%' ORDER BY id DESC");

```

Jobs that user has applied to are filtered out by querying them from “JOB_APPLICATIONS” table and outputting them into an array using a `mysqli_fetch_array()` function¹² as an argument for a *while* loop. This works as we are only selecting the `job_id` cell and the `mysqli_fetch_array()` function will iterate over the next row whenever it’s called, and saves a lot of code. The loop terminates when no rows are left to traverse.

```

// Filter out the jobs the user has already applied to
$pass2 = mysqli_query($mysqli,"SELECT job_id FROM JOB_APPLICATIONS WHERE applicant_email='{$_SESSION["email"]}'
ORDER BY job_id DESC");
$result2 = array();
while ($row = mysqli_fetch_array($pass2)) {$result2[] = $row[0];} // Create an array from fetched results
$pass3 = mysqli_query($mysqli,"SELECT comname, email FROM EMPLOYERS"); // Used in per-company filter (sidebar.php)

```

Jobs are outputted using a similar *while* loop. Results from main query are excluded if their ID is present in the abovementioned array (`in_array()` function¹³).

```

while ($result = mysqli_fetch_array($pass)) {
    // Don't show jobs the user has already applied to
    if (in_array($result["id"], $result2) == 0) {
        callCard($result['id']);
    }
}

```

`callCard()` holds an `include card.php` statement. `Job_id` is used to fetch additional information to display on the card.

```

$mysqli = new mysqli("127.0.0.1", "anon", "123", "SIT");
$pass = mysqli_query($mysqli,"SELECT * FROM JOBS JOIN EMPLOYERS ON company_email=email WHERE JOBS.id='{$job_id}'
ORDER BY JOBS.id DESC");
$result = mysqli_fetch_array($pass);

```

¹² php.net/manual/mysqli-result.fetch-array.php

¹³ php.net/manual/function.in-array.php

3.2 Job seeker dashboard

DASHBOARD		
JOB TITLE	EMPLOYER	STATUS
TEST @	Employer 2	Pending
TEST	Employer 2	Rejected
JOBBBB	Employer 2	Pending
JOB	Company	Accepted

Displays jobs the user applied to in a table with columns "job title", "employer" and "status". To fetch this information, a tri-table SQL join is used between JOB_APPLICATIONS, JOBS, and EMPLOYERS tables.

```
$pass1 = mysqli_query($mysqli,"SELECT * FROM JOB_APPLICATIONS JOIN JOBS ON JOB_APPLICATIONS.job_id=JOBS.id  
JOIN EMPLOYERS ON JOBS.company_email=EMPLOYERS.email WHERE applicant_email='{$_SESSION["email"]}' ORDER  
BY JOBS.id DESC");
```

A while loop outputs the data as table rows. Status is color coded for easier evaluation.

```
while ($result1 = mysqli_fetch_array($pass1)) { // Output as table rows  
  
    echo "<tr class='clickable-row' data-href='viewjob_seeker.php?id=".$result1['job_id']."'>";  
    echo "<td>". $result1['name']. "</td>";  
    echo "<td>". $result1['comname']. "</td>";  
    // Use appropriate color coding  
    if ($result1["app_status"] == "Accepted") {echo "<td><font color='#4CAF50'>Accepted</font></td>";}  
    else if ($result1["app_status"] == "Rejected") {echo "<td><font color='#E57373'>Rejected</font></td>";}  
    else {echo "<td>". $result1['app_status']. "</td>";}  
}
```

JQuery ready() method¹⁴ was used with the window.location property to make rows redirect on click.

```
jQuery(document).ready(function($) {  
    $(".clickable-row").click(function() {  
        window.location = $(this).data("href");  
    });  
});
```

¹⁴ https://www.w3schools.com/jquery/event_ready.asp

3.3 Employer dashboard

DASHBOARD		
JOB TITLE	STATUS	PENDING APPLICANTS
TEST @	Open	1
TEST	Open	0
JOB BBB	Open	1

Works similarly to the job seeker dashboard, but runs a second query inside the while loop to find the number of pending applications for each job, so users don't have to open each job to check for new applications.

```
// Loads hosted jobs
$pass = mysqli_query($mysqli,"SELECT * FROM JOBS WHERE company_email='{$_SESSION["email"]}' ORDER BY
id DESC");

while ($result = mysqli_fetch_array($pass)) {
    // Finds the number of pending applications
    $pass2 = mysqli_query($mysqli,"SELECT 0 FROM JOB_APPLICATIONS WHERE job_id='{$_result["id"]}' AND
app_status = 'Pending'");
    $rows2 = $pass2->num_rows;
```

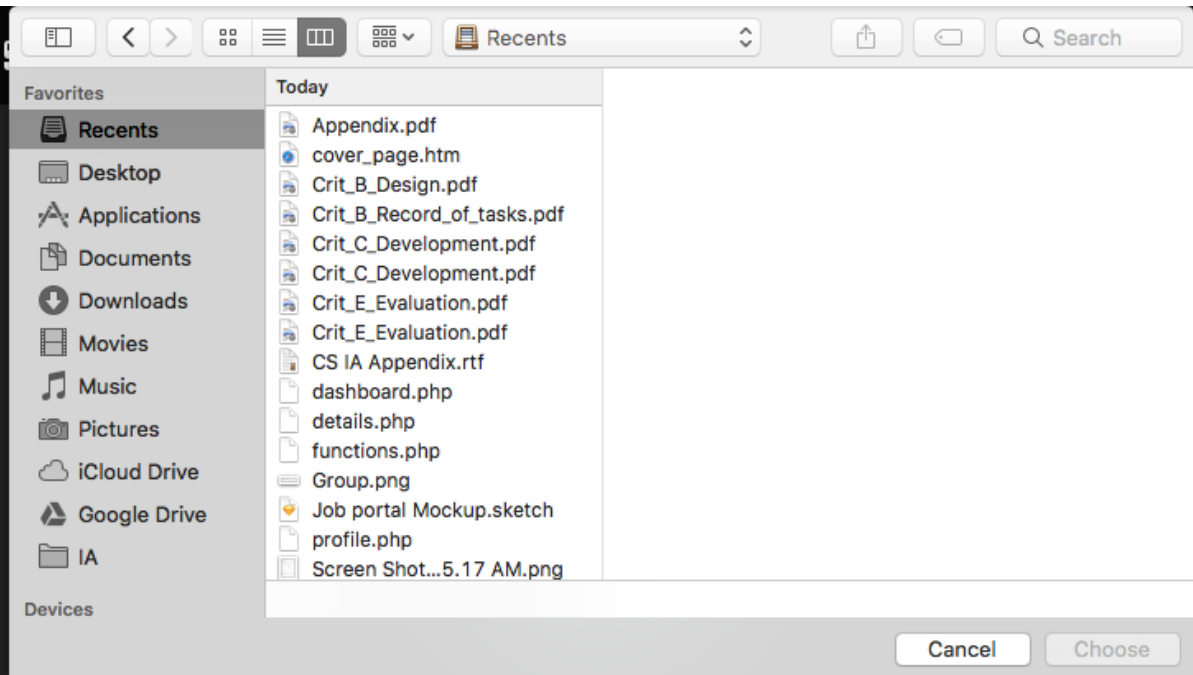
3.4 Upload handling

Currently, my *processUpload()* function handles profile picture uploads. It performs extension and size validation (allowed formats are stored in an easily modifiable array) and outputs errors to the user:

Wrong upload format

File size must be below 5MB

The file is renamed and moved to the users directory (based on session email token), created in section 2.1.



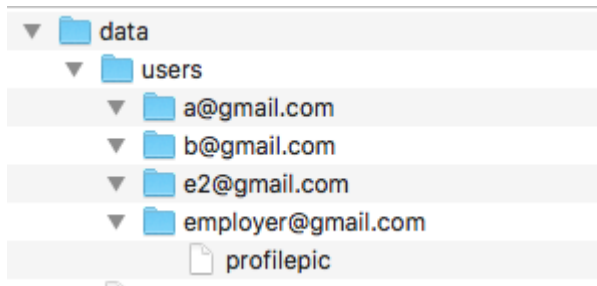
e2@gmail.com

DETAILS

```
function processUpload() {
    $allowed = array('png','jpg','jpeg'); // Allowed formats
    $filename = $_FILES['upload']['name']; // Fetches file
    $ext = pathinfo($filename, PATHINFO_EXTENSION); // Check format against $allowed
    if(!in_array($ext,$allowed) ) {echo "<div align='center' class='red'><br><br>Wrong upload
        format</div>";}

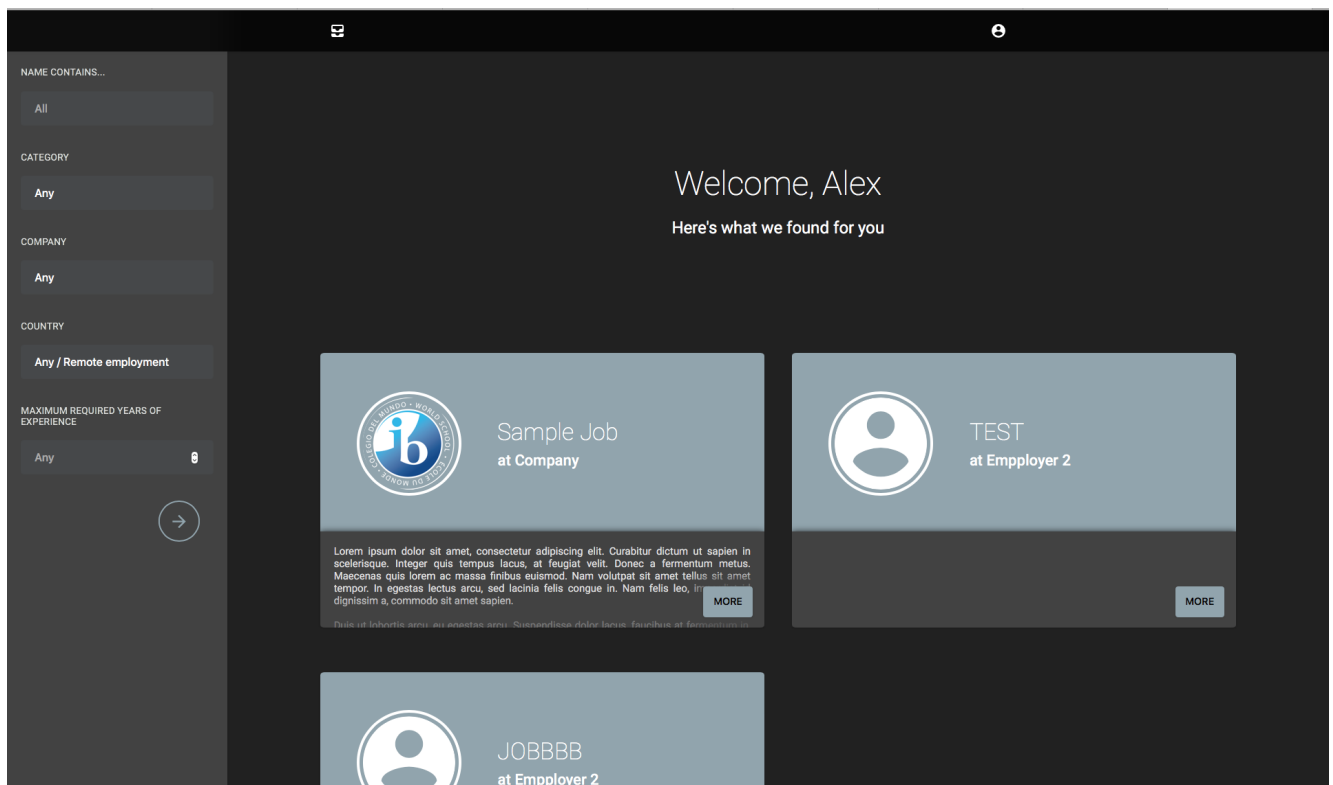
    // Limit files size (in bytes)
    else if ($_FILES['upload']['size'] > 5000000) {echo "<div align='center' class='red'><br><br>
        File size must be below 5MB</div>";}

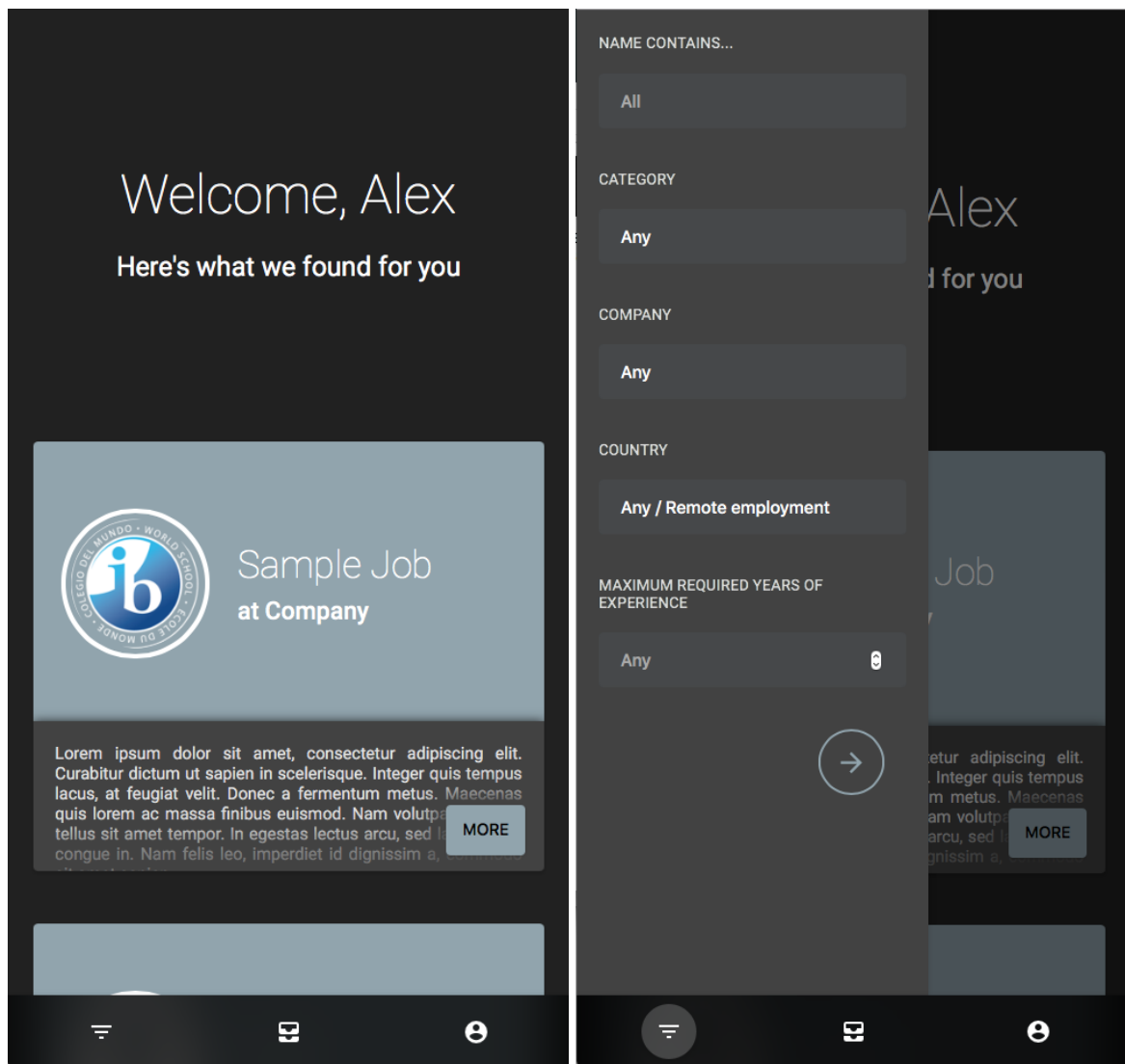
    // Move to user folder
    else {move_uploaded_file($_FILES['upload']['tmp_name'], "data/users/" . $_SESSION["email"] . "/"
        profilepic");}
}
```

4.0 GUI

4.1 Adaptive layouts





@media rules are used to dynamically adjust CSS properties (such as grid columns) to fit various screen sizes.

```
@media only screen and (max-width: 600px) {

  #menubar {
    height: 65px;
    bottom: 0;
  }

  .inmenubar {
    font-size: 1em;
    height: 100%;
    width: 45px;
  }

  body{padding-bottom: 65px}

  #sidepanel {display: none; height: calc(100% - 65px);}

  div.list {width: 100%; left: 0; position: absolute; border-radius: 0px; padding-bottom: 100px;}
  #dashgrid {display: block;}
  .cardlogo {width: 12vh !important; height: 12vh !important;}
}
```

Interactive cards use nested CSS Grid layouts and a mix of relative and absolute positioning to achieve something that saves space, works well on touchscreens, and is intuitive for the user. Frequent use of overlapping elements demanded a careful use of z-index properties throughout the website.



```
#cardpic {
  height: 66%;
  width: 100%;
  background: #90A4AE;
  position: absolute;
  display: grid;
  grid-template-columns: 30% auto;
  grid-column-gap: 5%;
  padding-left: 5%;
  grid-template-rows: 100%;
  align-items: center;
  font-size: 2.1em;
  font-weight: 100;
}

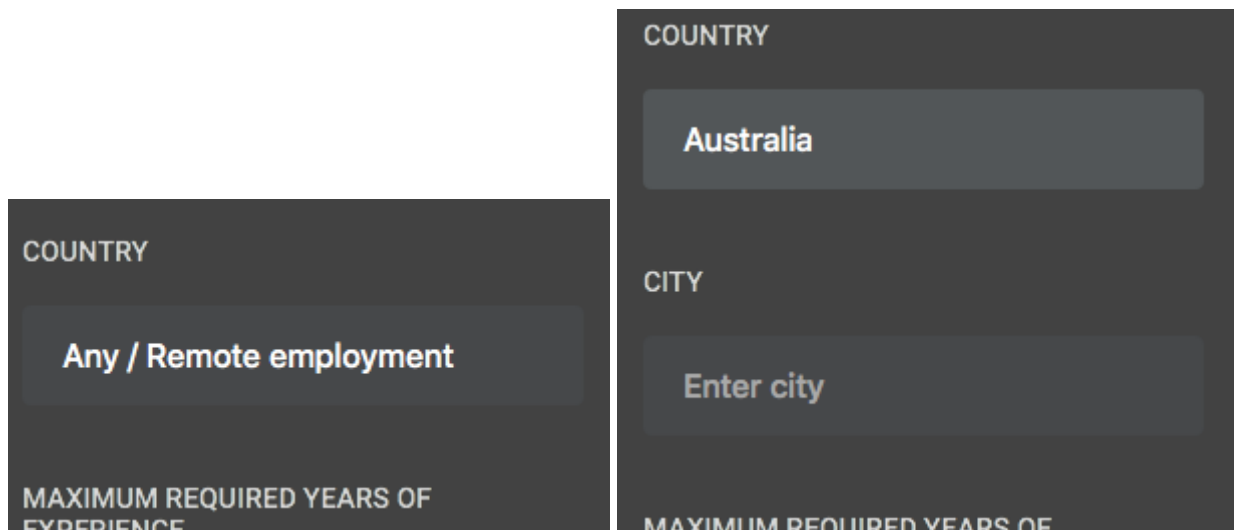
#cardcontent {
  bottom: 0;
  width: 100%;
  height: 35%;
  background-color: #424242;
  position: absolute;
  box-shadow: 0 1px 10px rgba(0,0,0,0.9);
  box-sizing: border-box;
  padding: 20px;
  transition: all 0.25s cubic-bezier(0.175, 0.885, 0.32, 1.1);
  color: #E0E0E0;
  overflow: hidden;
}

#cardcontent:after {width: 100%; content: ""; position: absolute; box-shadow: 0px 0px 30px 25px #424242; bottom: 0;}

#cardcontent:hover {
  height: 95%;
  color: white;
  border-radius: 1%;
}

#card {
  box-shadow: 0 1px 3px rgba(0,0,0,0.2);
  border-radius: 1%;
  width: 100%; height: 40vh; max-height: 400px; min-height: 240px;
  position: relative;
  display: grid;
  grid-template-rows: auto auto;
  overflow: hidden;
  margin-bottom: 10%;
}
```

3.2 Dynamic forms



The image shows two states of a web form. On the left, the 'COUNTRY' dropdown is set to 'Any / Remote employment', and the 'CITY' field is hidden. On the right, the 'COUNTRY' dropdown is set to 'Australia', and the 'CITY' field is visible with the placeholder text 'Enter city'.

Form parameters can be shown or hidden dynamically based on a parent element. Example: “city” and “address” filters are only shown if “country” filter is set. When changed, parent element calls the `yesnoCheck()` function¹⁵. Child elements are assigned a “dynamic” ID.

```
<h3>COUNTRY </h3> <br><br>
<select name="country" class="input" onchange="yesnoCheck(this);">
  <option value="">Any / Remote employment</option>
  <?php include 'countries.php' ?>
</select>
<br>
<br>
<br>

<div id="dynamic" hidden>
<h3>CITY </h3> <br><br>
<input type="text" placeholder="Enter city" name="city" class="input" value="<?php echo $city ?>">
<br>
<br>
<br>
</div>
```

Elements are toggled using JQuery `hide()` and `show()` methods.

```
function yesnoCheck(that) {
    if (that.value == "1") {
        $('#dynamic1').hide();
        $('#dynamic1 :input').prop('required',null);
        $('#dynamic2').show();
        $('#dynamic2 :input').prop('required',1);
    }
    else {
        $('#dynamic2').hide();
        $('#dynamic2 :input').prop('required',null);
        $('#dynamic1').show();
        $('#dynamic1 :input').prop('required',1);
    }
}
```

¹⁵ Adapted from <https://stackoverflow.com/a/29321711>

Sources cited

“foreach.” *Php*, php.net/manual/control-structures.foreach.php.

“glob.” *Php*, php.net/manual/function.glob.php.

“in_array.” *Php*, php.net/manual/en/function.in-array.php.

“isset.” *Php*, php.net/manual/en/function.isset.php.

“jQuery Ready() Method.” *W3Schools*, w3schools.com/jquery/event_ready.asp.

Kemarsky, Gleb “Show Input Field Only If a Specific Option Is Selected.” *Stack Overflow*, stackoverflow.com/a/29321711.

“mkdir.” *Php*, php.net/manual/en/function.mkdir.php.

“mysql_fetch_array.” *Php*, php.net/manual/en/function.mysql-fetch-array.php.

“Mysqli::real_escape_string.” *Php*, php.net/manual/en/mysqli.real-escape-string.php.

“password_hash.” *Php*, php.net/manual/en/function.password-hash.php.

“password_verify” *Php*, php.net/function.password-verify.

“random_bytes.” *Php*, php.net/manual/en/function.random-bytes.php.

“rmdir” *Php*, php.net/manual/function.rmdir.php.

“session_start.” *Php*, php.net/manual/en/function.session-start.php.

Smith, Christopher “Delete directory with files in it?” *Stack Overflow*, stackoverflow.com/a/49444840.