```
  // Redirect the user to the home page
  header("Location: http://www.asite.com/lab.php");
  }
?>
```

Putting the pieces together requires an `if` statement to determine if the user has entered the user ID and password. If the user has not done so, the HTML code to request them is displayed. If the information has been entered (and is valid using the HTML5 pattern expressions shown) the `else` portion of the statement will execute (storing the values in the session variables and calling the `lab.php` program). This provides you with the basic shell of accepting the user ID and password, verifying they exist, and calling the program in the interface tier if they do exist. Of course, you need to authenticate the user ID and password before calling the program.

> *Programming note—The server variables `PHP_AUTH_USER` and `PHP_AUTH_PW` can be used for user ID and password validation, instead of using session variables.*
>
> ```
> header('WWW-Authenticate: Basic realm="ABC Canine"');
>     header('HTTP/1.0 401 Unauthorized');
> ```
>
> *Unauthorized header messages can be created if the user has not entered a user ID/password or a valid user ID/password. This will automatically cause the system to request the user enter a user ID/password. This technique is pretty straightforward. However, there have been some reports, in the past, of browsers not functioning properly with this technique. Besides, creating your own technique allows you to design the login screen with the same style as the rest of your web site.*
>
> *For more information, visit:*
> *http://php.net/manual/en/features.http-auth.php*

```
$valid_useridpasswords = array ("sjohnson" => "N3working");
$valid_userids = array_keys($valid_useridpasswords);
$userid = $_SESSION['username'];
$password = $_SESSION['password'];
$valid = (in_array($userid, $valid_userids)) && ($password == $valid_useridpasswords[$userid]);
If($valid) { header("Location: http://www.asite.com/lab.php");}
```

There are several ways you can authenticate user IDs and passwords. If you are creating a system that does not require user IDs and passwords to change, you could use arrays. In the previous example the `$valid_useridpasswords` associate array contains the combination of valid user IDs and passwords. The PHP method `array_keys` places all keys (in this example the user IDs) into a separate array (`$valid_userids`). After the session variables have been placed in `$userid` and `$password`, the PHP `in_array` method is used to determine if the correct combination of user ID and password exists. `in_array` determines if the user ID exists in the array. Then the user ID is used as the subscript to pull the password from the `valid_useridpasswords` array and compare it to the value in `$password`. If the user ID exists and the passwords are the same, then everything is valid. `$valid` will contain `TRUE`. If either (or both) are not valid, `$valid` will contain `FALSE`. If `$valid` is `TRUE`, the application redirects to the `lab.php` program.

Technically properties in a session are secured from any access outside the session. However, there have been reported instances, in the past, of hacker programs breaking this security and accessing session information. If the user ID and password, in this example, are stored in session variables and passed across the Internet to another program, hackers might gain access to the information.