# Lab 1

You will submit a clickable HTML link to your site hosted on the opentech server (see Fundamentals resources if you need to review publishing your website, the only change is the folder on the opentech server is /var/www/html/webd3201/userid versus /var/www/html/webd2201/userid).

Your site will implement responsive Bootstrap layouts using the provided template (found in the assignment in DC Connect). It will consist of three (3) pages. The first, available at /index.php, will be your websites home page. The second will be named dashboard.php, this will be the user’s main page (and where they are directed to when they log in). dashboard.php will only be available to signed in users. The content of this dashboard will only be a placeholder for this assignment, other than a welcome message. The third page, available at /sign-in.php, will be your sign-in page. It will only be available to users who have not yet signed in.

You are to take the the provided template and break it into header.php and footer.php.

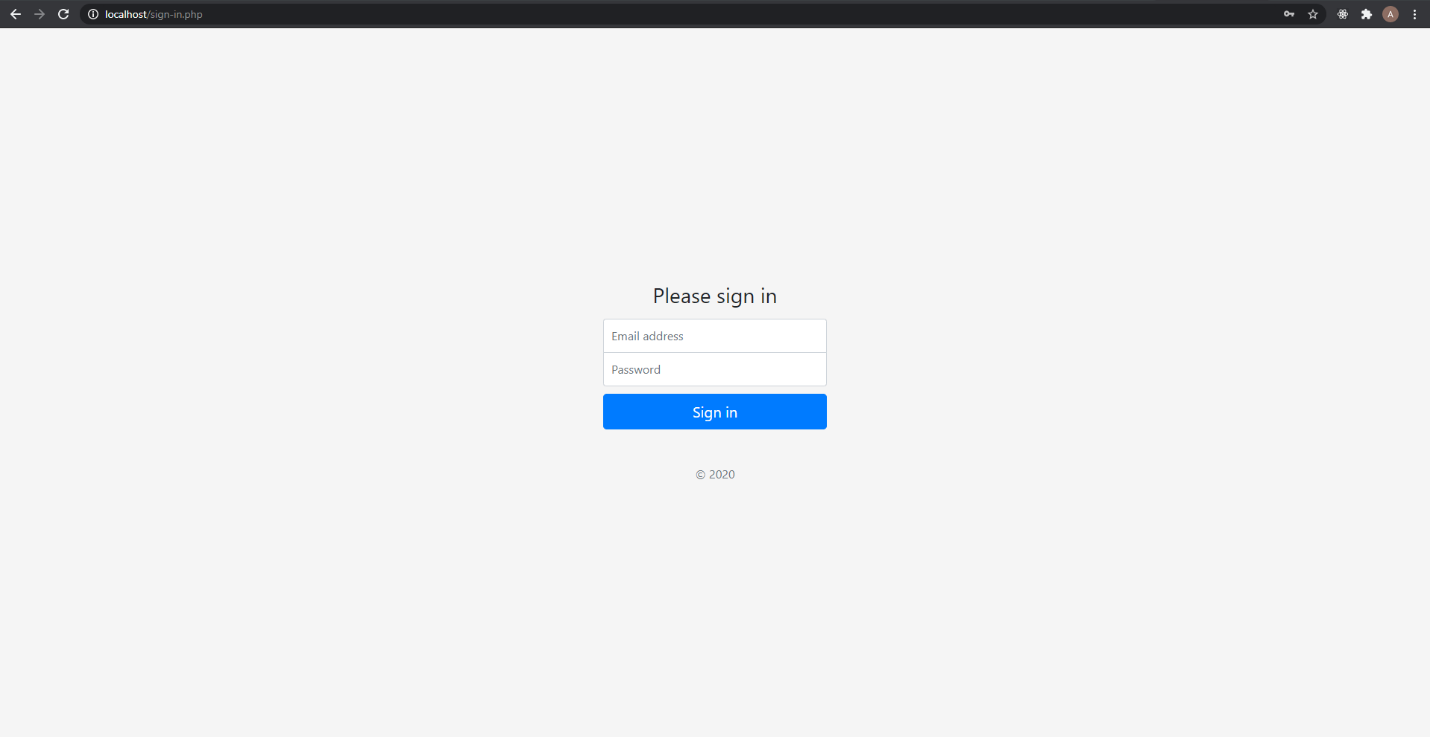
header.php should have /\* PHP comments \*/ with your name, date and course code, and dynamic <!-- HTML Comments -->. This file should also start a session (session\_start()), start an output buffer (ob\_start()), and require the files constants.php, db.php, and functions.php. The footer should contain the bottom of your web page. The files index.php, sign-in.php and dashboard.php should all implement your header.php and footer.php files.

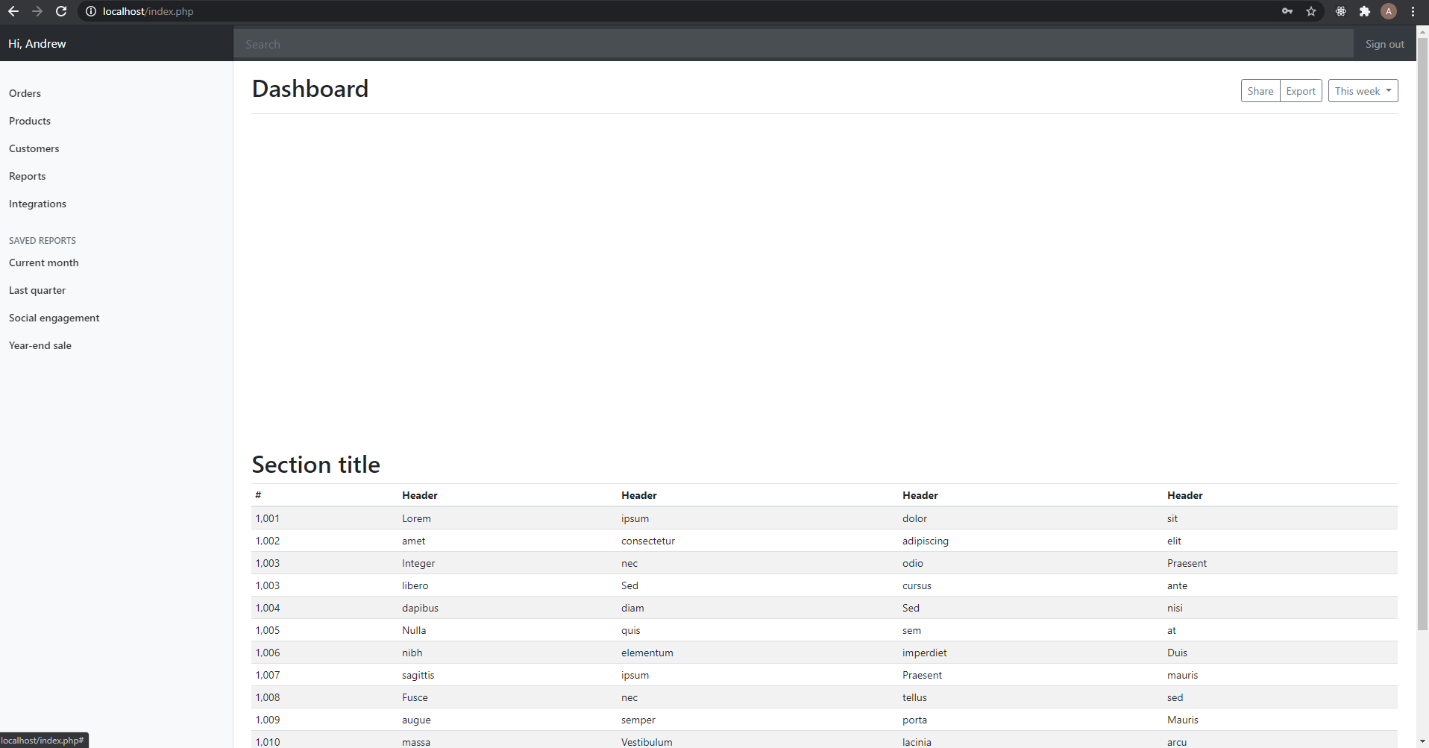
If a signed in user navigates to /sign-in.php, your site will redirect them to /dashboard.php. Similarly, if a non-signed in user navigates to /dashboard.php, your site will redirect them to /sign-in.php.

There will be a “Sign out” button on dashboard.php and in the dynamic nav bar (see below) that will destroy the session and direct the user to /sign-in.php (where an appropriate message will be displayed).

On the server side, every page will include the same header.php and footer.php, while implementing custom logic specific to that page in this index.php, sign-in.php and dashboard scripts. In your header, you will need to include the Bootstrap css library, as well as logic to include CSS specific to the script you are running.

For your reference:





## Initial Database Setup

Your site will need a database to store the users who sign in. For this course you will be using PostGreSQL.

You will create a script called db.php. This script contains a function named db\_connect() that returns a connection to your database using the PHP function pg\_connect() and creates your prepared statements using the PHP function pg\_prepare(). You will include this db.php script in header.php. There will be no other way to access the database except through this script and the prepared statements it creates.

You will implement the following prepared statements:

* + user\_select
  + user\_update\_login\_time

user\_select takes one parameter, an id, and retrieves that user.

user\_update\_login\_time takes two arguments, id and the current time, and updates the appropriate user’s record so the last login timestamp is set to the current timestamp.

You will create an SQL file called users.sql (including comments with your name, the date and the course code). This file will be run manually on the opentech server command line using syntax like:

psql -d DATABASE\_NAME -f users.sql

You will run this command before you submit lab, to ensure there is data to test with. This script will drop all the existing databases (if necessary), create them again, and insert some sample data. **This script needs to be available on opentech in a *sql* sub-directory in your working folder (i.e. */var/www/html/webd3201/userid*).** In a real-world scenario, you would not make database information available like this, but for this assignment you will (for assessment purposes).

The users.sql script will create a table called “users”. Using appropriate data types, you will need to keep track of the following:

* + Id (primary key)
  + Email (unique in the database)
  + First name
  + Last name
  + Password Hash (see security lecture)
  + Created time (a timestamp)
  + Last time user logged in (a timestamp)
  + Phone extension
  + User type (should be character data that can store up to 2-characters)

For this assignment, all users will be “admins”. You will need to store that as a single character (see notes on constants.php). It is not appropriate to use “email” as the primary key. You will create an appropriate field for that.

Regarding passwords, it is unacceptable to store them in plain text. You will hash the password before storing it, using the bcrypt algorithm.

Your script will create at least 3 users. Note that it is not necessary to pre-fill the “last login time” field on creation, since the user may never have yet signed-in.

## State Maintenance

Your session will keep track of whether the user is logged in, and if so, his or her details. It is suggested you put “session\_start()” in your header so that it gets called every time a page is displayed.

Additionally, your session will contain information about banners to be displayed. The two required banners for this assignment are error messages on sign-in, and a welcome message for the user on the dashboard, telling him or her the last login time. Banners must only display once. For instance, if I navigate back to /index.php after seeing the banner, the banner should not be displayed again (i.e. remove the message from the session).

## Site Constants (constants.php)

You will create a file called constants.php. This file will be included in the header.php file. This file will define all site-wide constants using appropriate define() syntax.

You will need to define a single character that will refer to the “Administrator” user type. It is suggested you use the letter “a”.

You will also need to define the connection parameters you will use to connect to the database. Tip: have one file for local development and one file for opentech development.

## db.php

You will create a file called db.php. This file will be the only place you will store your database functions (i.e. db\_connect() and all of your prepared statements). You will have the following functions:

* + db\_connect()
  + user\_select()
  + user\_authenticate()

db\_connect() should not take any arguments, and should return a PostgreSQL connection resource.

user\_select() takes one argument, id, and returns an associative array with that user’s information, or false if that user does not exist.

user\_authenticate() takes two arguments, id and password, and returns an associative array with that user’s information, or false if that user does not exist. If a record is retrieved (i.e. the user has authenticated) the last login time should be updated to the current timestamp.

NOTE: user\_authenticate(), when a record is returned based on the id provided, is to use password\_verify() to check if the password entered matches the bcrypt hash of the password for the user. If the user is retrieved successful, the function is to update the user’s last\_login time at this point (using the current timestamp). Regarding the banner shown on successful sign-in, we want to see the last\_login time, not the current time.

## Non-database functions (functions.php)

All functions, that are not database related, should be stored in the file named functions.php.

## Dynamic Navigation

Your header.php page should have a dynamic nav bar.

This means there should be link to the Home page (index.php) always. If there is not a user logged in the dynamic nav bar should have a link to the sign-in.php page. If a user is logged in, there should be a link to the dashboard.php page displayed, and the link to login (sign-in.php) should be changed to logout (logout.php)

## Secure Login Functionality

You will create a file called sign-in.php. It should include a self-referring form with two inputs, one for an id (a text input) and a password (a password input).

When the form is submitted to in POST mode, it should retrieve the inputted data (removing any leading and trailing whitespace), it should verify both fields were entered (display an error otherwise), and if so it should call the user\_authenticate() function, passing the inputted data.

If a record is not returned, the user should remain on the sing-in.php page, with a message stating the user did not authenticate correctly. If a record is retrieved, the user data should be placed on the session, a successful login message should be placed on the session, and the user should be redirected to the dashboard.php page (where the success message is displayed and then removed from the session).

## Logout Functionality

The logout.php file should perform the following: unset the session; destroy the session; restart the session; place a message stating the user has successfully logged out onto the session; and redirect the user to the login.php page, where the message from the session should be displayed (and then removed from the session.

## Activity Logging

All sign-in activity will be logged to a text fie (DATE\_log.txt), where DATE is the current date in YYYYmmdd form. A log entry will look like:

Sign in success at <time>. User <email> sign in.

For this assignment, you will log sign-in, sign-out and failed sign-in activity.