Web Development Server-Side

Assessment Brief 2025
Due before: 05 May @ 09.00am

Repeating students see alternative brief

1
Table of Contents

2 Project OVERVIEW

The assignment for this module is issued in conjunction with the modules:

- COMP H2027 Software Engineering & Testing, and
- COMP H2034 IT Business Management

This is a joint assignment between the three modules, where some deliverables will satisfy requirements for all 3 modules and others will be module specific. Note that the ultimate idea is to reduce your workload and focus on the project process, as well as some technical competencies.

Students are to create an e-commerce website (online store) to the specifications outlined in the sections below. In this module (Web Development Server-Side), you will be graded on the technical competencies related to the core features listed in Section 3 below. The other modules (Software Engineering & Testing, and IT Business Management) will focus on other aspects of this project, e.g., requirements analysis, design process, software testing, project management skills, documentation and so on… What is required for each module should be clearly listed in each project brief. I recommend that you take the time to put all the requirements together and consider it as one big project. That way you have a clear idea what you are dealing with.

NOTE 1: Remember, the learning outcomes for each module are what we are assessing. Given this, it is not appropriate for one student to complete the coding and another to complete the documentation, for example. All students in a group must showcase that they have knowledge of the learning outcomes of this module. That means that each student must take ownership of some coding aspect of this project.

NOTE 2: Students are not permitted to use any 3rd party code in this assignment. All code must come from the module learning materials of the modules under assessment.

3 Specific elements of the assignment

**See the grading rubric for more detail*

3.1 Summary of Core Features:

1. 3-tier architecture using MVC design pattern
2. HTML5 & CSS used for presentation (can use simple bootstrap templates)
3. PHP used for all logic and data processing
4. PDO used for all CRUD functions
5. MySQL DB used for data storage
6. Sessions used for persistent data, where required
7. User authentication
8. All user input correctly validated and sanitised

3.2 Core Features Explained

Students should include all the following:

- MVC design which separates files and functionality, at a minimum:
    - Public facing files in their own directory
    - secure DB server credentials
    - DB connection and separation of SQL queries
- Example of all CRUD functions with PDO (Create, Read, Update, Delete). *N.B. MySQLi or other APIs will not be accepted.*
- Appropriate web pages and templates to input and/or display data. If using Bootstrap, use only CSS and HTML templates. Be aware that no marks are

awarded for any logic that comes with templates (e.g., JavaScript shopping carts etc…). All logic must be written in PHP and by the student submitting.
- All user input correctly formatted and sanitised at the client/server side.
- Use of DB to store all stock available for sale on the store.
- Use of session for login and shopping cart facility. Use session destroy at end of a session. Shopping cart contents can be stored, after a session in the DB.
- Allow user to register a user account with the website, where user data is stored in a DB table to be checked at login.
- Well considered design and navigation structure.
- Use of templating.

3.3 For Improved Grades

- Make good use of custom PHP classes and functions.
- Use DRY code (Don't Repeat Yourself).
- Consider version control and testing.
- Demonstrate a solid understanding of the module learning outcomes and show more examples of the core features listed above.

- *Note: improved grades are gained from showcasing mastery of the lessons covered during the semester… not by adding fancy code you found on the internet.*

3.4 The User Experience

Some aspects of your website should be visible to all users, while others are only visible to those who have logged in. For example, users who have not logged in to the website might be able to view general information pages such as index.php, or the registration page. While only users who have logged in successfully will be able to see their account details.

CRUD functionality should be available for users to register as a user, if not already registered, and for them to search the products DB. In such a case, 2 separate tables are required in the DB.

The text **content** of your web pages should all be prose you have written yourself (or lorem epsum.)

4 Submission guidelines

4.1 Submission Overview

Students are required to carry out an on-campus demonstration of their project in week 13. Schedule to follow closer to the time.

Each student group will also submit all website and DB files for the project, to Brightspace, via a GitHub Repository. *Note, this is <u>in addition</u> to any submission arrangements for COMP H2027 Software Engineering & Testing.* One submission per group is fine.

4.2 Project Demonstration

The project demonstration is the only opportunity for students to have their work graded (unless evidence of a valid reason for absence is provided). Students who do not demonstrate will receive 0% for their assignment.

Students will be required to demonstrate different aspects of their project for COMP H2027 Software Engineering & Testing, and for this module. For this module, students are expected to demonstrate the website functionality and some of its source code, while arguing how well it has met all the project criteria listed above.

In demonstrating your project, consider:

- The grading rubric in Appendix 1: Grading rubric
- Provide a walk-through of the website's functionality, in context of your code design.
- An overview of software architecture design considerations, specifically the **MVC** pattern, and how it is implemented.

In all cases refer to your code via the project assessment criteria, so that your demo/defence forms a strong argument for how your project meets the various requirements of the brief.

4.3 Submission of Project-Files to Brightspace

All Project-Files must be in your GitHub Repo
Project-files include all files, e.g.,

- Website files
- SQL files (see Appendix 2: Submitting your DB for instructions)
- Readme file to include:
  - References for any 3rd party assets or tutorials used
    - *Though this is not permitted, if you did use 3rd party materials, you must reference them correctly or your project will be considered plagiarised*
  - Contribution statement
- Any other documentation you consider relevant.
- Version Control to be demonstrated via multiple consistent Git commits.

5 Penalties

1. Not submitting a project component as requested may result in a grade of 0% that component.
2. Non-attendance at the scheduled demo session will result in a grade of 0% for each student not in attendance.
3. Late submission will result in a grade of 0%
4. Unreadable files will achieve a grade of 0%
5. EMAIL submissions will NOT be accepted
6. A project that is simply the result of following a tutorial will achieve a grade of 0%
    a. 0% will be awarded for any project code that is not based on tutorials used in this module this semester. Therefore, it is best to avoid using any code 3rd party tutorials, or any 3rd party sources.
    b. Any projects that do use 3rd party code/tutorials, and do not reference them will be considered plagiarised (See Section 7).
7. Projects using the MySQLi API, or other DB APIs, other than PDO, will be marked down on ALL PDO and DB components.

6 Academic Honesty

In general: For every piece of work you submit to the university, your documentation must make it very clear which parts are your own creations; the work of others; and your adaptation of other's work, and what your adaptations were. This includes typing out code from a tutorial. Just because you typed it, doesn't make it your code. *In such a case, this is akin to retyping someone else's essay. You may have typed it, but that doesn't make it your work.*

Work submitted without full and unambiguous acknowledgements is plagiarism. Plagiarism and academic dishonesty can lead to failure of the module and other penalties outlined in the university's rules and regulations. For any project or coursework, you should discuss how to best declare the use of work from other sources with your lecturers.

All work submitted should be your own, except for:

- **(1) media files, such as images, fonts, and templates (for HTML/CSS only)**
    - **a. You must declare these in a sources document**
- **(2) you may use and modify any example codes from the lectures/labs without any need to reference such sources**

By submitting your project for assessment you agree to the following:

"The material contained in this assignment is my own original work, except where work is clearly identified and duly acknowledged. No aspect of this assignment has been previously submitted for assessment in any other unit or course."

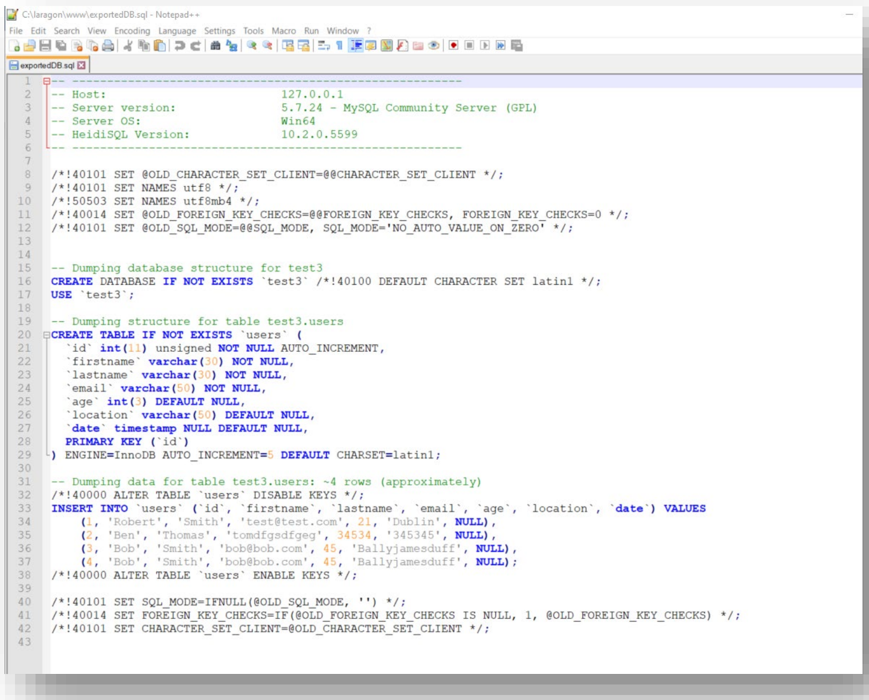
7 Appendix 1: Grading rubric

Structure	None/poor/incomplete *0 points*	Attempt to divide functionality as per MVC design pattern *4 points*	MVC using a front controller, securing DB credentials, DB connection and separation of SQL queries. *6 points*	
Structure	None/poor/incomplete *0 points*	Use of web page templates (header and footer) *4 points*		
CRUD	None/poor/incomplete *0 points*	Basic use of select statements with fetch and fetchAll PDO functions *4 points*	Solid examples of most CRUD functions with PDO. *8 points*	Example of all CRUD functions with PDO (Create, Read, Update, Delete). connection and separation of SQL queries. *12 points*
CRUD	None/poor/incomplete *0 points*	Allow user to register a user account with the website *6 points*	some attempt at shopping cart functionality *12 points*	Strong attempt at shopping cart functionality *18 points*
Output and formatting	None/poor/incomplete *0 points*	Appropriate (readable) display of DB data on a web page with PDO *4 points*	Appropriate web pages and templates to input and/or display data (use of tables, css, iteration, arrays etc...) *6 points*	
Form Validation	None/poor/incomplete *0 points*	Evidence of form validation on the client and the server side *4 points*	All user input correctly formatted and sanitised at the client & server side. *6 points*	
Data Storage	None/poor/incomplete *0 points*	Storing form data externally, in a JSON or a MySQL DB *4 points*	Use of DB for all storage. i.e., no JSON. *6 points*	
Login Logic	None/poor/incomplete *0 points*	Authenticate user login data against a list of stored user data *4 points*	Authenticate user login data against data in a DB *8 points*	
Sessions	None/poor/incomplete *0 points*	Use of sessions for persistence of login credentials *4 points*	Use of session for login and session destroy at end of a session. *8 points*	
Design and appearance	None/poor/incomplete *0 points*	Considered design and navigation structure. Clear and easy to use *4 points*	User can access more pages if logged in. Good use of authentication *6 points*	
Code quality and correctness	None/ poor/ incomplete *0 points*	Code works but could be more efficiently implemented *4 points*	Good use of DRY code (Don't Repeat Yourself) *6 points*	Good use of PHP classes and functions (create your own too) *11 points*
Version control and testing	None/ poor/ incomplete *0 points*	Proof of version control and testing *7 points*		

8 Appendix 2: Submitting your DB

It is important to note that I do not have a copy of your DB on my local machine. Therefore, you will need to provide a copy of your DB. The best approach for doing this is to use HeidiSQL, MySQL workbench or similar software, to export your DB (instruction video here).

Note that the SQL file which is generated by the export process is not actually a database. Rather it is a script that will create a new DB and table identical to yours. It must also insert any input that you have in your database tables. See an example of an export I did using HeidiSQL in Figure 1. *Note here that the exported file is simply a series of SQL commands that can be run in with install.php to create a new DB, and table, and insert the content.*


C:\laragon\www\exportedDB.sql - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Window ?
exportedDB.sql

```sql
-- --------------------------------------------------------
-- Host:                         127.0.0.1
-- Server version:               5.7.24 - MySQL Community Server (GPL)
-- Server OS:                    Win64
-- HeidiSQL Version:             10.2.0.5599
-- --------------------------------------------------------

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET NAMES utf8 */;
/*!50503 SET NAMES utf8mb4 */;
/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0 */;
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;


-- Dumping database structure for test3
CREATE DATABASE IF NOT EXISTS `test3` /*!40100 DEFAULT CHARACTER SET latin1 */;
USE `test3`;

-- Dumping structure for table test3.users
CREATE TABLE IF NOT EXISTS `users` (
  `id` int(11) unsigned NOT NULL AUTO_INCREMENT,
  `firstname` varchar(30) NOT NULL,
  `lastname` varchar(30) NOT NULL,
  `email` varchar(50) NOT NULL,
  `age` int(3) DEFAULT NULL,
  `location` varchar(50) DEFAULT NULL,
  `date` timestamp NULL DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=5 DEFAULT CHARSET=latin1;

-- Dumping data for table test3.users: ~4 rows (approximately)
/*!40000 ALTER TABLE `users` DISABLE KEYS */;
INSERT INTO `users` (`id`, `firstname`, `lastname`, `email`, `age`, `location`, `date`) VALUES
	(1, 'Robert', 'Smith', 'test@test.com', 21, 'Dublin', NULL),
	(2, 'Ben', 'Thomas', 'tomdfgsdfgeg', 34534, '345345', NULL),
	(3, 'Bob', 'Smith', 'bob@bob.com', 45, 'Ballyjamesduff', NULL),
	(4, 'Bob', 'Smith', 'bob@bob.com', 45, 'Ballyjamesduff', NULL);
/*!40000 ALTER TABLE `users` ENABLE KEYS */;

/*!40101 SET SQL_MODE=IFNULL(@OLD_SQL_MODE, '') */;
/*!40014 SET FOREIGN_KEY_CHECKS=IF(@OLD_FOREIGN_KEY_CHECKS IS NULL, 1, @OLD_FOREIGN_KEY_CHECKS) */;
/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
```

Figure 1 Export of the 'tests' DB demonstrated in a previous lecture