Module 15 Server-Side Web Development

Module title	Server-Side Web Development
Module NFQ level (only if an NFQ level can be demonstrated)	6
Module number/reference	BSCH-SSWD
Parent programme(s)	Bachelor of Science (Honours) in Computing Science
Stage of parent programme	Stage 2
Semester (semester1/semester2 if applicable)	Semester 2
Module credit units (FET/HET/ECTS)	ECTS
Module credit number of units	5
List the teaching and learning modes	Direct, Blended
Entry requirements (statement of knowledge, skill and competence)	Learners must have achieved programme entry requirements.
Pre-requisite module titles	None
Co-requisite module titles	None
Is this a capstone module? (Yes or No)	No
Specification of the qualifications (academic, pedagogical and professional/occupational) and experience required of staff (staff includes workplace personnel who are responsible for learners such as apprentices, trainees and learners in clinical placements)	Qualified to as least a Bachelor of Science (Honours) level in Computer Science or equivalent and with a Certificate in Training and Education (30 ECTS at level 9 on the NFQ) or equivalent.
Maximum number of learners per centre (or instance of the module)	60
Duration of the module	One academic semester, 12 weeks teaching
Average (over the duration of the module) of the contact hours per week	3
Module-specific physical resources and support required per centre (or instance of the module)	One class room with capacity for 60 learners along with one computer lab with capacity for 25 learners for each group of 25 learners

Analysis of required learning effort		
	Minimum ratio teacher / learner	**Hours**
Effort while in contact with staff		
Classroom and demonstrations	1:60	18
Monitoring and small-group teaching	1:25	18
Other (specify)		
Independent Learning		
Directed e-learning		
Independent Learning		54
Other hours (worksheets and assignments)		35
Work-based learning – learning effort		
Total Effort		125

Allocation of marks (within the module)					
	Continuous assessment	**Supervised project**	**Proctored practical examination**	**Proctored written examination**	**Total**
Percentage contribution	100%				100%

Module aims and objectives

This module introduces the learner to the fundamentals behind server-side web development. They are introduced to the core concepts behind dynamic, database driven web development, through server-side scripting and database integration and learn how to design and build web applications that deliver database information through server-side HTML pre-processing.

Learners are given practical experience of developing dynamic web sites using these technologies.

Minimum intended module learning outcomes

On successful completion of this module, the learner will be able to:

1. Explain how server-side dynamic web pages are delivered to end-users
2. Design and build dynamic server-side web sites
3. Integrate relational databases into server-side web applications
4. Build a state-based user experience on top of stateless protocols
5. Address security issues in web development and suggest and implement best practice solutions

Rationale for inclusion of the module in the programme and its contribution to the overall MIPLOs

This module enables learners to draw upon various concepts they have already been exposed to, to build dynamic data-driven websites using server-side scripting, client-side web technologies and relational databases. Appendix 1 of the programme document maps MIPLOs to the modules through which they are delivered.

Information provided to learners about the module

Learners receive a programme handbook to include module descriptor, module learning outcomes (MIMLO), class plan, assignment briefs, assessment strategy and reading materials.

Module content, organisation and structure

Web Application Development

- Web architecture
- Client-Server Relationships
- Three-tier applications
- Web Applications
- GET/POST
- Security

Server-side Programming

- Web scripting (PHP)
- processing form data
- validation
- state management (cookies/sessions)
- Security

Integrating Databases

- Database connectivity
- Security

Module teaching and learning (including formative assessment) strategy

The module is taught as a combination of lectures and lab sessions. The lectures discuss and explain to learners the various concepts that underpin server-side web development, from how dynamic web pages are delivered, server-side scripting, to database integration, security and application design.

In tutorials and practical lab sessions, learners practically apply these concepts by building a series of small server-side applications and database-backed websites.

Assessment is a series of practical assignments of increasing complexity that cover the core topics. The first is a simple server-side application with a user interface that writes to flat files. A second application integrates a simple relational database with a server-side web application. Finally, in groups, learners will plan, design and develop a complex database-backed web application.

Timetabling, learner effort and credit

The module is timetabled as one 1.5-hour lecture and one 1.5-hour lab per week.

There are 36 contact hours made up of 12 lectures delivered over 12 weeks with classes taking place in a classroom. There are also 12 lab sessions delivered over 12 weeks taking place in a fully equipped computer lab. The learner will need 45 hours of independent effort to further develop the skills and knowledge gained through the contact hours. An additional 44 hours are set aside for learners to work on worksheets and assignments that must be completed for the module.

The team believes that 125 hours of learner effort are required by learners to achieve the MIMLOs and justify the award of 5 ECTS credits at this stage of the programme.

Work-based learning and practice-placement

There is no work based learning or practice placement involved in the module.

E-learning

The college VLE is used to disseminate notes, advice, and online resources to support the learners. The learners are also given access to Lynda.com as a resource for reference.

Module physical resource requirements

Requirements are for a classroom for 60 learners equipped with a projector, and a 25 seater computer lab for practical sessions with access to SQLite and MySQL, a development environment (such as Notepad++) and a selection of web browsers (Chrome, Firefox, Edge) (this may change should better technologies arise).

Reading lists and other information resources

Recommended Text

Ullman, L. (2017) *PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide*. Berkeley, California: Peachpit Press.

Secondary reading

Connolly, R. and Hoar, R. (2017) *Fundamentals of Web Development*. Boston: Pearson.

Welling, L. and Thomson, L. (2016) *PHP and MySQL Web Development*. Hoboken: Addison Wesley.

Sturgeon, P. (2014) *PHP: The Right Way*. Lean Publications. Available at: https://leanpub.com/phptherightway [4]

Specifications for module staffing requirements

For each instance of the module, one lecturer qualified to at least Bachelor of Science (Honours) in Computer Science or equivalent, and with a Certificate in Training and Education (30 ECTS at level 9 on the NFQ) or equivalent.. Industry experience would be a benefit but is not a requirement.

Learners also benefit from the support of the programme director, programme administrator, learner representative and the Student Union and Counselling Service.

Module Assessment Strategy

The assignments constitute the overall grade achieved, and are based on each individual learner's work. The continuous assessments provide for ongoing feedback to the learner and relates to the module curriculum.

No.	Description	MIMLOs	Weighting
1	Simple web application that receives input from user and writes to text files.	1	15%
2	Simple database backed web application with reading from and writing to a database.	1,2	25%
3	Group-based project where learners design, plan and build a complex database-backed web applications	1,2,3,4,5	60%

All repeat work is capped at 40%.

Sample assessment materials

Note: All assignment briefs are subject to change in order to maintain current content.

[4] Last accessed 26/07/2018

ASSIGNMENT 1 TITLE: NOTE APP

For this assignment you are to use HTML and PHP to build a simple web application that allows user to create, read and edit notes.

The app should feature a homepage with a list of notes and the options to add a new note, edit an existing note or delete notes. Clicking on a note should take the user to a detail page for that note. The detail page should also feature options for editing and deleting.

In total it should have 4 pages:
The home page (listing all current notes)
A note detail page
A note edit page
A new note page

This app does not need a user login system. The notes should be stored as text files.

You can apply a simple amount of CSS to keep the application

Delivery details:
The home page should be named “index.php”.
All files should be contained in a folder named “lastname_firstname_studentnumber_assignment_01”.
Zip this folder and upload to Moodle.

Assessment Criteria:

All features present and working.
Best practice in PHP.
A useable interface

Learning Outcomes Being Assessed:

Understand how server-side dynamic web pages are delivered to end-users.
Design and build dynamic server-side web sites.

ASSIGNMENT 2 TITLE: STUDENT MANAGEMENT SYSTEM

For this assignment you are to build a student management system to store and manage all the necessary information Griffith College need to keep about their students.

The app should provide basic CRUD functionality to allow GCD staff to add, search, update, and delete learner related information.

The app should feature:
A **home page** to welcome and present the CRUD options: Add, Search, Update, Delete.

An **Add page**, with a form to capture the student's details. This should provide a form and an "Add" button. When the form is submitted, the data should be validated before being entered into the database.
Duplicate entries should not be allowed.
Upon success, a message should be displayed e.g. "New student added successfully" with a link to the homepage.

A **Search page**. The page should provide a search box and ask users to provide a student number. After entering a student number and clicking a search button, the system should display student information if found or display a message (e.g. "Student not found")

An **Update page**: The page should provide a search box and ask users to provide a student number. After entering a student number and clicking a search button, if the student is not found it should display a message; otherwise it should display the student's existing information displayed in a form with an update button. The user should be able to update all the information, except for the student number. After updating, an appropriate message should be displayed.

A **Delete Page.** Similar to the update page, this should provide a search box and ask users to provide a student number. After entering a student number and clicking a search button, if the student is not found it should display a message; otherwise it should display the student's existing information displayed as text with a delete button. Clicking the delete button should delete the entry and display an appropriate message.

This app does not need a user login system. The students should be stored in a MySQL database. It should have at least one table.
Form fields should be appropriate and all entered data should be validated.

You can apply a simple amount of CSS to keep the application usable.

Delivery details:
The home page should be named "index.php".
All files should be contained in a folder named "lastname_firstname_studentnumber_assignment_01".
Zip this folder and upload to Moodle.

Assessment Criteria:

All features present and working.
Best practice in PHP.
A useable interface

Learning Outcomes Being Assessed:

1. Design and build dynamic server-side web sites.
2. Integrate relational databases into server-side web applications

PROJECT TITLE: SOCIAL NETWORK

For this group assignment you are to design and build a student social network system. The system should allow users to:

- Sign up for an account
- Log in
- Edit a profile page (including uploading a picture)
- Post updates (short text updates or photo)
- Follow friends
- See updates from friends listed chronologically on a feed
- Like or comment on updates from friends

It should also allow for an admin account which can:
Delete users / posts

You should also add your own features as you see fit, for example:
A photo gallery
A private messaging system

The application should be built in HTML, CSS, PHP & MySQL. You can use a front-end templating system such as Bootstrap, but the PHP/MySQL should be self-written using not templates or frameworks.

The database should have as many tables as necessary to accommodate the features.

Delivery details:
The home page should be named “index.php”.
All files should be contained in a folder named “groupnumber_group_assignment”.
(Group numbers will be provided to all groups)
Zip this folder and upload to Moodle.

In addition to the site, a document should be provided that outlines:
An overview of the system including the features it has.

A diagram of the database
A review of the process, including what worked and what didn't.

This should be named "design_document.pdf" and saved in PDF format. Include this in the same folder.

Assessment Criteria:
10% - Visual design. Easy to use and understand with consistent navigation
10% - Database Design
30% - System design and implementation. Working features.
20% - Code Quality
10% - Extra features
10% - Security / Form validation
10% - Documentation / Demo

Learning Outcomes Being Assessed:

1. Understand how server-side dynamic web pages are delivered to end-users.
2. Design and build dynamic server-side web sites.
3. Integrate relational databases into server-side web applications
4. Build a state-based user experience on top of stateless protocols
5. Recognize security issues in web development and suggest and implement best practice solutions.

Notes:

This will be done in groups of 3.

All code must be original

In order to pass, students must present a working and usable application that covers the basic specifications provided above.