CLSICT0064 Assessment Task 2

Summary

CLS-ICT-0064: Assessment Task 2

Cluster Name: CLSICT0064: Web Services Cluster

Unit Names

ICTWEB514 - Create dynamic web pages

ICTWEB518 - Build a document using extensible markup language

Assessment Name: Dynamic Website and XML Portfolio

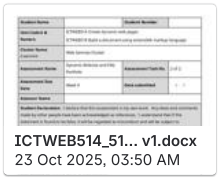
Purpose: To design a dynamic website that uses XML documents to insert data into a database

Due Date: 16-Nov-2025 (Week 9)

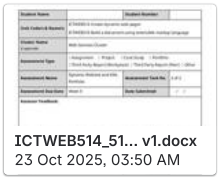
Contents

› High Street Gym

Documents



ICTWEB514_51... v1.docx
23 Oct 2025, 03:50 AM



ICTWEB514_51... v1.docx
23 Oct 2025, 03:50 AM

Due Date

Nov 16, 2025

Status

IN PROGRESS

High Street Gym

Contents

High Street Gym Solution Architecture

Purpose

Provide the **implementation architecture and evidence** for **CLSICT0064 AT2 – Part 2**: dynamic pages, database, and XML integration

Enterprise Architecture (HEAL) Reference

This solution **adopts HEAL's generic MVC** structure (Dynamic Page Pattern) and references HEAL's generic **Data Patterns** and **Validation & Accessibility** checklists. HEAL remains the **generic reference**; this page contains **project-specific evidence** only

- Dynamic Page Pattern
- Data Patterns
- Validation & Accessibility Checklist

Solution MVC Structure

```
highstreetgym/
├─ public/
│  └─ index.php              # front controller (single entry)
├─ controllers/
│  ├─ authcontroller.php
│  ├─ classescontroller.php
│  ├─ bookingcontroller.php
│  └─ blogcontroller.php
├─ models/
│  ├─ user.php
│  ├─ classmodel.php
│  ├─ sessionmodel.php
│  ├─ booking.php
│  └─ blogpost.php
├─ views/
│  ├─ layouts/base.php
│  ├─ partials/{nav.php, flash.php}
│  ├─ auth/login.php
│  ├─ classes/index.php
│  └─ blog/index.php
└─ support/
│  ├─ db.php     # PDO singleton (no globals)
│  ├─ view.php   # layout/partials renderer
│  ├─ csrf.php   # optional (self-directed)
│  └─ auth.php   # session helpers
└─ config/
   ├─ env.php      # DB settings (not in repo)
```

```
└── routes.php    # method + path → controller@action
```

HSG Boilerplate Templates

Boilerplate Sections & Features

Common sections & features

1. **Page Layout** (`views/layouts/base.php`) — head/meta/footer and `$content` slot
2. **Navigation** (`views/partials/nav.php`) — ARIA-labelled primary nav with skip link
3. **Flash** (`views/partials/flash.php`) — session messages (aria-live)
4. **Login form** (`views/auth/login.php`) — reusable auth form
5. **List template** (`views/resource/index.php`) — reusable list page structure
6. **Database connector** (`support/database.php`) — PDO singleton
7. **Router** (`config/routes.php`) — GET/POST mapping to controllers
8. **Controller skeleton** — minimal handler method returning a view

---

File Structure

highstreetgym/

```
highstreetgym/
├─ index.php
├─ controllers/
│  ├─ database.php        ← PDO connector (course naming)
│  ├─ authcontroller.php
│  ├─ classescontroller.php
│  ├─ bookingcontroller.php
│  └─ blogcontroller.php
```

```
├── models/
│   ├── user.php
│   ├── sessionmodel.php
│   ├── classmodel.php
│   ├── booking.php
│   └── blogpost.php
├── views/
│   ├── layouts/
│   │   └── base.php
│   ├── partials/
│   │   ├── nav.php
│   │   ├── header.php
│   │   ├── footer.php
│   │   ├── table.php        ← generic table/timetable partial
│   │   └── list.php         ← generic list partial
│   ├── auth/
│   │   ├── login.php
│   │   └── register.php
│   ├── classes.php
│   └── blog.php
└── assets/
    └── css/
        └── style.css
```

Boilerplate Files

<details open>
<summary>Page Layout</summary>

- `views/layouts/base.php`
  - `views/partials/nav.php`
  - `views/partials/header.php`
  - `views/partials/footer.php`
- `assets/css/style.css`

</details>

<details open>
<summary>Navigation</summary>

- `views/partials/nav.php`
  - Reuse site-wide

</details>

Data display (tables & lists)

- `views/partials/table.php` (timetable/table partial)
- `views/partials/list.php` (generic list partial)

Registration/Login

- `views/auth/login.php`
- `views/auth/register.php`

Database connection

- `support/database.php`

Controllers

- `controllers/*controller.php` (auth, classes, booking, blog)
- `index.php` (front controller/router)

Models

- `models/*.php` (user, sessionmodel, classmodel, booking, blogpost)

Styling

- `assets/css/style.css` (site-wide responsive + accessible defaults)

HSG Database

Purpose

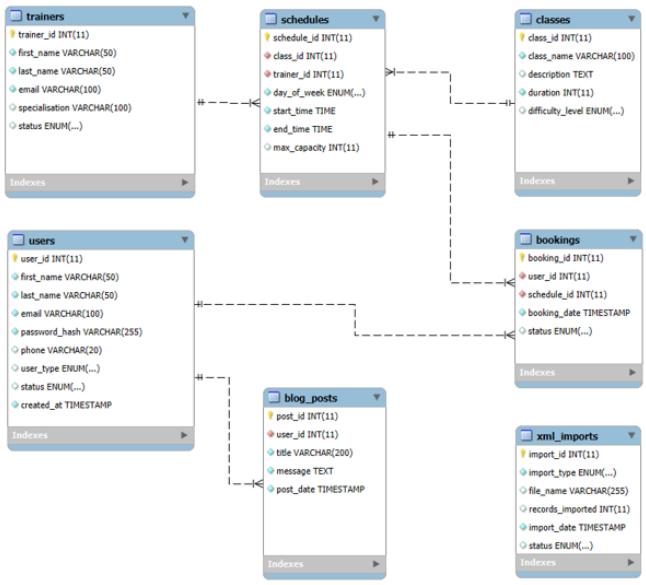
Describe High Street Gym's concrete implementation of the HEAL Booking pattern and expose the SQL artefacts for AT2 Part 2

HEAL Alignment & Variances

Booking Domain Reference Pattern

- Service → Class
- Provider → Trainer

Entity Relationship Diagram



Schema

High Street Gym Database Schema

- **users** – members/admins, `user_type ENUM('member','admin')`, password stored as `password_hash`
- **trainers** – staff with optional specialisation
- **classes** – catalogue (name, duration, difficulty)
- **schedules** – the "Session" table (day_of_week, start_time, end_time, max_capacity) referencing `classes` and `trainers`

- **bookings** – member bookings; **UNIQUE(user_id, schedule_id)**; cascades on delete
- **blog_posts** – member-generated messages (title, message, post_date)
- **xml_imports** – simple audit/log of XML batch loads (type, file, status)

The SQL DDL and seed data are stored in **schema.sql** in the repo
Execute this file to provision the database

Use Case Patterns

Timetable & booking (query snippets to implement):

- Weekly timetable:
  - select from `schedules` join `classes` + `trainers` filter by `day_of_week`, order by `start_time`
- Booking action:
  - insert into `bookings (user_id, schedule_id)` guarded by the unique key; surface friendly error if duplicate

---

Blog (dynamic list):

- Fetch latest `blog_posts` join `users` for author; reuse the **List** boilerplate view

---

Seed/admin

- One admin and one member are included in the seed for testing login flows (Passwords hashed with `password_hash`)

HSG Dynamic Pages

Login

<details>
<summary>Click here to expand...</summary>

- uses `views/auth/login.php` + `authcontroller.php`;
- server-side verifies `users.password_hash`;
- client-side form validation

</details>

Timetable/Book

<details>
<summary>Click here to expand...</summary>

- `classescontroller.php` lists sessions from `schedules` with class + trainer;
- `bookingcontroller.php` handles POST to create `bookings`, respecting unique constraint

</details>

Blog

<details>
<summary>Click here to expand...</summary>

- `blogcontroller.php` renders posts from `blog_posts`;
- creation limited to authenticated members

</details>

High Street Gym Solution Delivery

Solution Deliverables

Search

Expand all Collapse all

- Part 1: Confirm requirements and gather resources for the project website
- Part 2: Project Implementation
- Part 3: Project testing and debugging
- Part 4: Web programming research

Part 1: Confirm requirements and gather resources for the project website

Task 1:

Conversation Log

1.1: Conversation Log

Date	Topic	Discussion	Stakeholder
20/10/2025	Project Kick-off	Discussed the overall objective of creating a dynamic booking website for High Street Gym. Confirmed that the site will allow members to register/login, view weekly class timetables, and book sessions with trainers. Agreed that XML will be used for transferring data such as new member details and new class entries into the database.	Uptown IT Project Manager
21/10/2025	Confirmation of general database requirements	Confirmed that at least two XML documents will be implemented: one for adding a new member, and another for adding new classes.	High Street Gym Sponsor
22/10/2025	Legislative and organisational standards	Researched relevant Australian and Queensland legal and policy requirements for website design. Reviewed the Queensland Government's Online Standards, Policies and Legislation page, confirming privacy, accessibility, and security obligations under the Information Privacy Act 2009, Disability Discrimination Act 1992, and related standards such as IS18:2018 and WCAG 2.1. Discussed with manager how these apply to dynamic pages and XML data handling, and confirmed Uptown IT organisational standards for secure coding and accessibility.	Uptown IT Project Manager
23/10/2025	Software and technology selection	Confirmed use of HTML and CSS for the front-end interface, and PHP with MySQL for server-side processing and database management. Also confirmed XML 1.0 (Fifth Edition W3C 2008) for data input into database	Uptown IT Project Manager

Task 2:

Confirm dynamic website requirements and legislative requirements

1.2: Dynamic Website Requirements

Dynamic Website Requirements	Comments
Purpose: To provide a secure, dynamic and accessible website for High Street Gym members to view timetables, book classes with specific trainers and share experiences via blog.	Include admin functionality to add classes and members via XML documents
Scope: Included: · User registration/login/authentication · Class/trainer/member management · Member blog Excluded: · Membership payments · Financial management · Gym administration	Ensure architecture extensible and scalable for potential future expansion of functionality
Functional requirements: As a visitor to the High Street Gym website I want to see the weekly class schedule So that I can make an informed decision about joining As a High Street Gym member I want to login securely the High Street Gym members' area So that I can book classes online with my favourite trainers As a High Street Gym member I want to write blog updates on the website So that I can share my experiences and support my fellow members As a High Street Gym trainer I want to view assigned classes So that I can prepare and deliver my training sessions on time	System functions needed to meet user expectations: · Registration/login functionality · Class browsing/booking · Blog upload
Technical requirements: · Standards-compliant page rendering and styling · Reliable client–server communication and data validation · XML data transfer to the database · Compatibility with major browsers and devices · Code compliance	Validated HTML & CSS PHP connection Validated XML to update SQL database Semantic HTML & CSS media queries W3C Markup Validation/CSS Validator XML validation against DTD
Non-functional requirements: · Performance · Availability · Usability	Fast response time and rendering

· Compatibility	Reliable continuous browser access Easy to use and accessible Reliable across browser and devices
Business requirements: As a High Street Gym admin I want to add classes and assign trainers So that members can select the classes they want to attend As a High Street Gym admin I want to add new members So that they can access class selection and blog functionality	Administrative functions: · Trainer/class management · Member management
Stakeholder requirements: Delivery on time and within budget Methodology and change management agreement	Cost approval process defined MVP defined and costed Scrum methodology with embedded stakeholders for knowledge transfer
Security requirements: Role-based access controls Multi-factor authentication Password/data encryption User input validation Secure data transfer protocols	Prepared statements XML validation HTTPS with SSL/TLS XSS prevention
Quality requirements: Validated code Unit testing Functional testing Usability testing Penetration testing	HTML: W3C Markup Validation CSS Validator XML validation against DTD Test-driven development

1.3: Related Legislative & Organisational Requirements

Australian and Queensland laws and government policies that apply to dynamic websites and XML data:

- Disability Discrimination Act 1992 (Cwlth)
- Information Privacy Act 2009 (Qld)
- Public Records Act 2002 (Qld)
- Right to Information Act 2009 (Qld)

Supporting Queensland Government policies and standards include:

- Information Security Policy (IS18:2018)
- Digital Service Standard

- Web Content Accessibility Guidelines (WCAG 2.1)
- Consistent User Experience Standard

These ensure the High Street Gym website meets privacy, security, and accessibility expectations.

Dynamic and XML application:

- Dynamic pages (e.g. login, class bookings) will use secure PHP and MySQL
- XML documents will be used for data exchange and validated against a DTD

Organisational standards:

- Uptown IT applies coding and testing conventions that include:
  - Clean code structure (HTML & CSS validation)
  - Secure data management
  - Consistent accessibility design
  - Responsive design
  - Organisational XML development and management policies and procedures

1.4: Legislative & Organisational Requirements Confirmation

Task 3:

Select project software and obtain manager approval to proceed

1.5: Project Software

Software/Technologies	Justification
Client-side languages: HTML5 and CSS3	Used to build and style web pages that meet web and accessibility standards Provides a consistent structure and presentation for all pages
Client-side scripting: JavaScript ES6	Adds dynamic interaction and client-side form validation. Enhances the user experience without requiring a page reload
Server-side scripting: PHP 8.2	Processes user requests Connects to the database Manages authentication and XML parsing Enables dynamic page generation
Database platform: MySQL	Stores user, class, trainer, and booking information Integrates effectively with PHP for secure data storage and retrieval
XML platform: XML 1.0 (fifth edition)	Used to structure and exchange data, such as adding new members or classes Supports validation of data before insertion into the database
CSS framework: Responsive design framework (e.g. Bootstrap 5 or similar)	Ensures the site adapts to different devices and screen sizes while maintaining accessibility and usability

Development environment: Local web server environment (e.g. XAMPP) Visual Studio Code Github	Provides an integrated environment for developing and testing dynamic web pages locally, then deploying to production
Testing tools: Web validators Browser developer tools	Used to validate code syntax, confirm accessibility compliance, and test responsiveness and performance

1.6: Manager Approval

Project Requirements SIGNOFF Signing off on this document signifies that the project requirements have been identified and documented.	
Project Manager	**QA Officer**
Signature: *Cameron Hughes*	Signature: *Sheralyn Hume*
Date: 21st October 2025	Date: 21st October 2025
Project Requirements NOT APPROVED	
Please provide feedback on the changes needed.	
APPROVAL to proceed to next stage:	☒ Granted ☐ Not Granted

Part 2: Project Implementation

Task 1:

Dynamic features and database access

2.1: Identify & create boiler templates

- HSG Boilerplate Templates

The following common repeatable sections and features have been identified for the creation of boilerplate templates:

Common sections & features	Boilerplate templates	Solution uses
Page Layout	Default page layout: header logo/title nav bar main content footer	Classes Trainers (admin) Bookings Blog
Navigation	Default site navigation: nav bar nav list nav button	Site Navbar
Tables	Timetable template	Classes schedule
Lists	List template	Blog posts (populated from XML document) Bookings lists (populated from XML document)
Registration/Login	Registration form Login form	Member registration Admin/Member login
Database connection	Database connector	Secure database connection from site models
Controllers	Controller template Consistent method to manage authentication, sessions, page/function access	authentication controller classes controller booking controller blog controller
Models	Models template Consistent method to manage business rules and data integrity, logic, manipulation & access	user model session model class model booking model blogpost model

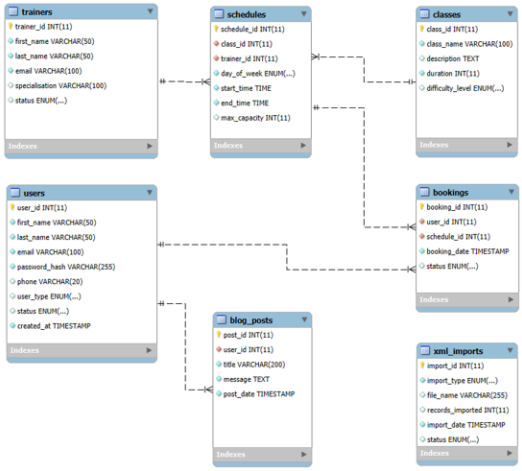
Styling	Site CSS file Consistent, responsive and accessible user experience	Site page and section styling rules

Create the template/s

2.2 Create the relational database

HSG Database

HSG Database | Untitled excerpt



2.3 Create the web pages

If appropriate, use the template/s created in Task 1 (2.1) to implement the web pages. The dynamic web features identified in the requirements must be implemented in at least three (3) separate pages. Each page must include client-side and server-side scripting.

The task includes creating pages to:

- Process authentication
- Display Gym classes calendar/timetable (weekday, time, duration, level and so on) from the database and the facility to book a class and a trainer – if more than one is available
- Display a members' blog with functionality to upload and read messages
- In consultation with your project manager, other pages may be included for extra functionality in the website

2.4 Document implemented functionality & technology

Complete the table below to identify which functionality has been implemented and which technology has been used for each function.

Page	Functionality	Software/Technology Used
Page 1		

	Add rows as necessary	

Task 2:

Implementation plan XML pages

2.5 Review & summarise XML requirements

As per the scenario presented, you need to create at least two (2) XML documents to parse XML data into your database using the server-side language selected in PART 1-Task 3. Review the XML requirements and complete the table below.

XML Document	Purpose	Expectations	Functionality

Add rows as necessary

2.6 Outline plan to create XML documents

Outline a plan to create the XML documents, taking into consideration that the XML documents must integrate with and complement the dynamic website features in an iterative process. Identify the design methodology that you are planning to use and indicate at which stage of the methodology you would create and integrate the XML document/s to the webpages. Identify the software that would be used to create the XML pages.

XML Document 1	Selected Design Methodology	Stage	Software/Tools

Add rows as necessary

XML Document 2	Selected Design Methodology	Stage	Software/Tools

Add rows as necessary

2.7 Identify & document the DTD

Identify and document the DTD (Document Type Definition) including the XML documents structure, their entities, elements, and attributes. Any media associated

features must also be included. You can use graphics to support your written explanation to describe the XML document structures and elements.

2.8 Explain method to embed XML into HTML pages

Determine and explain the method you plan to use to embed XML into the HTML pages.

2.9 Discuss organisational policies, procedures & standards with manager

Discuss with your manager the organisational policies, procedures, and standards applicable to developing the XML documents. Update the Conversation Log.

2.10 Create XML pages

Create XML pages as per the plan outlined in 2.6.

Part 3: Project testing and debugging

Task 1:

Project testing and debugging

3.1: Debug code

Debug the code as necessary and present proof of debugging by providing screenshots of the process. Two instances of debugging are sufficient.

3.2 Validate HTML & XML code

Validate the HTML and the XML code against current industry standard specifications. Provide screenshots of validation reports.

3.3 Test functionality

Test and evaluate each functionality required and identified in PART 2 is working as expected. This includes programming features and functionality (client and server-side programming). Test website functionality in at least two (2) browsers and two (2) devices. Fix as necessary. Re-test. Provide screenshots as evidence. Use template provided.

3.4 Test XML documents

a) Test XML document offline. Amend as required. Provide Screenshots.

b) Test XML document online. Amend as required. Provide screenshots.

3.5 Outline implemented security measures

Outline what security measures have you implemented on the website for the following three areas:

a) Authentication process

b) Programmatically engineered solutions to avoid cyber-attacks

c) Internet protocols

3.6 Record feedback meeting

Record a simulated meeting with the project manager seeking feedback on the project implementation. For this component you can conduct the meeting with a family member, friend, another student or work colleague. Update website and XML documentation as required. Record the meeting in the Conversation Log.

	Important
The above simulated meeting should be recorded and the recording submitted as part of your final assessment. The recording should be approximately 7 - 10 minutes.	

3.7 Email project completion to manager

Email to manager communicating the project completion.

Website Implementation SIGNOFF Signing off on this document signifies that all required dynamic features and XML documents comply with the Client's requirements.	
Project Manager	**QA Officer**
Signature:	Signature:
Date:	Date:

Dynamic Website and XML documents NOT APPROVED
Please provide feedback on the changes needed.

3.8 Contingency task

Assume that you have completed the project and have successfully tested it in three (3) popular browsers as requested. However, the client contacts you with a new request to make the website cross-browser compatible with a new not widely used browser. After some testing, you identify that one specific key dynamic feature is not supported by the new browser. How would you proceed with the client at this stage?

Part 4: Web programming research

Task 1:

Web programming research

4.1 Web programming concepts

a) Compare multi-factor authentication and certificate-based authentication.

Suggest the most appropriate web security method (multi-factor or certificate-based) for the web project that you have completed in this portfolio. Justify your selection.

b) Research HTTP (Hypertext Transfer Protocol) and identify the current version of the standard specification. Select and explain two (2) new features of the current specification that improve web security.

c) Propose and provide examples of two (2) limiting features of stateless programming that can be overcome using session management.

d) Suggest and outline two (2) strategies that can be used to extend data storage in web applications.

4.2 Web Technologies

a) Research the progression of HTML specifications from its beginnings in 1993 and theorise what could be the next areas/technologies that will need to be covered by HTML syntax.

b) Identify and describe three (3) differences between CSS grid layout and CSS Flex model. Based on the differences identified, explain which CSS model is better suited for the web project in this portfolio.

4.3 Programming good practice

a) Good coding techniques include (1) the use of an appropriate naming convention and (2) avoiding deep

nesting. For each technique, outline the consequences of not applying the technique to the code.

b) The basic programming control structures are sequence, selection, and iteration. Analyse how iteration works and propose other methods or techniques to achieve iteration or repetition.

c) Select three (3) the areas covered by syntax rules in programming. For each area, explain what is gained by using the specific programming language syntax.

d) In reference to the web project in this portfolio, provide an explanation to justify the use of client and server-side scripting.

e) Propose three (3) techniques that could be used to minimise code errors and the need to debug (fixing code errors). Provide examples to illustrate your answer.

f) Identify and describe two (2) debugging methods that can be used to test server connections.