University of Brighton

School of Architecture, Technology & Engineering

Assessment Brief

Module Title:	Web Application Development
Module Code:	CI527
Author(s)/Marker(s) of Assignment	Marcus Winter

Assignment No:	2
Assignment Title:	REST API server
Percentage contribution to module mark:	50%
Weighting of component assessments within this assignment:	n/a
Module learning outcomes covered:	LO1: Demonstrate an understanding of client-server Web applications and related concepts, protocols and technologies LO3: Develop data-centric server software exposing REST APIs to a given specification

Assignment Brief and Assessment Criteria: See following pages.

Date of issue:	05/02/2024
Deadline for submission:	10/05/2024 at **3pm** Note: Students are allowed to submit work within two weeks of the published deadline, or the last working day immediately prior to the feedback date if this is shorter than two weeks. Late work is capped at the pass mark of 40%.
Method of submission:	e-submission via student**central**
Date feedback will be provided	07/06/2024

Assignment brief – REST API server

In this assignment, you will develop a simplified REST API server that can be used to create, store and retrieve text comments for objects based on their unique ID. You will develop the REST API against the specification provided in **Appendix A1**.

Your REST API server will:

- be written in object-oriented PHP
- use a MySQL database to store and retrieve comments
- carry out appropriate security checks and error handling
- conform **exactly** to the specification provided in **Appendix 1**
- not use any external libraries or frameworks

You will carry out the required technical research in order to design, build, deploy and test the REST API server on your personal university web space. To support you in the process, there will be lab sessions where we discuss related concepts and patterns, develop technical prototypes and provide formative feedback.

Deliverables

Your REST API server **must be hosted on your personal university web space** e.g. `https://username.brighton.domains/ci527/assignment2/api.php`

The assignment must be submitted online, through the CI527 assessment area on studentcentral. Please submit the following:

1. A **single ZIP file** containing:

    - source code of your REST API server (`api.php`) script
    - export of the MySQL database used by your REST API server
      (use PHPMyAdmin to export your database in `.sql` format.)

    The zip file should be named as follows:

    lastname_firstname_studentnumber_CI527_assignment1.zip

2. In the **text field** of the submission form you must provide:

    - The URL of your REST API endpoint (see example above)

    Make sure the link works!

Marking criteria

- see **Appendix A2**: Marking criteria -

Appendix 1: REST API specification

The REST API must provide a single endpoint called **api.php**.

Example:

`http://username.brighton.domains/ci527/assignment2/api.php`

The REST AP must support two CRUD operations (Read, Create) through their related HTTP request methods (GET, POST).

- POST requests should submit data in the request body (*Content-Type: application/x-www-form-urlencoded*)
- GET requests should submit data as part of the URL (*URL Encoded*).
- Responses for both POST and GET requests should encode data in JSON format (*Content-Type: application/json*)

Other HTTP request methods should be answered with status code 405 (Method Not Allowed) and an empty response body.

Create	**Create new comment for the specified object**
HTTP method	**POST**
Parameters	**oid** Object ID: Unique ID of the object the submitted comment refers to. Alphanumeric string (0..9, a-z, A-Z), maximum length 32 characters. Example: `oid=AB123456X`
	name Name of the author of the comment. String. Minimum length 1 character. Maximum length 64 characters. Example (URL-encoded): `name=marcus%20winter`
	comment The submitted comment. String. Minimum length 1 character. No maximum length. Example (URL-encoded): `comment=this%20is%20a%20comment`
Response	**HTTP status codes:** 201 - OK, record created 400 - Bad request (e.g. parameter missing or invalid) 500 - Internal server error (e.g. database connection failed)

	Response body for status code 201:
	JSON data string containing the id of the newly created record.
	Example:
	`{`
	`    "id": 10`
	`}`

	Response body for all other status codes: - empty -
Read	**Read comments for the specified object**
HTTP method	**GET**
Required parameters	**oid** Object ID: Unique ID of the object for which comments are returned. Alphanumeric string (0..9, a-z, A-Z), maximum length 32 characters. Example: `oid=AB123456X`
Response	**HTTP status codes:** 200 - OK 204 - OK, no content (e.g. no comments for this object id) 400 - Bad request (e.g. parameter missing or invalid) 500 - Internal server error (e.g. database connection failed)
	Response body for status code 200: JSON data string containing the object id (oid) specified in the request, and an array of comment objects, sorted in ascending order by the date they were submitted (i.e. written to the database). Each comment object should include: • id (unique ID of the comment in the database, as a numeric integer value) • date (timestamp the comment was written to the database, formatted as a calendar date in the form *"dd mmmm yyyy"* see example below) • name (comment author) • comment (the actual comment)

Example:

```
{
    "oid": "AB123456X",
    "comments": [
        {
            "id": 12,
            "date": "23 February 2024",
            "name": "Mia Kumar",
            "comment": "This is a comment"
        },
        {
            "id": 64,
            "date": "08 May 2024",
            "name": "Aslan",
            "comment": "Another comment"
        }
    ]
}
```

Response body for all other status codes:

- empty -

Appendix A2: Marking criteria

	A 70% and above	B 60% to 69%	C 50% to 59%	D 40% to 49%	E 30% to 39%	F Less than 30%
Fitness for purpose 40%	The REST API server conforms fully to the specification, includes appropriate presence and security checks on submitted data, gracefully handles incorrect calls, and returns valid status codes, headers and data.	The REST API server conforms almost fully to the specification, includes presence checks and some security checks on submitted data, gracefully handles incorrect calls, and returns valid status codes, headers and data.	The REST API server conforms mostly to the specification, includes presence checks and basic security checks on submitted data, can handle correct API calls, and returns mostly valid status codes, headers and data.	The REST API server conforms mostly to the specification but does not appropriately check submitted data and/or has problems handling API calls and/or returns invalid status codes, headers or data.	The REST API server does not conform to the specification, does not appropriately check submitted data or has problems running.	Work submitted, but criteria for grade E not met.
Code quality 40%	The code is lean, logically consistent and follows best coding practice and security standards throughout.	The code has only minor consistency problems and mostly follows best coding practice and security standards.	The code has some consistency problems and/or problems with best coding practice or security standards.	The code has problems with consistency and best coding practice or security standards.	The code is of poor quality and/or has problems running.	Work submitted, but criteria for grade E not met.
Database design 20%	The database follows best practice, is appropriately structured and uses appropriate field names, types and sizes.	The database follows best practice, is appropriately structured but does not always use appropriate field names, types or sizes.	The database mostly follows best practice but is overly complex or does not use appropriate field names, types or sizes.	The database does not follow best practice but is able to hold the specified data.	The database is not fit for purpose but capable to store some of the required information.	Work submitted, but criteria for grade E not met.