# Solent University

# Coursework Assessment Brief

# Assessment Details

|  |  |
| --- | --- |
| Module Title: | Web Application Development (Level 5) |
| Module Code: | COM518 |
| Module Leader: | Nick Whitelegg |
| Level: | 5 |
| Assessment Title: | Points of Interest Application |
| Assessment Number: | AE1 |
| Assessment Type: | Software Product |
| Restrictions on Time/Word Count: | 2000-3000 words (for guidance only) |
| Consequence of not meeting time/word count limit:. | None, however it should not be necessary to exceed 3000 words. |
| Individual/Group: | Individual |
| Assessment Weighting: | 50% |
| Issue Date: | 23 January 2023 |
| Hand In Date: | 28 April 2023 |
| Planned Feedback Date: | 4 working weeks after hand in. |
| Mode of Submission: | via SOL |
| Anonymous Marking | This assessment **is** exempt from anonymous marking. |

**COM518 Web Application Development**

**Assignment 1**

**UPDATED VERSION – to allow the use of SQLite**

**Scenario**

Your task is to develop PointsOfInterest, a web application which allows users to look up information on places they might want to visit whilst on holiday, such as world cities, historical sites, countryside areas, or recommended restaurants or pubs/bars.

You are required to build PointsOfInterest according to the specification below. You must use Node and Express as the back-end technology, and SQLite, MySQL or MariaDB for the database.

**Database**

The database should follow the structure below.In your implementation, you may choose to use additional database tables. If you do, they must be documented, with justification.

You will be provided with an SQLite .db file.

*pointsofinterest - represents individual points of interest*

|  |  |  |
| --- | --- | --- |
| **Column** | **Type** | **Role** |
| id | INT, PRIMARY KEY | An auto-incrementing index uniquely identifying each record |
| name | VARCHAR(255) | the name of the POI |
| type | VARCHAR(255) | the POI type, e.g. city, historical building, restaurant, hotel, etc |
| country | VARCHAR(255) | the country that the POI can be found in |
| region | VARCHAR(255) | the region that the POI can be found in, e.g. Hampshire, Normandy, Bavaria, California, etc |
| lon | FLOAT | the longitude of the POI |
| lat | FLOAT | the latitude of the POI |
| description | TEXT | the POI's description |
| recommendations | INT | number of recommendations for this POI |

*poi\_users - represents PointsOfInterest's users*

|  |  |  |
| --- | --- | --- |
| **Column** | **Type** | **Role** |
| id | INT, PRIMARY KEY | An auto-incrementing index uniquely identifying each record |
| username | VARCHAR(255) | the username |
| password | VARCHAR(255) | the password |

*poi\_reviews - represents reviews of points of interest*

|  |  |  |
| --- | --- | --- |
| **Column** | **Type** | **Role** |
| id | INT, PRIMARY KEY | An auto-incrementing index uniquely identifying each record |
| poi\_id | INT | the ID of the point of interest that this review relates to (from the pointsofinterest table) |
| review | TEXT | the review itself |

**Completing the assignment**

Your submission must include:

a) a report describing how you developed the code, including details of how your code works, any problems encounterered, and how you solved them. This should be around 250-500 words per task; simpler tasks will require less, while more complex tasks will require more. (40%)

b) working code. (60%)

You will be graded separately for the report and the code, with the report grade counting for 40% of the final grade and the code worth 60% of the final grade.

The grades you will achieve for completing a given number of tasks are indicated in the Task Detail, below. These apply to both the report and code. For example if you completed perfectly up to task 8 for the code BUT only covered tasks 1-4 (perfectly) in the report, you would get C1 for the code and F1 for the report. To arrive at your overall grade, these are converted to numbers (C1=58 and F1=35) and an overall numerical mark calculated, e.g.:

58\*0.6 + 35\*0.4 = 48.8

The numerical mark is then rounded to the nearest grade (D1 in this case).

Errors in the code, or unclear discussion and/or omissions in the report, will lower your grade for the appropriate component (code and/or report).

**Task Detail**

**Part A – Develop a very simple REST API**

You should first develop a simple REST web API using Node and Express which allows clients to:

1. Look up all points of interest in a given region. It should return the results as JSON.

2. Add a new point of interest. This API endpoint should simply read in the point of interest details as POST data, and add them to the database.

3. Recommend a point of interest. This API endpoint should read in the POI ID and increase the number of recommendations by one for that POI.

Task 1 can be tested directly in the browser. Use RESTer or a similar tool to test tasks 2 and 3.

***If you get this far, you will achieve a F2***

**Part B – Develop a simple AJAX-based JavaScript front-end**

Next, you should build a simple HTML and JavaScript front-end which communicates with your REST API using AJAX (no page reload should be necessary). You can optionally use React for extra marks (see below)

4. Write an HTML page which allows the user to search for all points of interest in a given region. The user should be able to enter a region, and then, using JavaScript, and AJAX, the page should communicate with your REST API to find all points of interest in that region. The JSON must be parsed, and the results presented to the user in a user-friendly way.

5. Write another HTML page which allows the user to enter point of interest details. Again, using JavaScript and AJAX, the page should communicate with your web API. Finally, using a standard HTML hyperlink, link the HTML page to the task 4 HTML page.

6. Modify your code to process the search results, so that you create a “Recommend” button for each result. When the user clicks on this button, you should send an AJAX POST request to the REST API (task 3) to allow the user to recommend the POI.

***If you get this far, you will achieve a D2.***

**Part C – Adding simple error-checking**

.

7. Add error-checking to task 2, so that if any of the POI details are blank, an appropriate HTTP error code is sent back to the client. Then, in task 5, test for the HTTP code returned from the server and display an appropriate message to the user.

***If you get this far, you will achieve a D1.***

**Part D – Adding a map**

8. Using Leaflet, add an OpenStreetMap map to Task 4, so that the results are displayed as markers on the map. When a user clicks a marker, the point of interest name and description should appear as a popup.

You must use Leaflet and OpenStreetMap. In particular, Google Maps is NOT acceptable.

***If you get this far, you will achieve a C3***

9. Allow the user to add a POI by clicking on the map at a particular location. When the map is clicked, the user should be able to enter the POI details. When a button is clicked, the data should be sent over to your REST API (task 2) and a marker should be added to the map. The marker should appear only if the REST API returns a success HTTP code. Check for errors returned from the REST API and communicate them to the user in a user-friendly way.

***If you get this far, you will achieve a C1***

**Part E – logins and sessions**

10. Implement a session-based login system. A user should be able to login from the main index page (task 1). If a user logs in successfully, a message should appear within a <div> on the index page, e.g.

Logged in as jsmith

You should also inplement a logout facility. Ensure the logged-in message still appears if the user reloads the page. There is no need to implement a signup facility, as the SQL file to populate the database contains existing users.

***If you get this far, you will receive a B3***

11. Change task 2 so that a user must be logged-in to add a point of interest, sending back an appropriate HTTP error if they are not. Also change task 9 so that this error is checked, and an appropriate error message displayed to the user if they are not logged in.

*(Note – if you know anything about REST you will realise that this violates the REST principle of statelessness. However, I am requiring you to do it here as this is an introductory server-side development module. In the real world you would probably use something like OAuth2 for authentication but this is beyond the scope of this module. However it is something you might want to investigate for your final-year project!)*

***If you get this far, you will receive a B2***

**Part F – Implementing a review system**

12. Add an additional endpoint to your REST API allowing clients to review a POI. It should read in the POI ID and review as POST data, and must check that the ID exists in the database, the review is not blank, and that the user is logged in.You must return appropriate HTTP status codes if any of these checks fail.

***No additional credit for this task, as it uses similar techniques to earlier tasks.***

13. Add a review box to the popup from Task 8, allowing the user to enter a review. When the user enters a review, it must be sent, along with the POI ID, to the REST API (task 12). You must check for errors returned from the REST API, and display them to a user in a user-friendly way.

***If you get this far, you will achieve a B1***

**Part G – Using React**

14. Use React to implement the front-end. (Note that if you start the assignment after we have covered React, you may use React for Part B without having to develop a non-React solution in addition).

You must not use any additional libraries or frameworks besides React, though. In particular you must implement your AJAX using the fetch API.

***If you get this far, you will achieve A4***

**Part H – Improving your answer**

To get a Grade A you should enhance your answer, making use of more advanced topics from the module in your answer, for example some or all of the following:

- creating a well-structured Node application with controllers, DAOs, routers and custom middleware, and using ECMAScript 6 classes;

- adding a file-upload facility, to upload a photo of the point of interest. Both client and server components are required for this. The photo must then appear on the popup in the front-end when the user clicks the appropriate marker. Only logged-in users should be able to upload a photo.

- creating a professional-looking site, including handling errors in a user-friendly, “real-world” way.

**Marking Scheme**

See also the Task Detail, above, for more details.

|  |  |  |
| --- | --- | --- |
| **Grade** | **CODE (60%)** | **REPORT (40%)** |
| F | Standard for Grade D not reached | Standard for Grade D not reached |
| D | Code complete and working for tasks 1-6 (also task 7 for a high D) | Clearly-written report covering tasks 1-6 (also task 7 for a high D) |
| C | In addition, task 8 should be fully complete (for low C) and task 9 (for high C) | In addition to the tasks for grade D, the report clearly covers tasks 8 (also task 9 for a high C) |
| B | In addition, task 10 completed (some or all of tasks 11-13 for a higher B) . | In addition, report covers task 10 completed (some or all of task 11-13 for a high B) |
| A | Task 14 completed. For a grade of A3 or above, your answer must be enhanced using some or all of the suggestions in Part H. | Task 14 covered. For a grade of A3 or above, your report must cover some or all of the suggestions in Part H. |

**Handing in the assignment**

For the hand-in, a ZIP file of your code and report should be handed in (uploaded to Solent Online Learning) by the date on the front sheet.

It must be easily runnable and testable by the tutor. If you use third party node modules, ensure these are present in your package.json so that they will be installed successfully with npm install.

If the app will not build or run, we will NOT, under ANY CIRCUMSTANCES, attempt to correct either your code or your configuration files to make it run, and you risk losing significant marks, or even failing, if this occurs.

Bottom line – it MUST run in a standard Node 14.x environment. I will be testing on a Linux machine running Node 14.x, so it must be runnable in that environment.

**ACADEMIC MISCONDUCT - IMPORTANT**

We carefully scrutinise assignment hand-ins and will be on the lookout for any form of academic misconduct. Academic misconduct includes, amongst other things:

a) **Collusion**: two or more students working together on an assignment when it is supposed to be an individual piece of work. Do not be tempted to seek, or give, too much help to a colleague - even if they are your friend. **It is likely to get both of you into trouble**.

b) **Commissioning a third-party to write the assignment for you**.

This module operates a zero-tolerance policy to academic misconduct so please do NOT attempt either of these, or any other form of academic misconduct - it will most likely be detected. It is not fair on students who do the work themselves if you attempt to dishonestly pass, or get a high grade, in the module

# Learning Outcomes

This assessment will enable students to demonstrate in full or in part the learning outcomes identified in the unit descriptors.

**Living CV**

As part of the University's Work Ready, Future Ready strategy, you will be expected to build a professional, Living CV as you successfully engage and pass each module of your degree.

The Living CV outputs evidenced on completion of this assessment are:

1. A fully-functional web based application making use of Node, Express and client-side JavaScript. This can be published to GitHub for others to view.

2. A technical discussion of the application mentioned above.

Please add these to your CV via the Living CV builder platform on Solent Futures Online [Solent Futures Online](https://eur03.safelinks.protection.outlook.com/?url=https%3A%2F%2Fsolentfutures.careercentre.me%2Fprogrammes%2F%3FprogrammeID%3DThzJ%252bRbk%252bQXoSlEaujPR0g%253d%253d&data=04|01|ian.harris@solent.ac.uk|f1bda34c4d564e82f6cb08da067fdf48|d684e4cd491a4577bf33546478d72e3c|0|0|637829443517919744|Unknown|TWFpbGZsb3d8eyJWIjoiMC4wLjAwMDAiLCJQIjoiV2luMzIiLCJBTiI6Ik1haWwiLCJXVCI6Mn0%3D|3000&sdata=ObCFbM3zY7CgU6SVNtitaq1udg0%2Bzlp1GuCAJ1y1utw%3D&reserved=0)

# Late Submissions

You are reminded that:

1. If this assessment is submitted late i.e. within 7 calendar days of the submission deadline, the mark will be capped at 40% if a pass mark is achieved;
2. If this assessment is submitted later than 7 calendar days after the submission deadline, the work will be regarded as a non-submission and will be awarded a zero;
3. If this assessment is being submitted as a referred piece of work, then it must be submitted by the deadline date; any Refer assessment submitted late will be regarded as a non-submission and will be awarded a zero.

[Assessment regulations](https://www.solent.ac.uk/about/documents/assessment-regulations.pdf)

# Extenuating Circumstances

The University’s Extenuating Circumstances (EC) procedure is in place if there are genuine short term exceptional circumstances that may prevent you submitting an assessment. If you are not 'fit to study’, you can either request an extension to the submission deadline of 7 calendar days or you can request to submit the assessment at the next opportunity, i.e. the resit period (as a Defer without capping of the grade). In both instances you must submit an EC application with relevant evidence. If accepted under the university regulations there will be no academic penalty for late submission or non-submission dependent on what is requested. You are reminded that EC covers only short term issues (20 working days) and that if you experience longer term matters that impact on your learning then you must contact the Student Hub for advice.

Please find a link to the EC policy below:

[Extenuating Circumstances](https://students.solent.ac.uk/official-documents/quality-management/academic-handbook/2p-extenuating-circumstances.pdf)

# Academic Misconduct

Any submission must be your own work and, where facts or ideas have been used from other sources, these sources must be appropriately referenced. The University’s Academic Handbook includes the definitions of all practices that will be deemed to constitute academic misconduct. You should check this link before submitting your work.

Procedures relating to student academic misconduct are given below:

[Academic Misconduct](https://students.solent.ac.uk/official-documents/quality-management/academic-handbook/4l-student-academic-misconduct-procedure.pdf)

**Ethics Policy**

The work being carried out must be in compliance with the university Ethics Policy. Where there is an ethical issue, as specified within the Ethics Policy, then you will need an ethics release or ethics approval prior to the start of the project.

The Ethics Policy is contained within Section 2S of the Academic Handbook:

[Ethics Policy](https://staff.solent.ac.uk/official-documents/quality-management/academic-handbook/2s-solent-university-ethics-policy.pdf)

**Grade marking**

The University uses an alpha numeric grade scale for the marking of assessments. Unless you have been specifically informed otherwise your marked assignment will be awarded a letter/number grade. More detailed information on grade marking and the grade scale can be found on the portal and in the Student Handbook.

[Grade Marking Scale](https://staff.solent.ac.uk/official-documents/quality-management/academic-handbook/2o-assessment-regulations-annex-1-grade-marking-scale.pdf)

**Guidance for online submission through Solent Online Learning (SOL)**

[Online Submission](http://learn.solent.ac.uk/onlinesubmission)