# Solent University

# Coursework Assessment Brief

# Assessment Details

|  |  |
| --- | --- |
| Module Title: | Web Application Development (Level 5) |
| Module Code: | QHO540 |
| Module Leader: | Dr. Marwan Radwan |
| Level: | 5 |
| Assessment Title: | Book2Dine (Restaurant Booking Application) |
| Assessment Number: | AE1 |
| Assessment Type: | Software Product |
| Restrictions on Time/Word Count: | 2500 words |
| 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: | October 2024 |
| Hand In Date: | 7th of February 2025 by 4pm (UK time) |
| Planned Feedback Date: | 20 working days after hand in. |
| Mode of Submission: | On-line  Only FINAL submissions will be accepted. DRAFT submissions will not be considered an attempt and will not be marked. |
| Anonymous Marking | This assessment is exempt from anonymous marking. |

**QHO540 Web Application Development**

**Assignment 1**

**Scenario**

Your task is to develop Book2Dine, an online web site which allows users to look up information on restaurants they might want to dine in. The application offers booking online for a number of guests in a particular restaurant.

You are required to build this Book2Dine web application according to the specification below.

**You must use Node and Express as the back-end technology, and SQLite for the database.**

**Database**

The database should be built according to the structure described below. It is a simplest-possible schema to allow you to develop the application. In your implementation, you may, if you wish, choose to use additional database tables. If you do, they must be documented, with justification.

You will be provided with a .db file containing the database and populated with some sample data.

The database structure is as follows:

*Restaurants Table - stores main information about the restaurant.*

|  |  |  |
| --- | --- | --- |
| Column | Type | Role |
| id | INTEGER | A unique identifier for each record in the table (i.e. the primary key) |
| **name** | TEXT | the restaurant’s name |
| **type** | TEXT | the restaurant’s type (e.g. such as Indian, Chinese, Turkish, Vegetarian, …) |
| **location** | TEXT | the restaurant’s location. You can be flexible on this, it can either be a city (e.g Southampton, London) or a region (e.g. Hampshire, Normandy, Colorado) |
| **latitude** | REAL | the restaurant’s latitude |
| **longitude** | REAL | the restaurant’s longitude |
| **description** | TEXT | The Restaurant Description |

*Availability Table – stores availability at the restaurant on the given dates.*

|  |  |  |
| --- | --- | --- |
| *Column* | Type | Role |
| id | INTEGER | an auto-incrementing numerical ID which references each booking record. It is the primary key of the table, and automatically increases by 1 for each record. |
| **restID** | INTEGER | the ID of the restaurant that this booking entry relates to; |
| **theDate** | INTEGER | the date (for simplicity the date is stored in a simple numeric format YYMMDD e.g. 241101 for November 1st 2024) |
| **maxPlaces** | INTEGER | how many dining places are currently available at that particular restaurant on that date (theDate). |

*bookings - stores records of the restaurant bookings*

|  |  |  |
| --- | --- | --- |
| **Column** | Type | Role |
| id | INTEGER | an auto-incrementing numerical ID which references each booking. It is the primary key of the table, and automatically increases by 1 for each booking. |
| **restID** | INTEGER | the ID of the restaurant that this entry relates to; |
| **theDate** | INTEGER | the date (for simplicity the date is stored in a simple numeric format YYMMDD e.g. 241101 for November 1st 2024) |
| **username** | TEXT | the user who made the current booking. |
| **diners** | INTEGER | number of people the booking is for. |

*users – a record of users*

|  |  |  |
| --- | --- | --- |
| **Column** | Type | Role |
| id | INTEGER | an auto-incrementing numerical ID which references each user. |
| **username** | TEXT | the username |
| **password** | TEXT | the password |
| **mobile** | TEXT | mobile number |
| **isAdmin** | INTEGER | Is the user an admin? (0=no, 1=yes) |

**Completing the assignment**

Your submission must include:

**a) A report (40%)**

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

**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**

***First – please use some CSS!***

Please do not simply hand in a website with no CSS applied, e.g. black text on a white background. Please include at least a small amount of CSS (e.g. non-default background colour and font, and a simple layout). If you do not, you will drop up to three grades (e.g. A4 to B3) for the coding criterion. It does not have to be complicated, but there needs to be a little custom styling present.

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

You should first develop a simple JSON-based REST web API using Node.js and Express which allows clients to:

1. Look up all restaurants in a given location (e.g. all restaurants in Colorado).

2. Look up all restaurants of a given type in a given location (e.g. all Turkish restaurants in Hampshire).

3. Book in a certain restaurant for a given number of people on a given date. The API should expect the restaurant ID, the number of people, and the date. You must add a record to the *bookings* table and reduce the availability from the maxPlaces field in the *availability* table accordingly.

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

Tasks 1, 2 and 3 involve creating a Web API using Node and Express. No front-end code is needed at this point.

***If you get this far, you will achieve a F2 (20)***

**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**).

*If you intend to use React (see Part G, below), there is no need to develop non-React versions of these tasks.*

4. Write an HTML page (which MUST be named **index.html**) which allows the user to search for all restaurants in a given location. The user should be able to enter a location, and then, using JavaScript, the page should communicate with your REST API to find all restaurants in that region. The JSON must be parsed, and the results presented to the user in a user-friendly way.

5. Modify your code to process the search results, so that you create a “Book” 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 book the restaurant. For now, hard-code the number of people to two, and the date to any of the available dates in the availability table in the database. In other words, the user does not have to enter the number of people or the date at this stage.

***If you get this far, you will achieve a D2 (45).***

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

6. Add error-checking to task 3, so that if any of the details (restID, number of people and date) are blank, an appropriate HTTP error code is sent back to the client. Then, modify task 5 to test for the HTTP code returned from the server and display an appropriate user-friendly message to the user, which will be understood by a non-technical user.

***If you get this far, you will achieve a D1 (48).***

**Part D – Adding a map**

7. 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 restaurant name and type should appear as a popup.

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

*If you intend to use React (see Part G, below), there is no need to develop a non-React version of this task.*

***If you get this far, you will achieve a C3 (52).***

**Part E – logins and sessions**

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

**Logged in as Kate**

There is no need to implement a signup facility, as the SQL db file is already populated with some existing users.

***If you get this far, you will receive a C1 (58).***

9. Change task 3 so that a user must be logged-in to book in a restaurant, sending back an appropriate HTTP error if they are not. Also change task 5 so that this error is checked, and an appropriate user-friendly error message displayed to the user if they are not logged in.

*If you intend to use React (see Part G, below), there is no need to develop a non-React version of this task.*

*(Note – you may realise that Task 9 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 use a technology such as 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 B3 (62).***

**Part F – More advanced tasks**

*If you intend to use React (see Part G, below), there is no need to develop non-React versions of these tasks.*

10. Modify task 3 to check that there is availability at that restaurant for that number of people on the specified date. Return an appropriate HTTP error code if there is no availability or if the date is in the past , and include the specific error within the JSON message sent back. Also, again modify task 5 to test the HTTP code returned from the server and display an appropriate user-friendly message to the user.

***If you get this far, you will achieve a B2 (65).***

11. Add a booking button to your popup from Task 7 to allow the user to book a table in a restaurant On this popup, the user must be able to enter the date and the number of people. You should use a user-friendly approach to picking the date from the available dates in the availability table. When the button is clicked, the data should be sent over to appropriate routes on your REST API. Check for each type of error returned from the REST API and communicate these to the user in a user-friendly way.

***If you get this far, you will achieve a B1 (68).***

**You will notice that completion of the above tasks will obtain a maximum grade of B1. To increase your grade further, you must complete Part G and/or Part H, below. If you complete Part G and/or H without completing all earlier tasks, you can still obtain extra credit if completed to a high-enough standard.**

**Part G – React**

Use of React components to implement Tasks 4, 5, 7, 9, 10 and 11 will increase your grade (up to a maximum of 2 grades) if implemented to a sufficiently high standard.

**Part H – Middleware, DAOs, controllers and routers**

Extensive use of middleware, DAOs, controllers and routers on your Express server will increase your grade (up to a maximum of 2 grades) if implemented to a sufficiently high standard.

**Marking Scheme**

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

**You may drop up to three grades if you do not include adequate CSS. See above for more detail.**

|  |  |  |
| --- | --- | --- |
| **Grade** | **TASK COMPLETION (60%)** | **REPORT (40%)** |
| F | Standard for Grade D not reached | Standard for Grade D not reached |
| D | Code complete and working for tasks 1-5 (also task 6 for a high D) | Clearly-written report covering tasks 1-5 (also task 6 for a high D) |
| C | In addition, task 7 should be fully complete (for low C) and task 8 (for high C) | In addition to the tasks for grade D, the report clearly covers tasks 7 (also task 8 for a high C) |
| B | In addition, task 9 completed (and tasks 10/11 for a medium B or upwards) . | In addition, report clearly covers task 9 (and task 10/11 for a medium B or upwards) |
| A | In addition, Part G and/or H completed. | In addition, report clearly covers Part G and/or H. |

**IMPORTANT NOTE – ChatGPT (and similar software) PROHIBITED**

Please note that ALL use of ChatGPT or similar generative AI in relation to this assignment is **absolutely prohibited**. This includes not only using such software to write some or all of the assessment (code and/or report) for you, but also using such software for guidance on how to write code that might be useful for the assessment.

If evidence of use of generative AI is found in your answer, we will refer your work to an academic misconduct panel.

**Preparing the assignment for hand-in – important instructions**

Your application must be easily runnable and testable by the tutor. It must run on Port 3000.

You must name your main HTML page **index.html**. So, entering the URL

[http://localhost:3000](http://localhost:3000/)

in a browser MUST load the application’s main page, and it MUST be possible to navigate to all functionality from there. If the assessor encounters a 404 or “unable to connect” error when requesting the above URL, YOU WILL RECEIVE ZERO FOR THE CODE.

Any work not accessible from the main page will NOT BE MARKED!

Ensure your Node modules are present in your package.json so that they will be installed successfully with npm install.

If you use Webpack, you must provide an appropriate webpack.config.js and include a build script in your package.json to run Webpack.

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 if this occurs.

Bottom line – it MUST run in a standard Node 20.x environment. I will be testing on a Linux machine running Node 20.x, so it must be runnable in that environment.

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

**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) **Use of generative AI:** see above.

b) **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**.

c) **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 any 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.

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)