# 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: | Places To Stay 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: | 27 January 2022 |
| Hand In Date: | 29 April 2022, 16:00 |
| Planned Feedback Date: | 20 working days after hand in. |
| Mode of Submission: | via SOL |
| Anonymous Marking | This assessment **is** exempt from anonymous marking. |

**COM518 Web Application Development**

**Assignment 1**

**Scenario**

Your task is to develop PlacesToStay, a site which allows users to look up information on places they might want to stay whilst away, such as hotels, bed and breakfasts, and hostels.

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

**Database**

You should use database 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 an SQL file to pre-populate the database.

The database structure is as follows:

*accommodation*

**ID** – an auto-incrementing numerical ID which references each place to stay. It is the primary key of the table, and automatically increases by 1 for each item of accommodation. So the first accommodation will have ID 1, the second, ID 2, and so on.

**name** VARCHAR(255) – the accommodation’s name;

**type** VARCHAR(255) – the accommodation’s type (e.g. hotel, bed and breakfast, hostel)

**location** VARCHAR(255) – the accommodation’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** FLOAT – the accommodation’s latitude;

**longitude** FLOAT – the accommodation’s longitude;

*acc\_dates - stores availability at accommodation on given dates.*

**ID** – an auto-incrementing numerical ID which references each record. It is the primary key of the table, and automatically increases by 1 for each record. So the first record will have ID 1, the second, ID 2, and so on.

**accID** INT - the ID of the accommodation that this entry relates to;

**thedate** INT - the date (for simplicity the date is stored in a simple numeric format YYMMDD e.g. 180630 for June 30th 2018)

**availability** INT – how many places are currently available at that accommodation on that date

*acc\_bookings - a record of bookings*

**ID** – 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. So the first record will have ID 1, the second, ID 2, and so on.

**accID** INT - the ID of the accommodation that this entry relates to;

**thedate** INT - the date (for simplicity the date is stored in a simple numeric format YYMMDD e.g. 180630 for June 30th 2018)

**username** VARCHAR(255) - the user who made the booking.

***npeople*** *- number of people the booking is for*

**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 on average (guidance only); 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 JSON-based REST web API using Node.js and Express which allows clients to:

1. Look up all accommodation in a given location.

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

3. Book a place of accommodation for a given number of people on a given date. The API should expect the accommodation ID, the number of people, and the date. You must add a record to the *acc\_bookings* table and reduce the availability in the *acc\_dates* table. At this point, you do not need to check whether the accommodation is full on that date. ***Please note: assume that it is a simplified system in which you do not have to worry about room allocation, differences between adults and children, taking payment, etc.***

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***

**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).

4. Write an HTML page which allows the user to search for all accommodation 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 points of interest 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 accommodation. For now, hard-code the number of people to one, and the date to any of the available dates 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.***

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

.

7. Add error-checking to task 3, so that if any of the details (ID, number of people, 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.***

**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 accommodation 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***

**Part E – logins and sessions**

9. 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 jsmith

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 C1***

10. Change task 3 so that a user must be logged-in to book accommodation, 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.

*(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 B3***

**Part F – More advanced tasks**

11. Modify task 3 to check that there is availability at that accommodation 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***

12. Add a booking button to your popup from Task 8 to allow the user to book that item of accommodation. 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 a date. When the button is clicked, the data should be sent over to your REST API (tasks 3, 7 and 11). 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 an A4***

**Part G – Improving your answer**

To get a higher A (A3, A2 or A1) 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 (the more of these you implement, the higher will be your grade):

- creating a well-structured Node application with middleware, DAOs, controllers and routes;

- using Passport for authentication rather than express-session;

- add a file-upload facility, to upload a photo of the accommodation. 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.

- add a mock ‘payment’ system. A user should enter a credit card which should be validated by sending to a credit card validator API (you should create this yourself server-side; see <https://www.npmjs.com/package/card-validator>). The booking must only be completed if the user supplies a valid credit card.

**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 (and task 11 for a medium B or upwards) . | In addition, report clearly covers task 10 (and task 11 for a medium B or upwards) |
| A | Task 12 completed. For a higher A, your answer must be enhanced using some or all of the suggestions in Part G. | In addition, report clearly covers Tasks 12. For a higher A, report covers some or all of the suggestions in Part G. |

**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 if this occurs.

Bottom line – it MUST run in a standard Node 16.x environment. I will be testing on a Linux machine running Node 16.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 Module descriptors.

**Late Submissions**

Students are reminded that:

1. If this assessment is submitted late i.e. within 5 working 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 5 working 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.

<https://students.solent.ac.uk/official-documents/quality-management/academic-handbook/2o-assessment-principles-regulations-temporary-amendments-for-covid-19-contingency-plans.pdf>

# Extenuating Circumstances

The University’s Extenuating Circumstances procedure is in place if there are genuine circumstances that may prevent a student submitting an assessment. If students are not 'fit to study’, they can either request an extension to the submission deadline of 5 working days or they can request to submit the assessment at the next opportunity (Defer). In both instances students must submit an EC application with relevant evidence. If accepted by the EC Panel there will be no academic penalty for late submission or non-submission dependent on what is requested. Students are reminded that EC covers only short term issues (20 working days) and that if they experience longer term matters that impact on learning then they must contact the Student Hub for advice.

Please find a link to the EC policy below:

<https://students.solent.ac.uk/official-documents/quality-management/academic-handbook/2p-extenuating-circumstances.pdf>

# Academic Misconduct

Any submission must be students’ 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. Students should check this link before submitting their work.

Procedures relating to student academic misconduct are given below:

<https://students.solent.ac.uk/official-documents/quality-management/academic-handbook/4l-student-academic-misconduct-procedure.pdf>

**Ethics Policy**

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

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

<https://staff.solent.ac.uk/official-documents/quality-management/academic-handbook/2s-solent-university-ethics-policy.pdf>

**Grade marking**

The University uses a letter grade scale for the marking of assessments. Unless students have been specifically informed otherwise their marked assignment will be awarded a letter grade. More detailed information on grade marking and the grade scale can be found on the portal and in the Student Handbook.

<https://students.solent.ac.uk/official-documents/quality-management/academic-handbook/2o-annex-3-assessment-regulations-grade-marking-scale.docx>

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

<http://learn.solent.ac.uk/onlinesubmission>