# Southampton Solent University

# Coursework Assessment Brief

# Assessment Details

|  |  |
| --- | --- |
| Unit Title: | Web Application Development |
| Unit Code: | SWD601 |
| Unit Leader: | Nick Whitelegg |
| Level: | 6 |
| Assessment Title: | Places To Stay – Code |
| Assessment Number: | AE1 |
| Assessment Type: | Report |
| Restrictions on Time/Word Count: | none |
| Consequence of not meeting word count limit: | There is no penalty for submitting below the word/count limit, but students should be aware that there is a risk they may not maximise their potential mark.  There is no penalty for exceeding the word count |
| Individual/Group: | Individual |
| Assessment Weighting: | 60% |
| Issue Date: | 2/10/17 |
| Hand In Date: | 12/1/18 |
| Planned Feedback Date: | February 2018; it is hoped to complete marking by 22/1/18 |
| Mode of Submission: | Online |
| Number of copies to be submitted: | 1 |
| Anonymous Marking | No |

**Introduction**

**Important!**

There are two assignments for WAD, a software development assignment (this assignment) and a report. They both relate to the same scenario, described below.

**Scenario**

PlacesToStay is 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. At the moment the site is an ordinary database-driven website which allows users to search for and book accommodation.

However the site still believes it can go further, and would like to expand its system to take advantage of distributed and mobile web technologies and the collaborative nature of the contemporary Web. In order to expand the ways in which clients can interact with the system, they wish to make the system functionality available as a web service.

**Database**

A basic database has been set up on the Technology web server Edward2 (or another server if advised) to store places to stay. In your implementation, you may choose to use additional database tables. If you do, they must be documented, with justification.

This database contains the following tables:

*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 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

**Terminology - use of the word "client"**

***Throughout this brief, the word "client" refers to client software. It does not refer to a human client, such as a customer.***

**Tasks**

Implement these tasks using the technologies learnt in the unit. Tasks 1-8 are the same as those in the report. For tasks 1-8, you should use your report to help you implement the task. Tasks 9-11, necessary to achieve high A grades, are unrelated to the report.

***Tasks related to the report***

1. Create a web service to allow clients to look up all accommodation in a given location; all accommodation of a given type; and both. **(Hint: Use “isset” to check which query string parameters are set).**
2. Create a web service to allow clients to book a place of accommodation for a given number of people on a given date. This should communicate its success or otherwise to its clients appropriately. ***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.***
3. Create an AJAX front end which connects to the web service, and in doing so, allows users to look up all places to stay in a given location. Results must be presented in a user-friendly manner.
4. Create a third-party website, VisitHampshire, which is a tourist information site for Hampshire. It should use the PlacesToStay web service rather than its own database. It should allow its own users to look up places to stay of a given type within Hampshire, making use of the web service. Results must be presented in a user-friendly manner.
5. PlacesToStay's own site (not VisitHampshire!) should include web maps (e.g. Leaflet, Google Maps or similar**). AS A RESULT OF THE RECENT CHANGES IN WHICH HTTPS IS NOW REQUIRED FOR GEOLOCATION, YOU DO NOT HAVE TO IMPLEMENT GEOLOCATION CODE**.
6. On PlacesToStay's own site, extend the AJAX search (task 3) so that all items of accommodation in that location will be shown on the map as markers, as well as shown as text search results.
7. VisitHampshire should allow a user to book an item of accommodation on a given date, by interacting with the PlacesToStay web service. This task should be linked with task 4. Ideally, the user should be able to specify the number of people; if not, assume one.
8. When the user clicks on each marker, a popup describing that item of accommodation should appear. The popup should have a "Book" link allowing the user to book the item of accommodation. When the user clicks "Book" they should enter a date (plus ideally the number of people, if not, assume one) and the item should be booked for that date. The user should be provided with feedback as to whether the booking was successful or not. The front-end must fully use JavaScript and AJAX without any page refresh required.

***Further tasks***

Choose ONE OF 9), 10) or 11). Do not attempt more than one of these tasks. It is NOT possible to get extra marks this way.

1. Create a new "Check Availability" link on the popup from Task 8. When the user clicks on this link, a graphical view should appear, showing which dates are available, ideally with a colour scheme such as red=booked, green=available. This need not be arranged by month; it can simply be an arbitrary 20 days sometime in the future (e.g. 1-20 June 2018). The server-side component of this task must be written as a web service and the front-end must make use of HTML5 <canvas> (not just <div>s and CSS, or any jQuery UI components or similar) and fully use JavaScript and AJAX without any page refresh required. For an A1 it should then be possible to book a date by clicking on it.
2. Use WebSocket to implement a system in which the map updates every time a new place to stay is added, as follows:

* First of all implement a new web service to add a place to stay, and connect this to your AJAX map so that a user can add a place by clicking on the map and implementing the details;
* Secondly implement functionality so that if a user adds a place to stay, the AJAX map of all other users currently connected is automatically updated, via a push notification with WebSocket, to show the new place.

1. Implement your web services using node-js and MongoDB. Ensure that your AJAX front-end fully communicates with your node-js and MongoDB back-end. Ensure you fully handle errors both client-side and server-side.

**Marking Scheme**

|  |  |  |
| --- | --- | --- |
| **Grade** | **BREADTH/TECHNICAL COMPLEXITY- successful implementation of tasks (70%)** | **QUALITY - good programming practice, extent of error handling, advanced features in code (30%)** |
| F | Standard for Grade D not reached | Standard for Grade D not reached |
| D | Tasks 1 to 4 implemented correctly; task 5 should be implemented for higher Ds. | Appropriate use of HTTP GET and POST. Web services recognise and handle errors which are likely to occur. Appropriate techniques for error handling used in your web services. |
| C | In addition, task 6 completed. | In addition to the tasks for grade D, appropriate use of HTTP codes by your web services |
| B | In addition, task 7 completed. | In addition, a suitable authentication mechanism is present in your web services (where appropriate). Errors fully handled client-side with user-friendly error messages, integrated into the interface in a user-friendly way. |
| A | Task 8 completed. For higher As, you should in addition complete EITHER Task 9 OR Task 10. | In addition, your answer appropriately uses some of the more advanced techniques covered during the unit and/or additional advanced features gained from further research; events handled with unobtrusive JavaScript. Good quality JavaScript code. |

**Handing in the assignment**

A ZIP file of your code should be handed in (uploaded to SOL) by 16:00 (4pm) on Friday 12/1/18. Your code should also be uploaded to your space on the Edward, Edward2 or Neptune server (to be advised) by the same time. Please note that, while late submissions are accepted subject to the University's rules, help and assistance on the assignment by the tutors will only be given up to the official hand-in date.

You must provide an HTML index page on the server with links to all of your working code. This must be saved as **index.html** in your **public\_html** directory on the server. It must be obvious to the marker(s) (which might include external examiners) to run your code; for instance, if you have a web service, you must document on the index.html page what query string parameters are required for testing. Any code that is not linked to from this index.html page, or any code which I cannot test, will not be marked. I will not look at your source code to discover how to test your work!

If doing the node.js task, it is not necessary to start your node.js server; however please give me instructions on your index.html page as to how it can be started.

**Other info**

This assessment will enable students to demonstrate in full or in part the learning outcomes identified in the unit 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 (second or third attempt) 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.

<http://portal.solent.ac.uk/documents/academic-services/academic-handbook/section-2/2o-assessment-policy-annex-1-assessment-regulations.pdf?t=1411116004479>

# 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 a Student Achievement Officer for advice.

A summary of guidance notes for students is given below:

<http://portal.solent.ac.uk/documents/academic-services/academic-handbook/section-4/4p-extenuating-circumstances-procedures-for-students.pdf?t=1472716668952>

# 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:

<http://portal.solent.ac.uk/support/official-documents/information-for-students/complaints-conduct/student-academic-misconduct.aspx>

**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:

<http://portal.solent.ac.uk/documents/academic-services/academic-handbook/section-2/2s-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.

Policy: <http://portal.solent.ac.uk/documents/academic-services/academic-handbook/section-2/2o-assessment-policy.pdf>

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

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