Southampton Solent University
Coursework Assessment Brief

Assessment Details

Module Title:	Web Application Development
Module Code:	QHO540
Module Leader:	Muhammad Ibrahim
Level:	5
Assessment Title:	Event Ease (Event Booking System)
Assessment Number:	1
Assessment Type:	Software Development with Report
Restrictions on Time/Word Count:	2000-3000 words (for guidance only)
Consequence of not meeting time/word count limit:	There is no penalty for submitting below the word limit, but you should be aware that there is a risk you may not maximise your potential mark due to lack of content. There is no penalty for submitting above the word count but it should not be necessary to do so.
Individual/Group:	Individual
If a group	N/A
Assessment Weighting:	50%
Issue Date:	Friday 28 February 2025
Hand In Date:	Friday 06 June 2025 16:00
Planned Feedback Date:	20 working days after the above
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 will use anonymous marking. Please avoid adding your name or email address to your answer. Use your student number instead.

Assessment Task

Scenario

Your task is to develop **EventEase**, an **event booking platform** that allows users to search for events in different locations, view event details, and book tickets for available events. The system will manage event listings, available seats, and reservations.

You are required to build EventEase 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 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 containing initial data.

events – Stores information about available events

Column	Type	Role
id	INTEGER, PRIMARY KEY	An auto-incrementing index uniquely identifying each record
name	TEXT	The event's name
category	TEXT	The event's category(e.g.,concert,workshop,sports,conference)
location	TEXT	The city or venue of the event
date	TEXT	The event date (stored in YYMMDD format, e.g., 250301 for March 1, 2025).
lon	FLOAT	the longitude of the POI
lat	FLOAT	the latitude of the POI
description	TEXT	the event description

tickets - Stores ticket availability for an event.

Column	Type	Role
id	INTEGER, PRIMARY KEY	An auto-incrementing index uniquely identifying each ticket
eventID	INTEGER, Foreign Key	The ID of the event this ticket belongs to.
ticketType	TEXT	The type of the ticket booked(General,VIP,Student)
price	REAL	Cost per Ticket
availability	INTEGER	No. of available ticket

bookings - stores user bookings.

Column	Type	Role
id	INTEGER, PRIMARY KEY	An auto-incrementing index uniquely identifying each booking
eventID	INTEGER, FOREIGN KEY	The event ID being booked
ticketType	TEXT	The type of ticket booked.
username	TEXT	The user who made the booking.
quantity	INTEGER	Number of tickets booked.

users – Stores registered users

Column	Type	Role
id	INTEGER, PRIMARY KEY	Auto-incrementing user ID
username	TEXT	The username
password	TEXT	password
isAdmin	INTEGER	0 = Regular User, 1 = Admin.

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.

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 - Security issues

Ensure your code is protected against common web security exploits and weaknesses, as examined in the Security Issues session (Topic 11). If you have not adequately secured your code, you may lose up to three grade points (e.g. 68 to 58, 100 to 74) from your coding grade.

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 events** in a given **location**. It should return the results as JSON.

✓ 2. Look up **available tickets** for an event and their **prices**.

✓ 3. Book **tickets** for an event. The API should expect:

- Event ID, ticket type, quantity, and username.
- Reduce the **availability** in the **Tickets table** accordingly.

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

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

✓ ***Before you begin part B, you should decide whether to use React or not. If you choose React, you do not need to build a non-React front-end for any of your tasks. React is more difficult but will allow you to obtain extra credit - see Part G below.***

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).

I used react

✓ 4. Next, you should build a simple HTML and JavaScript front-end that interacts with your REST API using AJAX, ensuring no page reloads occur. You must create an **index.html** file where users can enter a location to search for events. The page should display event names, descriptions, and available ticket types, communicating with the REST API to fetch and present results in a user-friendly manner.

all done in react however I added also date price and amount of available tickets

5. Modify your code to process the search results, so that you create a "Book Ticket" 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 book the ticket

done however i did it to book one ticket once as behavior is not specified

If you get this far, you will achieve 45%. ***<u>This is the standard to pass.</u>***

Part C - Adding simple error-checking

6. Add error-checking to task 2, so that If any booking details are missing, 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 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 event details should appear as a popup.
<u>You must use Leaflet and OpenStreetMap. In particular, Google Maps is NOT acceptable.</u>

If you get this far, you will achieve 52%

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 implement a logout facility. <u>Ensure the logged-in message still appears if the user reloads the page</u>. Additionally, create a signup route on your Express server to add users. <u>There is no need to implement a signup form, however.</u>

If you get this far, you will receive 62%

11. Modify the booking functionality so that only logged-in users can make ticket reservations. If a user is not logged in, return an appropriate HTTP error message, and modify the front-end to handle this error gracefully.

(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 65%

Part F - Implementing Additional Features

12. Extend your application by modifying the booking functionality to ensure that users cannot book tickets for past events. If a booking request is made for an expired event or if tickets are sold out, return an appropriate message.

Additionally, enhance the map feature by adding a booking button inside the event popups, allowing users to select the ticket type and quantity before proceeding with the booking.

If you get this far, you will achieve 68%

You will notice that completion of the above tasks will obtain a maximum grade of 68%. 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 well-written React components to implement client-side code from Task 4 onwards will increase your grade (up to a maximum of 2 grade scale points, e.g. 68 to 83) if implemented to a sufficiently high standard.

Part H - Middleware, DAOs, controllers and routers

<u>Extensive</u> use of middleware, DAOs, controllers and routers on your Express server will increase your grade (up to a maximum of 2 grade scale points, as per Part G) if implemented to a sufficiently high standard.

Report

You must accompany your work with a report of around 1000-2000 words which should be written before you start coding.

This should include:

- a written analysis of the requirements of each task and what techniques, from the module's material, are needed to code the task. This should then lead into a technical discussion of how you would code the task, making use of the material learnt in class. Accompany this discussion with code extracts. This must be done before you code the corresponding task.
- a description of problems encountered and explanation of how you solved them. This should be done after you have coded the application.

For each task, the write-up will be between 250 and 500 words, the exact length depending on the complexity of the task.

If you are using React for the front end, you do not also have to discuss a nonReact implementation.

Important!

The report must not read like a specification of your code. It must clearly show your thought processes when analysing the requirements and writing the report; if it does not, you may receive a fail grade for the report component of the assessment.

You do not need to use formal analysis and design artefacts (use cases, sequence diagrams or class diagrams, for example).

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

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!

You must ensure your Node modules are present in your `package.json` so that they will be installed successfully with `npm install`. If you do not provide a `package.json`, your grade will be reduced by three grade points (e.g. 68 to 58, 100 to 74).

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 failing the assessment if this occurs.

Bottom line – it MUST run in a standard Node 20.x or 22.x environment. I will be testing on a Linux machine running Node 20.x or 22.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. **You must name this using your student number. For example if your student number is 1048576, then you would name your ZIP file 1048576.zip. Please do not use any other name - if you do, you will lose three grade points.**

Anonymous marking will be used for this assessment so please **do not leave your name, username or email address** (or any other personally-identifiable information) within your submission.

Assessment criteria

Grade	**TASK COMPLETION (60%)**	**REPORT (40%)**
<40	Tasks 1-5 not complete, or contain errors.	Tasks 1-5 not covered in report, or are incomplete or unclearly-written. Report written in a specification-like manner and/or lacking evidence of your thought processes.
40-49	Code complete and fully working for tasks 1-6 (also task 7 for a grade in the higher 40s)	Clearly-written report covering tasks 1-6 (also task 7 for high 40s)
50-59	In addition, task 8 should be fully complete (for lower 50s) and task 9 (for high 50s)	In addition to the above, the report clearly covers tasks 8 (also task 9 for high 50s)

60-69	In addition, task 10 completed (and task 11 for mid-60s and 12/13 for high 60s).	In addition, report clearly covers task 10 (and task 11 for medium 60s and 12/13 for high 60s).
70-100	In addition, Part G and/or H completed.	In addition, report clearly covers Part G and/or H.

Use of AI in this Assessment

Generative AI is permitted at Solent University under specific conditions and must continue to follow the university's rules around Academic Misconduct and the AI and Academic Integrity policy. However, in this assessment, as it is a software development assessment designed to test your software development abilities, **you are not allowed to use generative AI to create even small pieces of code or report content. You are however allowed to use AI to clarify the instructions on the assignment brief, as long as you declare it.**. This includes tools such as ChatGPT as well as GitHub Copilot or any tool which generates code for you within an IDE.

Any work found to be produced by generative AI will either be ignored and not marked (if referenced) or referred to an academic misconduct panel (if not referenced).

Learning Outcomes

This assessment will enable you to demonstrate in full or in part your fulfilment of the learning outcomes identified in the Module Descriptor.

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. An object-oriented analysis and design of a graphical Kotlin application. This can be published to GitHub for others to view.
2. A fully-functional object-oriented Kotlin GUI application using Compose Multiplatform. This can be published to GitHub for others to view.

Please add these to your CV via the Living CV builder platform on Solent Futures Online Solent Futures Online

Important Information

Solent University Academic Regulations 2024-25

Late Submissions

You are reminded that:

i. 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;
ii. 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;
iii. 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
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. You are able to self-certify for up to two assessment dates in any semester without supporting evidence for an extension of up to seven calendar days for coursework or to defer an exam to the resit period.

Alternatively, if you are not 'fit to study' (or you have used up your two selfcertification opportunities), you can request:

- an extension to the submission deadline of 7 calendar days, or
- a 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

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 Regulations 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

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

Grade marking

The University uses an numeric grade scale for the marking of assessments. More detailed information on grade marking and the grade scale can be found on the portal and in the Student Handbook.

Grade Marking Scale

Guidance for online submission through Solent Online Learning (SOL)

Online Submission