Department of Computer & Information Sciences

ASSESSMENT BRIEF	
Module Title:	Web Application Integration
Module Code:	KF6012
Academic Year / Semester:	2023-24 / Semester 1
Module Tutor / Email (all queries):	John Rooksby john.rooksby@northumbria.ac.uk
% Weighting (to overall module):	100%
Assessment Title:	Individual development project to create a web-based application.
Date of Handout to Students:	October 2023
Mechanism for Handout:	Module Blackboard Site & Seminar in Week 5
Deadline for Attempt Submission by Students:	18th January 2024 23.59
Mechanism for Submission:	Document upload to module Blackboard site
Submission Format / Word Count	Upload your work in a single zip folder to Blackboard. You must include all source files except the 'node_modules' React folder. Your work must also be deployed on a web server. Submissions for this module are not anonymous and your work should include your name.
Date by which Work, Feedback and Marks will be returned:	15th February 2024
Mechanism for return of Feedback and Marks:	Mark and individual written feedback sheet will be uploaded to the module site on Blackboard. For further queries please email module tutor.

LEARNING OUTCOMES

The learning outcomes (LOs) for this module are:-

MLO1: Knowledge & Understanding: You will be able to develop a web-based system using server-side and client-side languages, and appropriate methods including object-oriented programming and component-based design (KU1). You will be able to meet non-functional software quality requirements including for maintainability, security, robustness and scalability (KU2). You will have the ability to select and use appropriate tools for developing and testing web-based systems and APIs (KU3) and you will have an applied understanding of appropriate software architectures for web-based systems including n-tier and REST (KU1 & KU3).

MLO2: Intellectual / Professional skills & abilities: You will plan and manage a complex development project that produces a complete web application (IPSA1). You will be able to critically evaluate and apply tools and approaches for the project and choose which are appropriate for you (IPSA2 & IPSA7). You will be able to meet complex requirements and create a system that presents complex data in a way that is easy to interact with (IPSA3). You will draw from a range of classic and cutting-edge literature on a range of relevant topics including system architecture, interaction design, and Web APIs (IPSA4).

MLO3: Personal Values Attributes (Global / Cultural awareness, Ethics, Curiosity) (PVA): You will demonstrate professional and reflective practitioner attributes by managing your time and making choices in a complex development project including deciding and making trade offs between client side and server side features and architecture, and deciding, implementing and documenting your own Web API interfacing between client and server code (PVA4).

The coursework will consist of two parts. Part one will test your knowledge and understanding (MLO1) of server side technologies and approaches, including object oriented PHP. Part two will test your knowledge and understanding (MLO1) of client side development, including component based design in JavaScript. Parts one and two will test your intellectual and professional skills and abilities (MLO2) by requiring you to plan and manage a complex project, and for you to decide what tools and approaches you will use to build the finished product. Parts one and two will test your personal values attributes (MLO3) by requiring you to make trade offs and find balance between decisions regarding the client and server side, thus requiring some level of iterative development. You will also be asked to work with a real-world dataset.

This assessment addresses learning outcomes LO1, LO2, LO3.

Introduction

For this coursework you will build a web application that shows information about research that was presented in 2023 at an academic Human Computer Interaction conference called "The ACM CHI Conference on Human Factors in Computing Systems" or "CHI 2023". You will be given a database containing information about the 'research content' which consists of academic papers, work-in-progress reports, case studies and other items. Each item of content has a title, an abstract and other forms of information associated with it. Each item of content also has one or more authors and most of these authors are affiliated with one or more university or organisation.

This coursework consists of two parts, each worth 50 marks. For part 1 you will build a web API using object-oriented PHP. The web API must communicate with the database to output information about the research in JSON format. The web API will also enable users to authenticate and to create short notes about items of research. For part 2 you will build a web application using the JavaScript framework 'React' and the CSS framework 'Tailwind'. The web application must communicate with the web API to display information about the research, to allow users to authenticate and to store and retrieve notes.

For each part, requirements are given. You can go beyond these if you wish, for example by creating additional functionality for the web API and web application. There is also flexibility in how you can address many of the requirements. Note that you may need to add additional functionality to your web API to meet all requirements for part 2.

The conference data is contained in an SQLite database provided on Blackboard. The database file is called *chi2023.sqlite*. Further information about the database is given in appendix 2. You must not copy additional data from other sources such as the official conference website. A second database, *users.sqlite* with be used to store user information. Only *users.sqlite* can be modified by you.

Your work for part 1 and part 2 must be deployed on the nuwebspace server. You can use a folder structure and URL path that suits you. We suggest the following paths would be appropriate:

- https://[your-username].nuwebspace.co.uk/kf6012/coursework/api
- https://[your-username].nuwebspace.co.uk/kf6012/coursework/app

You must also submit the following on Blackboard in a zip file:

- A readme file containing the information specified in appendix 1
- A folder containing your work for part 1. This must include all code, the databases, .htaccess files, and all other necessary files needed to run your system.
- A folder containing your work for the part 2. This must include all code (not simply the build files) with the exception of 'node modules'.

Your work on this module will be marked using the grading criteria at the end of this document. When marking your work we make a decision about which grade category your work fits into. To fit into a grade category your work must meet everything in the descriptor for that grade category as well as satisfying each of the descriptors beneath it. Some categories have a range of marks and a judgement will be made by the marker where you work fits within these.

The use of appropriate generative AI tools is permitted for this coursework. You may use ChatGPT, Google Bard, Bing and Github CoPilot. Other similar generative AI tools may also be used with permission of the module leader. Use of these tools must be acknowledged within the source files.

Some students will be invited to attend a viva. You do not need to prepare a presentation for your viva but will be asked to demonstrate your work and explain parts of your code. If you fail to attend the viva you marks will be withheld and the viva rearranged.

Your work will be marked using Firefox, Chrome, Postman and Visual Studio Code.

Part 1: Web API

For part 1 you will use object-oriented PHP to implement a web API.

Essential requirements for part 1

(pass / fail)

Your work for part 1 must meet the following essential requirements. These essential requirements are pass/fail and do not have marks directly associated with them. These requirements are taken into consideration in the grade descriptors for the quality requirements and for tasks 1.1 and 1.2. If you fail to meet any of these essential requirements your marks for part 1 are unlikely to exceed 40% and may be zero.

Essential requirements for all tasks in part 1
• The web API must be deployed on the nuwebspace webserver. • The web API must be implemented using object-oriented PHP and must not use a third-party framework or library except the JWT library specified for task 1.2. • The coursework must use the *chi2023.sqlite* database provided and must not modify this. The *users.sqlite* database should also be used and can be modified. • Your work must include a readme file giving the full URL for every endpoint implemented and clearly listing what parameters are supported and how to use them (see appendix 1).

Quality requirements for part 1

(10 marks)

The following are quality requirements that your code for tasks 1.1 and 1.2 should meet. You will be graded for how well you meet these requirements.

Quality requirements for part 1
• You must use Object-Oriented PHP code that is well structured and easy to read. No specific design patterns or system architecture is required but your work should be well organised. • No files, classes, or sections of code should be included in your submission unless they have a purpose for your Web API. No code should be commented out. • You must use an appropriate coding style following the recommendations in PHP FIG PSR-1 and PSR-12 as well as the guidance on the module. • You must follow the PHP-FIG PSR-5 PHPDoc draft standard when commenting code. Tags from section 5 of PSR-19 should also be used in the comments where appropriate. You should include the @author tag followed by your name in every script you have written or modified. If you have used AI tools you must state this with use of the @generated tag followed by a short explanation. You do not need to produce any documentation for your code, just include comments using the 'doc comments' format. • Your code must make use of an autoloader.

- Your code must make use of an exception handler.
- The web API must use clean URLs. No API endpoint should require use of a .php or other file extension within the URL.
- An .htaccess file must be used to enforce the single point of entry pattern covered in the module. All requests must be handled by a single 'front door' script (typically called index.php or api.php). It should not be possible to directly access any other files or folders via HTTP requests.

Task 1.1

(20 marks)

For task 1.1 you will create a Web API with (at least) five endpoints. A list of general requirements for each endpoint is given below:

General requirements for task 1.1
• All responses from the API must contain relevant headers, including a relevant HTTP status code. • All responses from the API that contain data must return this in valid JSON format. • All endpoints created for this task must support the request method GET. • Requests to invalid endpoints, use of invalid parameters, invalid combinations of parameters, invalid values, server errors (e.g. database errors), or any other error should be responded to with a relevant HTTP status code and, where appropriate, an error message in JSON format. • Endpoints and parameter names and values should not be case sensitive. • Additional endpoints can be created if you wish. Additional parameters can be supported if you wish. The endpoints listed below can also return more data than is specified if you wish. Any additional endpoints, parameters and data must be appropriate to the purposes of the API. • All endpoints except endpoint 1 must retrieve data from the *chi2023.sqlite* database. • The name of the endpoint is what should appear in the URL path, for example the endpoint 'developer' might have the URL https://nuwebspace.co.uk/kf6012/coursework/api/developer • The full URL for each endpoint should be given in the readme file. For each parameter supported there should also be a clear example (see appendix 1). • An endpoint should work whether or not there is a trailing slash at the end of the URL.

Requirements for the five endpoints you should implement are specified below:

Endpoint 1	
Name	developer
Description	This endpoint must return your name and student id.
Parameters	None required

Endpoint 2	
Name	country
Description	This endpoint should return the country names contained in the affiliation database table. The name of each country should be returned only once.
Parameters	None required

Endpoint 3	
Name	preview
Description	This endpoint should return links to preview videos together with the associated content title. The data should be retrieved from the content table. Any content that does not have a preview video should not be returned. The data should be in a random order.
Parameters	1. **limit** (example: /preview?limit=1). The endpoint should support a parameter called 'limit' which defines an integer limit for the number items the can be in the response. Items must still be returned in a random order. A limit of one should therefore return one random item.

Endpoint 4	
Name	author-and-affiliation
Description	This endpoint should return the country, city and institution each author is affiliated with for each publication they are associated with. For each of these affiliations the author id, author name, content id and content name should also be returned. Note that authors can have more than one affiliation for each item of content and that authors can have different affiliations on different items of content. Therefore, authors may have multiple records returned by this endpoint.
Parameters	1. **content** (example: author-and-affiliation?content=99140). Where a content id is specified the response should just contain details for authors of that content. 2. **country** (example: author-and-affiliation?country=japan). This should return only the details where there is affiliation with the specified country. Attempting to use both parameters together should not be supported and result in an error message being returned.

Endpoint 5	
Name	content
Description	The content endpoint should return information about all research 'content' at the conference. This endpoint must return (at least): the title, the abstract, and the content type.
Parameters	1. **page** (content?page=2). If this parameter is used a 'page' of 20 results should be shown (page 1 should show records 1 to 20, page 2 should show records 21 to 40, etc.). 2. **type** (content?type=course). Only content of the specified type should be returned. This parameter should use the type name not the type id.. It should be possible to use both parameters together (for example using content?type=paper&page=1 should return the first 20 results with the type 'paper').

Task 1.2

(20 marks)

For task 1.2 you must add (at least) two further endpoints to your Web API. You will also need to make a new database table. A list of general requirements for each endpoint is given below:

General requirements for task 1.2
• All general requirements for task 1.1 must be met except regarding the request methods. • The endpoints for this task should support appropriate request methods, not limited to GET • A full URL for both endpoints must be given in the readme file and the parameters clearly explained (see appendix 1). • When generating a JWT a unique secure key consisting of random characters must be used • An external library 'firebase/php-jwt' introduced on the module can be used for this task • *The user.sqlite* database should be used. This database can be modified in any way you wish but the two example users provided in the accounts table must be able to log in using their credentials (see appendix 2).

The endpoints you should create are specified below:

Endpoint 6	
Name	token
Description	This endpoint must accept a username and password submitted in the headers of the request. If the username and password are valid, a JWT should be created and returned (if not, an appropriate response should be made). The token must contain the user's id. The token should not contain private data (such as a password) and should not be used to transfer data needed by the client. The token should expire no more than 30 minutes after being issued.
Parameters	**Username** and **Password** must be transmitted via authorization headers

Endpoint 7	
Name	note
Description	Individual users should be able to make short notes about each item of content. These notes should be visible only to that user. This endpoint should be available to authorised users only. A valid JWT should be used to determine authorisation. If the JWT is not valid or is expired then an appropriate 'not authorized' response should be returned. For requests from authorised users the note endpoint should act differently depending on the request method used. If there is a GET request the endpoint must return all notes made by the user identified in the JWT. If there is a POST or PUT request a new note should be added or an existing note updated. A DELETE request should result in a note being deleted.
Parameters	The parameters supported by this endpoint should be determined by you. Different parameters may be needed when using different request methods.

Part 2: Web application

For part 2 you will use React to create a web application that interacts with the web API produced in part 1.

Essential requirements for part 2

(pass / fail)

Your work for part 2 must meet the following essential requirements. These essential requirements are pass/fail and do not have marks directly associated with them. These requirements are taken into consideration in the grade descriptors for the quality requirements and for tasks 2.1 and 2.2. If you fail to meet any of these essential requirements your marks for part 2 will be unlikely to exceed 40% and may be zero.

Essential requirements for all tasks in part 2
• The application must be deployed on the nuwebspace webserver. • The application must be implemented using React. It must be a 'front-end' application and not use server-side React components. You can use any appropriate build tools and third-party components you wish. You should use JavaScript (not TypeScript). • The application must interact with your Web API created for part 1. • You must submit the source code on Blackboard, not just the build files used for deploying the app. • You must not copy text, images, logos or colour schemes from the original CHI 2023 conference website or any related sources. • There must be a readme file giving the full URL for the main landing page of your web application (see appendix 1)

Quality requirements for part 2

(10 marks)

The following are quality requirements that your work for tasks 2.1 and 2.2 should meet. These requirements concern the code as well as the user interface.

Quality requirements for part 2
• You must use React code that is well structured and easy to read. No specific design patterns or system architecture is required but your work should be well organised into components, files and folders. • No files, folders, components, or sections of code should be included in your submission unless they have a purpose for your application. No code should be commented out and messages should not be logged to the console (unless meaningful to do so in a public-facing system). • The React code should follow the style recommendations covered on the module.

- All components should contain comments using the Doc Comment style explained on the module. You should include the @author tag followed by your name in every component you have written or modified. If you have used AI tools you must state this using the @generated tag followed by a short explanation. You do not need to produce any documentation for your code, just use the 'doc comments' format.
- The application must be styled using Tailwind
- The application must have a high-quality design, including an appropriate colour scheme, appropriate and consistent page structure, appropriately presented interface elements and consistent and easy-to-follow navigation. The interface should be fluid or responsive so that it can be viewed on a variety of screen widths.
- The design should not incorporate any 'unfinished' elements or placeholders such as lorem-ipsum text, blank pages, inactive components, deliberate errors or invalid links.

Task 2.1

(20 marks)

For task 2.1 you should create a web application with at least 3 pages. A list of general requirements for these pages is given below.

General requirements for task 2.1
• A router must be used and each page specified below should represent a distinct route in the application. • In addition to the specified pages there must be a '404 not found' page that is displayed when an invalid route name is used. • All pages should have a footer containing your name, student id and the text "Coursework assignment for KF6012 Web Application Integration, Northumbria University". • At least one page should contain an image. This should not be a CSS 'background image' but displayed using React. This could be on any page including the '404 not found' page. • Navigating between routes should not trigger unnecessary new fetch requests if that data has previously been fetched. • If there is no data to display there should be an appropriate message or visual representation. • Additional pages to those specified can be included if they would be meaningful in a real-world, public-facing application.

The three pages you must implement are specified below:

Page 1	
Name	Home
Description	This is the main landing page of your application. It should include: • The heading "CHI 2023" • A menu with links to other pages • The title and a link to a random preview video. More data or features can be included on this page if you wish.

Page 2	
Name	Countries
Description	This page should list each country represented in the affiliation table in the database (giving the name once). This information can be presented as a list or in some other way. There should be a search feature on the page to allow the user to search for the name of a country. More features and functionality can be included if you wish, for example when clicking on a country name further relevant information might be shown.

Page 3	
Name	Content
Description	This page should show details of each item of research content including the title, the abstract, the authors' names, the authors' affiliations, the content type and whether it has won an award. The page should show blocks of 20 items of content at a time, with the ability to navigate to further or previous blocks of content. There should also be a 'select' component giving options to view "All content" or specific types of content such as "Papers", "Posters" etc. If a specific type of content is selected then that data must still be presented in blocks of 20.

Task 2.2

(20 marks)

For task 2.2 you will create two or more new features for the web application that enable users to authenticate (sign in and out) and to retrieve, add and delete short notes about items of content. For both features the following requirements should be met:

General requirements for task 2.2
• The features listed below can be presented as pages within the application or as components within existing pages. The features should be consistent with the overall design of the app. • A signed in user should remain signed in when navigating within the application. • Information should be stored using a cookie or local-storage so that if the user closes the browser they are still signed in when they revisit the application. • When a user signs out, the application should not retain data about that user. • Confidential data about a user must not be logged in the console. • Additional related features and functionality can be included if these would be meaningful in a real-world, public-facing application.

The following features should be implemented:

Feature 1	
Name	Sign in/out
Description	It must be possible to sign into the application. Authorisation should be handled by the backend Web API. Once a user is authenticated they should have the ability to sign out. The JWT returned by the Web API should be stored in an appropriate way by the browser. If a user is signed in but their request is rejected by the Web API (e.g. if they have an expired token) they should be prompted to sign in again. When a user signs out the token should no longer be stored by the browser. Relevant error messages should be displayed when there are unsuccessful attempts to sign in.

Feature 2	
Name	Notes
Description	Signed in users must be able to add notes about items of content. The user should be able to type a short note (limiting the note to around 250 characters is acceptable if you wish) regarding an individual item of content. This note should be saved by the API. Logged in users should be able to see notes they have previously created. They should be able to edit and delete them. Users should only be able to see, edit or delete their own notes, not those made by other people.

Assessment Regulations

You are advised to read the guidance for students regarding assessment policies. They are available online here http://www.northumbria.ac.uk/about-us/university-services/academic-registry/quality-and-teaching-excellence/assessment/guidance-for-students/

Late submission of work

Where coursework is submitted without approval, after the published hand-in deadline, the following penalties will apply.

- For coursework submitted up to 1 working day (24 hours) after the published hand-in deadline without approval, **10% of the total marks available for the assessment** (i.e. 100%) **shall be deducted** from the assessment mark.
- Coursework submitted more than 1 day (24 hours) after the published hand-in deadline without approval will be marked as zero but will be eligible for referral.

The full policy can be found here.

Academic Misconduct

In all assessed work you should take care to ensure that the work you submit is your own. The University takes academic dishonesty and cheating very seriously and it is your responsibility to ensure that you don't attempt to cheat or become victim to cheating.

There are many different forms of academic misconduct or 'cheating'. Plagiarism is the most common and both the University library and your academic tutors are able to provide further guidance on proper citation and referencing in your assessed work.

The full Academic Misconduct Policy is available here.

Useful guidance for avoiding academic misconduct can be found here.

Additional module specific guidance relating to Academic Misconduct
• Materials subject to copyright including images and logos should not be used unless you can demonstrate you have permission or licence to use them. • Publishing code on services such as Github before the feedback date is not acceptable. • Generative AI Tools can be used for programming support. If a tool you wish to use is not one of the following you must first seek permission from the module tutor: ChatGPT, Github CoPilot, Bard, Bing. • Code provided by the teaching team as part of the lectures, workshops, or other forms of teaching and help-giving can be used in full or in part without acknowledgement and without risk of an accusation of misconduct. You should add an @author tag followed by your own name when using code from the module. • Use of code copied from online sources such as StackOverflow is acceptable if properly acknowledged using comments in the code.

Module Specific Assessment Criteria and Rubric

Part 1: Quality Requirements

9 – 10	Exceptional work that significantly exceeds the quality requirements in multiple ways by thoroughly building upon the teaching materials and further independent study.
8	Outstanding work of a professional standard that meets and exceeds the quality requirements by building upon the teaching materials.
7	Excellent work to a high standard that meets the quality requirements. Any minor quality issues are balanced with strengths in meeting other requirements.
6	Very good work that mostly meets the quality requirements but with a fault or problem.
5	Good work but one of the quality requirements is not met or is met in a limited way. Alternatively, there may several distinct minor faults or problems.
4	Acceptable work that shows at least some use of object-oriented programming techniques but fails to adequately address more than one of the quality requirements.
0 – 3	Fail. One or more of the essential requirements specified at the start of Part 1 is not met or there are substantial problems in meeting the quality requirements. Alternatively, the code makes little or no use of object-oriented PHP.

Part 1: Task 1.1

18 – 20	Exceptional work that thoroughly builds upon the teaching materials and independent study to meet and significantly exceed the specification for this task. There is significant additional functionality including additional endpoints and parameters. The API has a highly consistent design in terms of naming and parameter support and in terms of how the response is structured.
15 – 17	Outstanding work of a professional quality that builds upon the teaching materials and independent study to meet and exceed the specification for this task. Additional data is returned and there is likely to be additional parameter and endpoint support. There is a thorough approach to handling parameters including any unexpected data, and a thorough approach to error handling.
14	Excellent work that meets the general requirements and endpoint specifications.
11 – 13	Very good work that mostly meets the general requirements and endpoint specifications. All endpoints and parameters have been implemented but there may be some minor instances of missing or malformed data in the response body. Inappropriate or malformed headers may be set. There may be some issues with error handling.
9 – 10	Good work that attempts the general requirements and each of the endpoints specified. There may be several instances of malformed or missing data or a single more major problem. Parameters are attempted. Error handling is attempted.

8	Acceptable work. There is an attempt at most of the endpoints specified. There may be limited support for parameters, and limited error handling. The API may be limited but can still output JSON data useful for a client web application.
0 – 7	Fail. One or more of the essential requirements given at the start of Part 1 is not met. Alternatively, the essential requirements are met but only one or two of the specified endpoints have been implemented.

Part 1: Task 1.2

18 – 20	Exceptional work that thoroughly builds upon the teaching materials and independent study to meet and significantly exceed the specification for this task. Significant additional functionality relating to how users can authenticate and how authorised users are supported has been added. The JWT is well designed following appropriate standards and is handled appropriately in a variety of situations.
15 – 17	Outstanding work of a professional quality that builds upon the teaching materials and independent study to meet and exceed the specification. There is extra functionality relating to how users authenticate and are authorised, or additional support for authorised users. The JWT is well designed following appropriate standards.
14	Excellent work that fully meets the general requirements and endpoint specifications.
11 – 13	Very good work that mostly meets the general requirements and endpoint specifications. Both endpoints have full functionality but there may be missing data, minor errors, or problems with error handling.
9 – 10	Good work that attempts the general requirements and each of the endpoints specified. There is missing functionality or significant errors. If only one endpoint has been attempted, it must fully meet the general requirements and endpoint specification.
8	Acceptable work. There is an attempt at both endpoints demonstrating some understanding of how they should work, or there is an attempt at one endpoint that offers most of the required functionality.
0 – 7	Fail. One or more of the essential requirements given at the start of Part 1 is not met. Alternatively, the essential requirements are met but there is only a very limited attempt at this task.

Part 2: Quality Requirements

9 – 10	Exceptional work that significantly exceeds the quality requirements in multiple ways by thoroughly building upon the teaching materials and further independent study.
8	Outstanding work of a professional standard that meets and exceeds the quality requirements by building upon the teaching materials.
7	Excellent work to a high standard that meets the quality requirements. Any minor quality issues are balanced with strengths in meeting other requirements.
6	Very good work that mostly meets the quality requirements but with a fault or problem.
5	Good work but one of the quality requirements is not met or is met in a limited way. Alternatively, there may several distinct minor faults or problems.
4	Acceptable work where a working web application has been built using React but which fails to adequately address more than one of the quality requirements.
0 – 3	Fail. One or more of the essential requirements specified at the start of Part 2 is not met or there are substantial problems in meeting the quality requirements. Alternatively, the work makes little or no use of React.

Part 2: Task 2.1

18 – 20	Exceptional work that thoroughly builds upon the teaching materials and independent study to meet and significantly exceed the specification for this task. Significant additional functionality has been added including additional pages and features and there is a thorough approach to how data is presented and can be explored and interacted with in the application.
15 – 17	Outstanding work of a professional quality that builds upon the teaching materials and independent study to meet and exceed the specification. Additional, well-structured, and useful data is presented. There are additional features to allow the data to be explored. Errors are well handled.
14	Excellent work that meets the general requirements and page specifications.
11 – 13	Very good work that mostly meets the general requirements and page specifications. All pages have been implemented but there may be some minor omissions in what data is displayed or problems interacting with the data. There may be some minor issues with error handling. Alternatively, the specification for this task is met in full but the design includes extra pages, information or features that would not be appropriate for a public facing web application.
9 – 10	Good work. There is attempt at every page and feature but with several instances of malformed or missing data or a single more major problem such as a missing feature. Errors are mostly handled.

8	Acceptable work. There are multiple pages and a router is used. Alternatively there is only one page but it successfully displays data from the Web API.
0 – 7	Fail. One or more of the essential requirements for part 2 are not met. Alternatively, the essential requirements are met but the React application does not have multiple pages or display data.

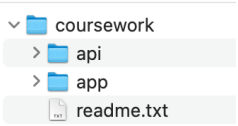
Part 2: Task 2.2

18 – 20	Exceptional work that thoroughly builds upon the teaching materials and independent study to meet and significantly exceed the specification for this task. Significant additional functionality has been added for handling user accounts and for supporting relevant actions by authorised users.
15 – 17	Outstanding work of a professional quality that builds upon the teaching materials and independent study to meet and exceed the specification. All requirements for the endpoints are met and additional functionality is added.
14	Excellent work that fully meets the general requirements and feature specifications.
11 – 13	Very good work that mostly meets the general requirements and feature specifications. Both features have full functionality but there are problems with how they are implemented or with how errors are handled.
9 – 10	Good work. There is an attempt at both features but full functionality is not offered for either or there are multiple problems. If only one feature has been attempted it must fully meet the general requirements and specification for that feature.
8	There is an attempt at one of the features with some meaningful functionality offered.
0 – 7	The essential requirements for part 2 are not met or the task is not addressed in a meaningful way.

Appendix 1: Readme file

There must be a readme file in the main folder of your submission. This file should ideally be in plain text (.txt) or markdown (.md) format but other formats including .doc are acceptable. This file must be included as part of your submission on Blackboard but should not be placed on the web server.

Below is an example of how the work you submit on blackboard might be structured:



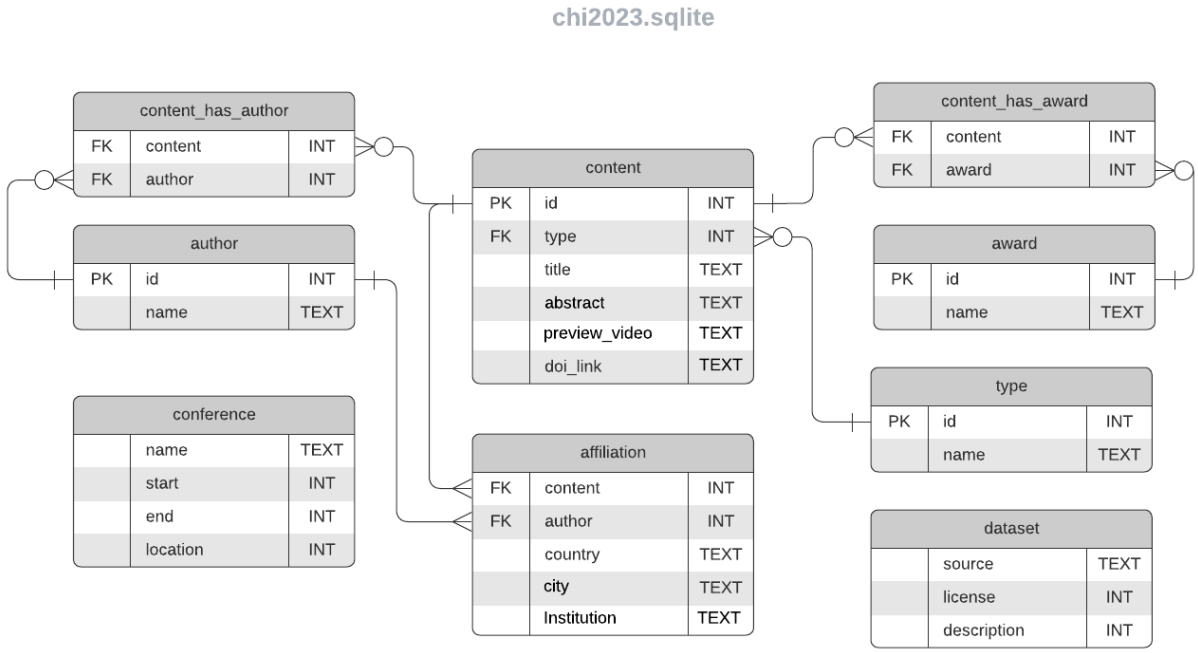
The readme file should include:

- Your name and student id
- Any URLs needed for marking your coursework
    - For Part 1 you should give the URL of each endpoint and a full URL giving an example of each parameter. Where the API endpoint requires data to be posted or transferred in the headers you must still give the full URL for the endpoint and you should briefly specify what the parameters or headers need to contain.
    - For Part 2 just give the URL for the main landing page of your application. If for any reason your application does not contain a menu or is difficult to navigate you can include links to your other pages.
    - If you do not include the correct URLs or if you do not describe parameters in a clear way (e.g. by misspelling the parameter name or by providing an example that does not work) the marker may judge you as not having completed that task.
- You can include any essential information needed for using your application (most projects will not need to give additional information)
- Other than your name and student id, the file should not contain personal, private, or confidential information.

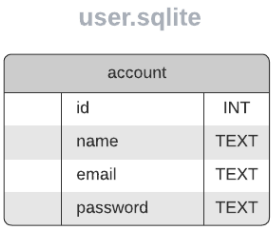
Do not make the readme file a page within the web application you create for part 2. Do create separate readme files for different parts of the work or different tasks.

Appendix 2: Databases

This is an Entity Relationship Diagram for the main database containing the research content. This database should not be modified.



This is an Entity Relationship Diagram for the database containing user information. This database can be modified but you should not change the existing users' email and password.



The following accounts are stored in the user database:

Email	Password
john@example.com	examplePassword1234
admin@example.com	CHI2023