Northumbria University NEWCASTLE

Department of Computer & Information Sciences

ASSESSMENT BRIEF

Module Title:	Web Application Integration
Module Code:	KF6012
Academic Year / Semester:	2021-22 / 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:	14th October 2021
Mechanism for Handout:	Module Blackboard Site & Seminar in Week 5
Deadline for Attempt Submission by Students:	18th January 2022 11.59pm GMT
Mechanism for Submission:	Document upload to Module Blackboard Site
Submission Format / Word Count	Upload your work in a single zip folder to blackboard. Your work must also be deployed on a web server. You must include all source files except the 'node_modules' folder for part 2. You must include the database and a 'readme' file. Submissions for this module are not anonymous and your work should include your name.
Date by which Work, Feedback and Marks will be returned:	1st February 2022
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

You are asked to build a web application that presents information about academic papers on the topic of Designing Interactive Systems (DIS). The application you build should be suitable for people interested in discovering and reading DIS research papers. You will be given a dataset to use for this coursework (as an SQLite database). An SQLite database will also be used for user accounts.

The work you do for this coursework will have two parts:

- Part 1: You will build a Web API using object-oriented PHP that outputs JSON data.
- Part 2: You will build a Client Application using ReactJS.

The Web API built for part one will be used by the Client Application built for part two. Together these will display information and handle user accounts.

General Information

Before stating the requirements for part 1 and part 2, here is some general information about the coursework and your submission. Please read this carefully well in advance of the submission date. Failure not to have read this information and not to have acted upon any issues in a timely way may lead to you failing the module.

Deployment on web server

The coursework must be deployed on a public webserver. You can use newnumyspace, or you can use an alternative if you prefer. If you are using an alternative it is a good idea to discuss this with the module tutor first.

If you wish to use newnumyspace but do not have an account or cannot access it, you must inform the module tutor during the semester one teaching period. An account cannot be created for you over the Christmas holiday and a lack of an account will not be grounds for an extension or exception.

Recommended URLs

We recommend that part 1 and part 2 of your coursework can be accessed via the following URLs. These are recommendations, not rules - your folder structure and URL *can* be different. Please note you will need to use your personal newnumyspace URL that includes your student id at the start.

- `newnumyspace.co.uk/kf6012/coursework/part1/`
- `newnumyspace.co.uk/kf6012/coursework/part2/`

Only upload the necessary files to the webserver for your system to function (for part 2 this will just be the content of the build folder and not the readme file, source files, node modules, etc.). You will need to upload more files to Blackboard.

Submission files

Your submission on Blackboard will differ in several ways to the code you deploy on the web server. You Blackboard submission must be as a single zip folder containing:

Content	Type	Description
README	File (.txt or .md)	A readme file in text or markdown format. See next section.
part1	Folder	A folder containing all of your code and files for part1 as deployed on the server. This must include the database and .htaccess files as well as the code.
part2	Folder	A folder containing all files and folders from your work for part2, except for the node_modules folder. Please delete the node_modules folder from your work before submitting. Make sure to include the source folder (src) or you may receive no marks for your code.

Use of README file in submission folder

Your submission must include a README file. This should be in either text (.txt) or markdown (.md) format (not both). You can call the file either "README" or "README.txt" or "README.md". The file should contain the following:

- Your name and student id.
- The URLs (website links) for part 1 and part 2 of your coursework.
- Where necessary: Clear and concise additional information about the submission. Do not include irrelevant or overly detailed information; if a substantial amount of extra detail is given the marker may not be able to read all of it.

Do not put the README file on the webserver.

Databases

Two SQLite databases are provided.

- `dis.sqlite`
- `users.sqlite`

Your coursework must use the databases provided. You are free to make changes to the users.sqlite database if you wish but should not change the dis.sqlite database.

Appropriate use of the databases

This coursework uses a real-world dataset featuring information about real people and real research papers. Therefore, care must be taken with how you present this data.

You have permission to use the dataset (dis.sqlite), but please note:

- You should not modify information in the dataset such as author names, titles, etc.

- You should use the database as the source of information for your application. You should not copy data from elsewhere or attempt to hard code any data.
- You must not use any text, logos or branding associated with the Designing Interactive Systems (DIS) conference, its partner organisations, and its sponsors.
- The site you build should not claim or imply that it is an official site or officially endorsed.
- The site you build should acknowledge the Designing Interactive Systems conference (as described in the specification for part 1 and part 2 below).

Copying text, images and logos including from the websites associated with the DIS conference or other sources may be considered academic misconduct and lead to you failing the module.

Users database

The user table in the users.sqlite database has information about the following users:

1. email: `john@example.com`
   password: `johnpassword`
2. email: `kay@example.com`
   password: `kaypassword`

A user's email is their username and will be unique. The passwords have been encrypted in the database and can be unencrypted using standard PHP functions. Please ensure these users exist in your final submission and the passwords are unchanged.

Marking

Your work will be marked using an up-to-date version of the following: Firefox, Chrome, Edge, and/or Postman.

Your work on this module is marked using the grading criteria at the end of this coursework specification. This means that a decision is made when marking 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 meeting all of the descriptors beneath it. If your work meets the coursework specification in full then you can expect to gain 70-80% for your work (a first class). Outstanding work (80%+) and exceptional work (90%+) must meet and exceed the specification in this document. If your work does not fully meet a grade descriptor but also includes outstanding or exceptional work, it will only be awarded the grade for the descriptor that it meets in full.

A selection of 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. A viva is helpful in several circumstances including explaining excellent and adventurous solutions, discussing bugs and errors, as well as where academic misconduct is suspected.

Part 1: Web API

Overall Value: 50 marks

Part 1 is worth 50 marks. There are three tasks. You will be graded for functionality of task 1.1 (10 marks), functionality of task 1.2 (10 marks), functionality of task 1.3 (10 marks) and overall implementation quality for tasks 1.1, 1.2 and 1.3 (20 marks). Grade descriptors are given at the end of this document.

General quality requirements

Your work must meet these general requirements:

- The web API must be deployed and working on a web server (such as Newnumyspace or an alternative).
- The web API must be written in object-oriented PHP.
- The web API must use the provided SQLite databases.
- Your work must not use a PHP framework, template engine, or package manager.
- The code must set appropriate headers and HTTP status codes.
- Errors must be handled appropriately with a correctly formatted, non-empty response.
- The code must use an autoloader, exception handler and error handler.
- The code must be appropriately commented, and every class must include a comment with an author tag with your name and student id.
- The code should follow appropriate quality standards and guidelines, and must be written in a consistent style.
- The code should be appropriately modular and concise. There should be no redundant (unused) code or commented-out code.
- Clean URLs (that do not end with a .php file extension) and a single point of entry must be used.

Task 1.1: Human-readable webpages

You must make two human-readable webpages plus a human-readable error page. These should use Object-Oriented PHP to output HTML.

Page 1: **Home page**

- This should be displayed if the user requests the base URL for part 1 (e.g. `newnumyspace.co.uk/kf6012/coursework/part1`).
- The page must contain:
  - Your name and student id number.
  - A message saying your work is university coursework and not associated with or endorsed by the DIS conference.
  - A link to the documentation page.
  - It can include more information if you wish.

Page 2: **Documentation page**

- This should be displayed if the user requests part1/documentation (e.g. `newnumyspace.co.uk/kf6012/coursework/part1/documentation`).
- The page must contain:

- Your name and student id number.
- A link to the home page.
- Clear, structured information about every API endpoint you have implemented (see tasks 1.2 and 1.3). This must cover:
    - The endpoint name.
    - A full URL (link) for the endpoint.
    - Accurate information about what HTTP methods are supported by the endpoint.
    - Full information about what parameters are supported .
    - An overview of the likely content and structure of JSON responses.
    - Likely HTTP status codes set for responses (including for errors)
    - One specific example, including:
        - A specific example request (URL and parameters).
        - The expected JSON response for that specific request.

Page 3: **Error page**

- This should be returned in the event of the user requesting a page that does not exist. It must not be displayed if the error is associated with an attempt to access the Web API (see part 2).
- This page would not normally be linked from a menu, but may contain links back to the home or documentation page.

The pages must be generated using object-oriented PHP. The pages should be clear, structured and well-designed. They should use CSS or a framework such as Bootstrap. Some JavaScript can also be used so long as the page itself is generated using object-oriented PHP. ReactJS must not be used.

Task 1.2: API endpoints and parameters

You must build a web API that uses object-oriented PHP to return JSON encoded data from the databases. The web API must have the endpoints listed below. It can also have additional endpoints.

Endpoint 1: `/api`

- This should be the base endpoint, e.g. `newnumyspace.co.uk/kf6012/coursework/part1/api`
- This endpoint should return JSON containing your name and student id, a short explanation of what the API is, and information on how to access the API documentation. This data must be structured, clear and concise. It must use appropriate keys and values. It can include further relevant data and metadata if you wish.

Endpoint 2: `/api/authors`

- e.g. `newnumyspace.co.uk/kf6012/coursework/part1/api/authors`
- This endpoint should return an array of authors. Each item in the array should describe an author including their name and author_id. The endpoint can return further relevant data and metadata if you wish.
- This endpoint should support the following parameters:
    - `id`
        - e.g. `/api/authors?id=8032`

    - This should return the author with the specified id or an appropriate error message if there is no author with that id.

Endpoint 3: `/api/papers`

- e.g. `newnumyspace.co.uk/kf6012/coursework/part1/api/papers`
- This endpoint should return an array of academic papers. Each item in the array should describe a paper, including the paper_id, title, abstract, and awards. The endpoint can return further relevant data and metadata if you wish.
- This endpoint should support the following parameters:
  - `id`
    - e.g. `/api/papers?id=50`
    - This should return the paper with the specified id or an appropriate message if there is no author with that id.
  - `authorid`
    - e.g. `/api/papers?authorid=8032`
    - This should return all papers authored by the person with the specified id, or an appropriate message.
  - `award`
    - e.g. `/api/papers?award=all`
    - This should return only the papers that have won an award (all awards).

The API must set appropriate HTTP status codes for each response. All requests beginning with /api should be considered a request to the API and any requests to an API endpoint that does not exist must be considered an error. The API must respond with an appropriate error message encoded in JSON in the event of a request to an API endpoint that does not exist (not an empty response or an HTML response).

Your API can have additional endpoints to those listed here and/or additional parameter support. All endpoints must be fully documented. The API will be used in part 2 of the coursework. You may wish to redesign or extend your API when you work on part 2.

Task 1.3: Authentication and reading list

For this task you must implement additional web API endpoints

Endpoint 4: `/api/authenticate`

- This endpoint should support POST requests
- The endpoint must accept two parameters: a username and password. An email address should be used as a username for this system.
- The endpoint must return a JSON Web Token (jwt) if the username and password are correct
- The jwt should hold appropriate data including an expiry date.
- The token must be signed using a secure server key that is unique to your coursework submission.

Endpoint 5: `/api/readinglist`

- This endpoint should support POST requests that include a valid JSON web token.

- The endpoint should allow individual users to add or remove a paper from their personal 'reading list'.
- The endpoint should allow users to retrieve the ids or details of papers on their reading list
- You will need to consider how to design this endpoint and what parameters are appropriate.
- You will need to create a new database table to support this functionality.

You are not limited to using the specific endpoint names given in this section or using only the two suggested endpoints for this functionality. You may also offer additional endpoints and related functionality. You must make sure to properly document all of your endpoints on the documentation page (see task 1.1.)

Part 2: Client Application

Overall Value: 50 marks

Part 2 is worth 50 marks. There are three tasks. You will be graded for functionality of task 2.1 (10 marks), functionality of task 2.2 (10 marks), functionality of task 2.3 (10 marks) and overall implementation quality for tasks 2.1, 2.2 and 2.3 (20 marks) Grade descriptors are given at the end of this document.

General quality requirements

The application must meet the following general requirements:

- The application must be created in ReactJS.
- The application must be deployed and functional on a web server.
- .htaccess files can be used on the webserver. If so, you must add these to the build folder as part of your blackboard submission.
- The application must use a router (react-router-dom). You may use additional packages but must justify these in the README file.
- You must make use of JavaScript classes for your components (or you must briefly explain and justify an alternative approach you have taken in your main README file).
- You must use an appropriate format for commenting your code, and include appropriate author tags for each component.
- The code should meet appropriate quality standards and guidelines and must be consistently styled.
- The application must be appropriately styled using CSS and or an appropriate framework.
- The functionality of the application must be clear to someone using it. For example pages should be linked by a menu, and clickable content should be easily identifiable.
- The code should be appropriately modular and concise. Components should be reused where possible. There should be no redundant (unused) components, unused code, or commented-out code.

Task 2.1: Web pages

You should build a client application in ReactJS. The application should have the following pages:

Page 1: **Home**

- This must be the main 'landing page' for the application. It should be displayed when the user navigates to /part2/. For example, the URL might be:
    - `newnumyspace.co.uk/kf6012/coursework/part2/`
- This page must have a menu linking to the other pages, and a footer containing your name, student id and a short statement that this is university coursework.
- This page should include an appropriate image. You must use an image that is copyright-free or you have permission to use.
- Note: You will add more content to this page for task 2.2.

Page 2: **Papers**

- This page should have a consistent menu and footer with the home page.
- Note: You will add more to this page for task 2.2.

Page 3: **Authors**

- This page should have a consistent menu and footer with the home page.
- Note: You will add more to this page for task 2.2.

The design of these pages is up to you, but the design must be appropriate and consistent. The pages must not imply that it is an official DIS site, that it is connected to, or it is endorsed by the conference or any or its sponsors.

Task 2.2: Display data from web API

The client application must make use of the API produced in part 1 to display data about research papers and authors. If you have carefully designed your API for part 1, creating the functionality listed here will be easier. You may wish to add additional features to your API to make it easier to meet the requirements given here.

Add the following content to the pages you created in task 2.1:

Data for page 1: **Home**

- The home page should display details for one randomly selected research paper.

Data for page 2: **Papers**

- This page should list research papers. The papers should be listed in alphabetical order by title.
- When the title is clicked, the following information should be displayed:
    - The abstract.
    - The authors.
- Papers should be presented in pages, not as a long list.
- The user should be able to search all of the papers (by title and abstract).
- The user should be able to select a paper by award (e.g. if the paper won an award).

Data for page 3: **Authors**

- This page should list authors by name. Clicking on an author's name should list the details of papers associated with that author, including title, abstract, and authors.
- This page should not provide one long list of authors, but should divide the list into sub-pages (e.g list 25 authors per sub-page).
- The user should be able to search authors by name.

Additional data and functionality can be presented on the home, papers and authors pages, beyond that specified above. This should contribute meaningful further information and be well presented. The client application does not need to make use of every API endpoint and parameter you have implemented, but should aim to use a range of these.

Task 2.3: Log in and reading list

The client application must provide a means for a user to log in and log out. A user should be considered logged in if the username and password they enter is validated by the API and a JSON Web Token is returned. How the login page or component is designed is up to you, but your application must:

- Keep the user logged in if they navigate away from the login page or leave your site.
- Use localstorage to store the web token and remove the web token when logged out.
- Log the user out and/or prompt them to re-enter their username and password if the token has expired (you can rely on the server to tell you if the token is expired, you do not need to read the token using ReactJS).

A logged in user should have access to the following functionality:

- Add papers to their reading list.
- Remove papers from their reading list.
- View papers on their reading list.

How you implement this reading list functionality is up to you (you can modify an existing page, or you can add extra pages if you wish).

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 working day (24 hours) after the published hand-in deadline without approval will be regarded as not having been completed. A mark of zero will be awarded for the assessment and the module will be failed, irrespective of the overall module mark.

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.

Remember, this is an INDIVIDUAL assessment and should be entirely your own work. Where you have used someone else's words (quotations), they should be correctly quoted and referenced in accordance to the Harvard System. Help regarding referencing and guidance on avoiding plagiarism can be found in the Study Skills section of the module site on Blackboard.

Module Specific Assessment Criteria and Rubric

Part 1

Functionality of human readable webpages (task 1.1)

Mark	Criteria
9-10	Exceptional work that exceeds the specification in several ways. The work should include exceptionally high-quality documentation. All three webpages should be exceptionally informative and well designed.
8	Outstanding work that exceeds the specification in terms of the information provided on the web pages and quality of presentation. The documentation page has additional information and/or features, making the API easy to understand and use.
7	Excellent work that meets the specification in full
6	Very good work that mostly meets the specification. The work is outstanding or exceptional in some respects but does not meet the full specification. The documentation may have one or more errors such as an inaccurate parameter name or an inaccurate or unclear example. The documentation may be presented in a way that is difficult to read and follow. Some required information on the documentation or home page might be omitted.
5	Good work that meets parts of the specification. There may be multiple errors or ambiguities on the documentation page. Some of the required information may be missing from the documentation page. The home or error pages may lack most of the required information. The pages may be poorly designed, making them difficult to read and navigate.
4	Acceptable work that somewhat meets the specification. Significant information may be missing from the documentation page, or there are significant inaccuracies. The pages may need to be accessed from the browser using a .php file extension rather than a “clean” URL.
1-3	Not acceptable. The required pages are missing or cannot be accessed on the server when using the URLs in the README file. There is no relevant information on the documentation page. The site implies it is an official site associated with the DIS conference and/or misuses text and logos associated with the conference and it’s sponsors.
0	No attempt, or the attempt does not use object-oriented PHP.

Functionality of web API endpoints and parameters (task 1.2)

Mark	Criteria
9-10	Exceptional work that exceeds the specification in several ways, including additional useful endpoints and parameters, exceptionally well designed JSON responses, and clear and meaningful error handling for a range of possible situations. Any deviations from the specification are justified in the README file.
8	Outstanding work that exceeds the specification. Meaningful additional endpoints and parameter support are provided (these need to be documented, and ought to be used by the client application in part 2). Any deviations from the specification are justified in the README file.
7	Excellent work that meets the specification in full
6	Very good work that mostly meets the specification. The work is outstanding or exceptional in some respects but does not meet the full specification. Some of the required information might not be returned in the JSON response. There may be an error with one of the parameters or when certain parameters are used together. Metadata in the JSON response and/or the HTTP status codes might not be accurate or appropriate.
5	Good work that meets parts of the specification. There may not be full parameter support. Significant information may be missing or inaccurate in some responses. Error responses might be empty or return HTML. Invalid JSON may be returned in some situations, or JSON headers are not set. An inappropriate endpoint revealing user data such as emails or passwords might have been created.
4	Acceptable work that somewhat meets the specification. Some data is returned from the database, although this is limited and often not in a valid format. API endpoints might have to be accessed from the browser using a .php file extension rather than via a ‘clean’ URL.

1-3	Not acceptable. The API does not return any data from the database. The dis.sqlite database has been inappropriately modified.
0	No attempt

Functionality of authentication and reading list (task 1.3)

Mark	**Criteria**
9-10	Exceptional work that exceeds the specification in several ways. There is additional functionality such as further support user accounts creation and management and/or for reading-list features. Advanced support and features for handling JSON Web Tokens may be implemented.
8	Outstanding work that exceeds the specification. There may be additional appropriate slots implemented for the JSON Web Token, and excellent support for error handling.
7	Excellent work that meets the specification in full
6	Very good work that mostly meets the specification. The work is outstanding or exceptional in some respects but does not meet the full specification. There may be minor errors such as a missing or invalid expiry date in the token. There might be use of an inappropriate or insecure secret key, or there may be some problems with error handling. There may be some errors or limitations with reading list functionality.
5	Good work that meets parts of the specification. The API allows users to authenticate, but there may be errors with the encoding of the token, inappropriate content might be contained in the token, or there might be poor error handling. Support for reading list functionality might only be partially implemented and/or authorisation might not validate the JSON Web Token or check the expiry date.
4	Acceptable work that somewhat meets the specification. The work performs some form of authentication when given a username and password but the JSON response might not contain a valid JSON Web Token. There is an attempt at the reading list functionality, but it may be limited and/or contains multiple bugs and errors.
1-3	Not acceptable. The work does not attempt to use JSON Web Tokens. The work wrongly attempts to set a PHP 'session', perhaps in conjunction with a JSON Web Token. There is an attempt at either the authentication or reading list functionality but not both.
0	No attempt

Code quality for part 1 (tasks 1.1, 1.2 and 1.3)

Mark	**Criteria**
18-20	Exceptional work that exceeds the quality requirements in several ways, demonstrating insight and skill in object-oriented approaches. The work demonstrates exceptional attention to detail and best practices in all aspects of the work. Approaches may be justified in the README.
16-17	Outstanding work that exceeds the quality requirements. The code is carefully and appropriately commented, consistently follows appropriate coding standards, and is thoughtfully modularised and organised using object-oriented methods.
14-15	Excellent work that meets the quality requirements in full and consistently follows the approaches discussed on the module.
12-13	Very good work that might be outstanding or exceptional in some respects but is incomplete in others. The work might not handle errors and exceptions appropriately, might not restrict direct URL access to the databases or other files, or may contain bugs or errors. The quality of the code might be inconsistent or there might be inappropriate use (or non-use) of comments
9-11	Good work. A good attempt at object-oriented PHP but with several bugs, errors, omissions or other shortcomings. There may be a substantial amount of procedural PHP, or a failure to appropriately modularise the code contained within classes. An autoloader might not be used. There may be a lack of attention to detail with the quality of code.
8	Acceptable work focusing on at least one of the tasks, using an object-oriented approach and at least partially following the quality requirements. The work might not be deployed on a web server.
1-7	Not acceptable. The work is in PHP but is mainly not object-oriented. A framework, template engine or package manager might have been used. The work may be deployed on a server but the source code has not have been submitted on Blackboard and so cannot be marked.

0	No attempt, or PHP not used.

Part 2

Functionality of web pages (task 2.1)

Mark	**Criteria**
9-10	Exceptional work that exceeds the specification in several ways. The pages are exceptionally well designed, suitable for a range of screen sizes, make use of a range of appropriate components and prioritise usability and accessibility.
8	Outstanding work that exceeds the specification. There is outstanding page design and styling. There is use of appropriate interactive interface features on the page. The visual presentation and interactivity of data on the pages is outstanding.
7	Excellent work that meets the specification in full
6	Very good work that mostly meets the specification. The work is outstanding or exceptional in some respects but does not meet the full specification. There may be some problems with the styling or formatting of the page. There may be minor problems with navigating or interacting with data displayed on the page.
5	Good work that meets parts of the specification. There may be several omissions or errors. The pages might not be styled, or use only very basic styling. Data on the page might not be appropriately styled or presented, or presented in ways that makes it difficult to read or interact with.
4	Acceptable work but only one page is available or there is not a menu linking pages. There may have been problems deploying the site meaning it cannot be viewed on the server.
1-3	Not acceptable. There is an attempt but the web pages do function.
0	No attempt, or the attempt does not use ReactJS.

Functionality of displaying data from web API (task 2.2)

Mark	**Criteria**
9-10	Exceptional work that exceeds the specification in several ways. The pages are used to present a range of detailed and accurate information. The user can interact with the data in various, meaningful ways. There is a sense of cohesion and integration of data when navigating the site. Data is not logged in the console during ordinary use of the site.
8	Outstanding work that exceeds the specification. The web application presents additional, accurate and meaningful information and/or provides additional meaningful functionality beyond the specification.
7	Excellent work that meets the specification in full.
6	Very good work that mostly meets the specification. The work is outstanding or exceptional in some respects but does not meet the full specification. Most, but not all, of the required data might be presented, or there may be some limitations with how users can interact with the data in the required ways. There may be minor errors or misrepresentations in the required data or in any additional data that the pages display.
5	Good work that meets parts of the specification. The work presents some of the required data. There may be major restrictions to how the data can be interacted with in the required ways. There may be significant bugs, errors or misrepresentations in the data or how it is displayed.
4	Acceptable work that presents at least some data fetched from the Web API. The Web API might require debugging to be fully functional, for example it uses the wrong URLs or tries to fetch data from localhost.
1-3	Not acceptable. No data is presented on the pages, but there is at least some meaningful attempt at this functionality.
0	No attempt, or the data is hard coded. ReactJS is not used.

Functionality of log in and reading list (task 2.3)

Mark	Criteria
9-10	Exceptional work that exceeds the specification in several ways. There is meaningful additional functionality for managing or creating user accounts, managing logged-in status in the client, and/or for reading list management.
8	Outstanding work that exceeds the specification. The work offers some additional functionality and/or integrates the reading-list functionality with the other pages and features in meaningful ways.
7	Excellent work that meets the specification in full
6	Very good work that mostly meets the specification. The work is outstanding or exceptional in some respects but does not meet the full specification. The token might not be deleted from local storage when logging out, or there may be minor limitations, problems or errors with the authentication or reading list functionality.
5	Good work that meets parts of the specification. The work might implement either the authentication functionality or reading list functionality, but not both in full. The user might not stay logged in when navigating between pages. Reading list items might not be stored or retrieved for the user.
4	Acceptable work that attempts the functionality in meaningful ways, but does not offer meaningful login functionality or meaningful reading-list related functionality
1-3	Not acceptable. There is some attempt at the functionality, but this cannot be used in any meaningful way.
0	No attempt

Code quality for part 2 (tasks 2.1, 2.2 and 2.3)

Mark	Criteria
18-20	Exceptional work that exceeds the specification in several ways, demonstrating insight and skill in ReactJS. The work demonstrates exceptional attention to detail and best practices in all aspects of the work. Use of approaches beyond those taught are justified in the README.
16-17	Outstanding work that exceeds the quality requirements. The code is carefully and appropriately commented, consistently follows appropriate coding standards, and is thoughtfully modularised and organised.
14-15	Excellent work that meets the quality requirements in full, following the approaches discussed on the module.
12-13	Very good work that mostly meets the quality requirements but with some shortcomings or inconsistencies. There may be some minor bugs or errors. There may be inappropriate use (or non-use) of comments. There may be unused components or unduly repetitive code. Some components might not be stored appropriately, for example within the files containing other components.
9-11	Good work that meets parts of the specification. There may be multiple errors, bugs or problems with the code even though it is mostly functional. The code may not be appropriately modularised, for example with all components in one file. The presentation of the code may be poor or highly inconsistent. The code may be well written but not enough features are implemented to justify awarding marks.
8	Acceptable work focusing on at least one of the tasks. The work uses ReactJS and at least partially follows the quality requirements. The build code may not have been deployed on the server properly and the application cannot be used without running the source code locally.
1-7	Not acceptable. There is an attempt at ReactJS but it is not complete or does not execute. The build files may have been successfully deployed on the server, but the source code has not been submitted on Blackboard and so cannot be marked.
0	No attempt, or ReactJS is not used.