Assessment Brief

Programme:	Computer and Information Science
Module Code:	KF6012
Module Title:	Web Application Integration
Distributed on:	16 October 2020
Submission Time and Date:	To be submitted by 23:59 GMT on 4th January 2021
Weighting	This coursework accounts for 100% of the total mark for this module
Submission of Assessment	The coursework must be submitted via the link on Blackboard. Each part must be in the same .zip folder with your student number. Note, you must submit all files necessary for your system, including the database. You must include all source code. You can include a readme file to give any necessary information needed to run your code. Note that this coursework is not anonymous. You may use @author tags in your code. It is your responsibility to ensure that your assignment arrives before the submission deadline stated above. See the University policy on late submission of work.
Marking environment	Your work will be marked using your submitted code and Newnumyspace (unless otherwise agreed).

Instructions on Assessment:

Introduction

You will build two related web applications that, when used together, present information about an academic conference (CHI2018). This is an annual, international conference with over 3000 attendees and which features hundreds of presentations by international authors on research topics related to Computer-Human Interaction (CHI). The conference takes place over several days and features multiple parallel sessions.

The applications you build should be suitable for people attending the conference. They should help people find out about:

- The **schedule** for the conference (what is happening when). The main conference schedule takes place over four days (Monday to Thursday). These days are divided into 'slots' spanning a period of time (e.g. 9am to 11am). Each slot may have one or more 'sessions' in which presentations are given. There may be multiple sessions in a slot, each running in parallel in a different room. Each session will have a session chair whose job it is to manage the session.
- The **content** of the conference. There are multiple forms of presentation at the conference, including papers, panels, courses and more. Presentations will have a title and, in most cases, an abstract. Many of the presentations will have multiple authors associated with them. A presentation is a form of 'content' at the conference. Some content such as coffee breaks are not presentations, but should still appear on the schedule.
- The **authors** at the conference. Authors are people that have written work to be presented at the conference, and/or may be involved in other aspects of the conference such as chairing a session. An author might have written multiple papers and be involved in more than one session. Most papers have more than one author.

You will be given a dataset to use for this coursework (as an SQLite database). This dataset includes information about the schedule, the presentations and the authors. This is a real-world dataset featuring information about real people and real research papers, and therefore care must be taken with how you present this data. You have permission to use the dataset, but please note that you must not copy any data directly from the CHI2018 conference website, and you must not use any logos or branding associated with the CHI conference or partner organisations (including its sponsors). The site you build should not claim or imply that it is an official site or officially endorsed. Copying text, images and logos from the conference may be considered academic misconduct and lead to you failing the module, so don't do that.

To help you understand the conference and dataset, you can look at the PDF programme or the conference website (https://chi2018.acm.org/). To be clear, your task is to present information about the conference but not to replicate the existing website or design. Do not copy images, logos or text from the original conference website.

To understand the terminology used in this coursework, see P26 of the PDF programme. An example of a **slot** is "Monday 11.30 – 12.50". There are 18 **sessions** in that slot, including "Devices and Interactions for Care". For that session, the **session chair** is "Elizabeth Murnane". The **content** of that session is four paper presentations, including "Exploring the Design of Tailored Virtual Reality Experiences for People with Dementia". That presentation has four **authors** including "James Hodge", and an **abstract** that begins "Despite indications that recreational virtual reality (VR) experiences could be … ". That presentation has won an award, as signified by the medal shown on the programme.

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

- Part 1: You will build a Web API using object-oriented PHP. The API will serve information about the conference schedule and presentations in JSON format.

- Part 2: You will build a Client Application using either React or AngularJS. The client application should use data from the API for presenting information about the conference schedule and presentations.

Part 1: Web API Created Using Object Oriented PHP (50%)

For part one of the coursework you will build a web API. This must be implemented using Object Oriented PHP and SQLite. It should also use HTML5 and CSS, but should not use React or AngularJS. You may use a framework such as Bootstrap to style the human readable pages for part1 (but note you may not do so for part 2).

- You must produce a web API and deploy this on the webserver Newnumyspace.
  - You will need to provide a link to your work on Newnumyspace for marking.
  - You also need to submit all of your code and your database in a zip file.

- The API must include the following human readable webpages (written in PHP, HTML5 and CSS):
  - A **documentation** page that
    - States each endpoint of the API
    - Gives a brief description of what data the endpoint returns
    - If applicable, states what parameters or data are required by the endpoint
    - States whether authentication is required by the endpoint
    - Gives an example of a request to that endpoint
  - An **about** page that states who developed the website (your name), that it is University coursework, and is not associated with the CHI conference or any of its sponsors.

  The site does not need to present the schedule and presentation data from the database as human readable web pages. The site should simply give information about the web API.

- The API must output data about the conference presentations and schedule in JSON format. The endpoints should be the same as those you list on your documentation page. The endpoints do not need to mirror the structure of the database, but should be appropriate for presenting a schedule and information about presentations.
  - The following endpoints are required:
    - A general endpoint **/api/** that should give basic information about the API in JSON format including a welcome message, the developer (your name), and a list of available endpoints
    - A login endpoint **/api/login/** that accepts a username and password and returns an error message or a JSON Web Token.
    - An update endpoint **/api/update/** that checks for a valid JSON Web Token and if appropriate will update some part of the database (such as the title of a session) and return an appropriate response.
  - The other endpoints are up to you. You need to create several more endpoints beyond the three listed above. For example you could create an endpoint to return slots **/api/slots/** or authors **/api/authors/** or a specific author **/api/authors?id=1234**
    - You should carefully design your endpoints with respect to the functionality required in part two. If you have a single endpoint that returns the entire database, your application for part two may run extremely slowly. If you endpoints simply mimic the structure of the database then that may make the client code unnecessarily complex.
  - With the exception of authentication data, it is up to you how to pass parameters to the API. For example to return a specific author you could pass the author id as a variable **/api/authors?id=1234** or as part of a directory structure **/api/authors/1234/** or you might use HTTP POST to send the author id within the request body, e.g. **{"author" : "1234"}**.
    - The login endpoint must only accept a username and password posted in the body of an HTTP request. You should not pass these in the URL using a GET request.
    - The update endpoint, and all other endpoints that require a JSON Web Token, ought to accept the token as part of the body of an HTTP request.

- The API must support JSON Web Tokens for authentication.
  - A valid token should be returned upon successful login, and that token must state whether a user has admin status, and should have other appropriate data including an expiry date.
  - The token must be checked for all endpoints that require authentication (as specified on your documentation page).

- Users with admin status should be able to update the title of a session (via the appropriate API endpoint, if a valid JSON web token is supplied)

- The API must be written in Object-Oriented PHP
    - The application should have an appropriate architecture and directory/file structure
    - You should use appropriate design patterns including a singleton database connection, a record set and page controllers. You should use an autoloader.
    - Important information such as the location of the database and the secret key for JSON Web Tokens should be included in a config file (e.g. config.ini or config.json)
    - The API should use a single point of entry with an appropriate .htaccess file to direct all requests through a single script (e.g. index.php). Having a separate script for each endpoint is not acceptable.
    - An .htaccess file should be used to secure directories such as /classes/.
    - The code should be appropriately indented and commented. Block "doc comments" should be used for each class, and for methods and properties where appropriate. Appropriate @tags should be used.

- The API should handle errors and exceptions appropriately
    - An exception handler should return an appropriate message in an appropriate format (JSON for API requests, and HTML for page requests)
    - Invalid endpoints (e.g. if someone types /api/logon/ instead of /api/login/) should return an appropriate error message in JSON format.
    - Appropriate HTTP status codes (e.g 200 "ok", 400 "bad request", 403 "forbidden", 404 "not found", 500 "internal server error") should be returned as part of each response from the API.

- The Documentation and About pages should be well designed
    - They could use appropriate CSS or be styled with Bootstrap.
    - They should be appropriately laid out and adjust fluidly or responsively to screen size
    - They must use valid HTML5

Part 2: Client Application created using React or AngularJS (50%)

For part two of the assignment you will make a client application that can present information about the conference based upon the JSON encoded responses from the Web API. This should be created in either React or AngularJS (you must choose one, you may not use a combination of both). If you choose AngularJS this must be v1.x and not Angular 2 or later. This application does not need to have a full set of features that visitors to the conference might need, but it should be of good quality and meet the criteria below.

- The client application must show the full schedule for the main conference (Monday – Thursday)
  - The detail of the schedule should initially be hidden. Users should be able to expand information according to what day of the conference they are interested in, what time slots, and which sessions
  - It should be possible to view the details of presentations in each session including the title, authors and abstract. Information about any awards should also be given.
  - For each session it should be possible to see the type of session, the session title, the session chair and the location (room).

- On a separate page to the schedule, the client application should be able to list all of the authors involved in the conference
  - There should be the ability to search the authors by their name.
  - There should be the ability to click on each author to see details of what presentations they are involved in.

- There should also be an admin page accessible to users who have authenticated with the API.
  - If a user has not authenticated they should see a login form, otherwise they should see a logout option
  - The JSON Web Token returned upon successful login should be stored in a cookie or local-storage.
  - For authenticated users, the admin page should list the title of each session in the conference, and enable details to be viewed.
  - It should be possible to update the title of any session.
  - Any authenticated user should be able to view the admin page, but only users with 'admin' set to true should be able to update the title of a session.

- The client must be written in React or AngularJS, HTML5/JSX and CSS.
  - The client application must be entirely separate to the API and communicate with it only via the published endpoints.
  - The client application should not use PHP and does not need its own copy of the database
  - The code must be appropriately indented and commented
  - Hand written CSS must be used. Frameworks including Bootstrap are not allowed for the client.

- The client application must be well designed
  - It should be styled appropriately
  - The application should be appropriately laid out and responsive to different screen sizes and should be suitable for use on a mobile device during the conference.
  - The application must meet accessibility guidelines.

Useful information

Database user table

username	email	admin	password
John	John@example.com	True	johnpassword
Kay	Kay@example.com	True	kaypassword
Jerry	Jerry@example.com	False	jerrypassword
Tanni	Tanni@example.com	False	tannipassword

Table 1: Content of users database table. The password in the database has been encrypted using the password_hash() function with default settings.

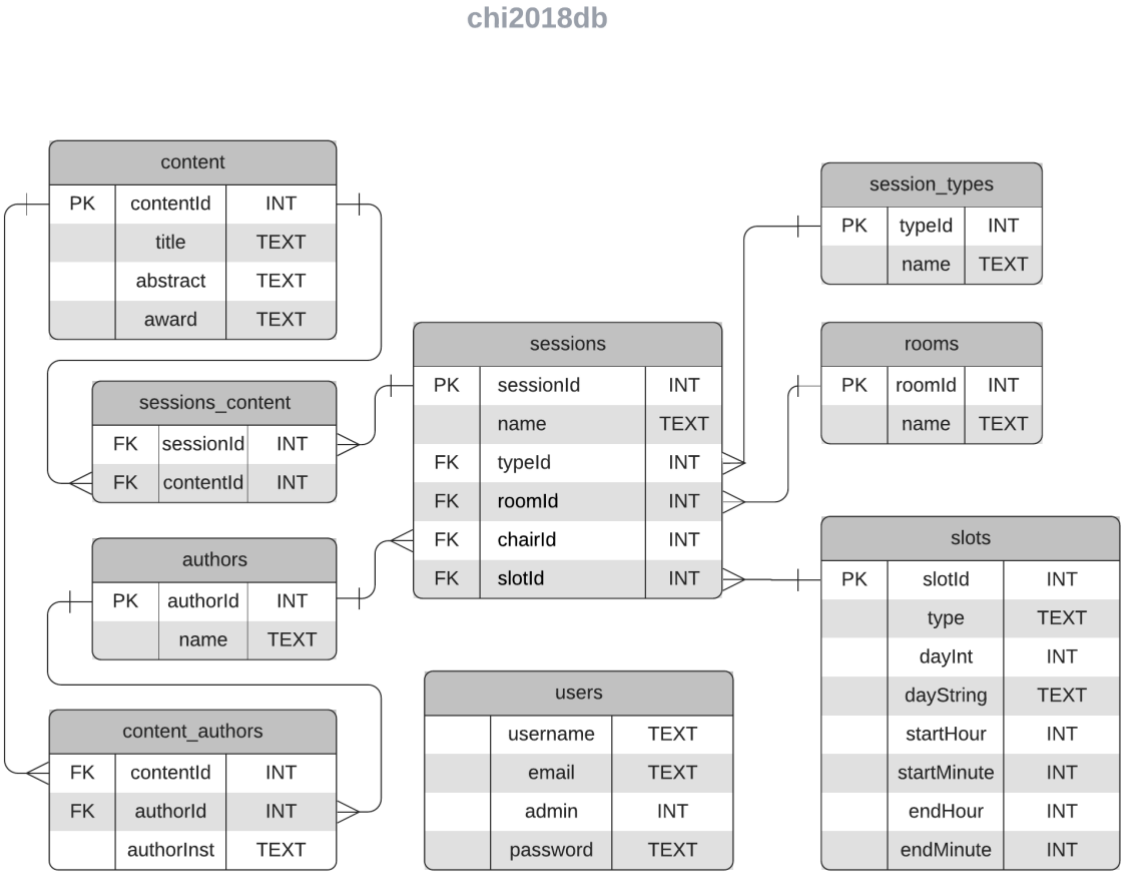

Figure 1: CHI2018db Entity Relationship Diagram

Notes

- The work will be marked using any of the following browsers: Firefox, Edge, Chrome.
- Part one and part two should be separate and should not have any interdependencies or shared code. All communication between part one and part two should be via your API endpoints. Both parts need to be submitted in separate folders within a single zip file.
- You can include a document with your submission called README.txt that contains relevant information about understanding and running your code. This should be in text (.txt) format. It is helpful if this contains a link to your application on newnumyspace.
- There is no one right way to design the endpoints for the API. You should think carefully about these and you can design them in a way that reflects the needs of the client application. When your API is marked we will look at your documentation page to find out what the endpoints are and how they are used.
- You may go beyond the specification for part 1 and part 2, and may be awarded extra marks for this, but only if the specification has been met in full. If you have added additional features please briefly state what these are in a README file.
- If you use @author tags in comments, you should add your own name and you may delete the name of the module tutor. If you submit code with the module tutor's name on, your marks may be awarded to the module tutor instead of you. The exception to this is if you use the JWT class or any third-party code that you have not modified. Please do not remove the @author tags or any license information from third party code.
- The CHI2018 conference has events from Saturday to Thursday, but you are just being asked to present data for Monday to Thursday. The workshops and doctoral consortium run on the Saturday and Sunday and do not need to be included on your site.

Submission files

You should submit one .zip folder containing your work. The zip folder title should be your student id number. Within that folder please submit part1 and part2 of your work. The way you organise your work is up to you but please keep your code for part1 and part2 separate. You must also provide a link to your work on newnumyspace. Please put this link in the README file.

Using Newnumyspace

Your work will be marked using Newnumyspace. You should make sure your work functions on Newnumyspace and you should provide a link. If you wish to use your own webserver you may do so, and you must provide a link to this. The webserver must be online for the full marking period. If using Newnumyspace your code must be compatible with PHP 5.6. The directory structure of and URLs for your project are up to you, but a suggestion is:

- http://unn-1234.newnumyspace.co.uk/KF6012/part1/ - For the documentation and about page for part 1
- http://unn-1234.newnumyspace.co.uk/KF6012/part1/api/ - For the API for part 1
- http://unn-1234.newnumyspace.co.uk/KF6012/part2/ - For the client application, part 2.

Note that while your config file and database would usually be above your public_html folder, they can be within the directory structure of part1 for the purposes of this coursework. Note that you must use the SQLite database and that this can be uploaded to Newnumyspace using FileZilla into an appropriate directory for part 1. You may not use MySQL, and therefore you have no reason to use PHPmyAdmin.

Mapping to Programme Goals and Objectives

The aim of this assignment is to allow you to adequately fulfil the following learning outcomes as specified in the Module Descriptor: (http://nuweb.northumbria.ac.uk/live/webserv/mod.php?code=kf6012)

- *Knowledge & Understanding*: Ability to develop a multi-tier system for data processing over the web using mixed data sources, including databases, taking into account security and transaction integrity.

- I*ntellectual / Professional skills & abilities*: Plan and manage a development project and critically evaluate tools, software architecture and technologies appropriate for it.

- *Personal Values Attributes (Global / Cultural awareness, Ethics, Curiosity) (PVA)*: Demonstrate professional and reflective practitioner attributes managing time and evaluating progress aiming for continuing development in the design, build, and testing of a secure web application.

Module Specific Assessment Criteria and Rubric

Part 1: 50%

Please note that you must meet all of the criteria within any band in order to gain a mark within that band. The first band (e.g. 18-20) represents outstanding work that exceeds the expectations of the module. The second band is first class work (i.e. worth 70%+).

Functionality (20 marks)

Mark	Criteria
18-20	The work is outstanding, meeting and exceeding the specification. The API has eight or more appropriate and functional endpoints, including a search, login and update endpoint. The API returns appropriate data in valid and well-designed JSON format including an HTTP status code. The data returned does not simply reflect the structure of the database but is appropriately designed for the needs of the client application. A limit can be set on the amount of data returned in a response (where appropriate), allowing data to be retrieved a 'page' at a time. The API is error and fault tolerant, returning appropriate error messages in JSON format including for exceptions. The API is extensively and clearly documented, and functions as described.
14-17	The work is excellent, meeting the specification. The API has eight or more appropriate endpoints, including a login and update endpoint. The endpoints are functional. The API returns appropriate data in valid and well-designed JSON format including an HTTP status code. The data returned does not simply reflect the structure of the database but is appropriately designed for the needs of the client application. The API is mostly error and fault tolerant, returning appropriate error messages in JSON format when appropriate. The API is clearly documented and functions as described.
10-13	The work is good / very good, mostly meeting the specification. The API has five or more appropriate endpoints but may not include a completely functional login or update endpoint. The endpoints are mostly functional. The API returns appropriate data in valid and well-designed JSON format. The data returned might closely resemble the structure of the database. An unnecessarily large amount of data may be returned from the API. The API is not fully error and fault tolerant, and error messages are not always appropriate and/or are not in JSON format when necessary. The API is documented and mostly functions as documented.
8-9	The work is acceptable, somewhat meeting the specification. The API has some appropriate end points and returns data in valid JSON format. The API does not handle all errors and faults properly, and returns inappropriate error messages to the client. The API is documented but some endpoints are not explained well enough to understand. Several endpoints might not be fully implemented.
0-7	The work is not acceptable. The API returns some JSON data but this is poorly designed or not valid. The API does not handle errors and faults at all. The API is not fully documented and is not easy to understand from the code. The API is barely useable by a client application. For a score of 0-3: the work does not contain a useable API that returns JSON data, or the JSON data is hard-coded. The API is not written in OO-PHP and/or does not use the SQLite database provided.

Code (20 Marks)

Mark	Criteria
18-20	The code is outstanding, written entirely in well-crafted and neatly organised object-oriented PHP. The code uses, but goes beyond patterns and principles described on the module. The quality of the code, including indentation, naming, and file structure is outstanding demonstrating thoughtfulness and consistency. The code clearly and accurately indicates what versions of PHP it is compatible with. Appropriate comments are used for all classes including @author tags, and for all methods including @param and @return tags where relevant. A secure config file in .ini or JSON format is used to store a range of relevant system information including the paths to resources and the database, and a secret key for use with the JSON Web Tokens. .htaccess files are also used securely and appropriately. Additional features have been added such as logging errors to an SQLite database.

14-17	The code excellent. It is written in object-oriented PHP. The code is well organised and uses patterns and principles described on the module including a router/controller, a recordset, singleton database connections, and an autoloader. The quality of the code, including indentation, naming, and file structure demonstrates thoughtfulness and is mostly consistent. Appropriate comments are used for all classes including @author tags, and for some methods including @param and @return tags where relevant. A secure config file in .ini or JSON format is used to store a good range of relevant system information including the paths to resources and the database and a secret key for JSON Web Tokens. .htaccess files are also used securely and appropriately.
10-13	The code is very good, making use of object-oriented PHP but also containing a lot of procedural style code (perhaps in index.php but also perhaps within classes). The code is well organised and uses some of patterns and principles described on the module including a router/controller, singleton database connections and an autoloader. A recordset class might be present but is not made full use of, and/or prepared statements are not used for querying the database. The quality of the code, including indentation, naming, and file structure is reasonable. Appropriate comments are used for most classes, and for some methods, but they may also be inappropriate inline comments or commented out code. A config file in .ini or JSON format is used to store some system information. .htaccess files are used appropriately.
8-9	The code is acceptable. It is written in object-oriented PHP. The code uses patterns and principles described on the module but with mistakes and limitations. The quality of the code, including indentation, naming, and file structure is acceptable but may not be consistently good. Comments are not always appropriate, for example inline comments are often used and/or there are sections of commented out code. A config file in .ini or .json format is not meaningfully used to store system information. .htaccess files are not used appropriately, for example the classes directory is not secured. The code may be written in a way that is specific to one computer, and does not also function on a server without modification.
0-7	There has been no serious attempt at object-oriented PHP other than for the database connection, or a framework such as Laravel has been used. For a mark of 0-3: No serious attempt at using Object Oriented PHP to output JSON data, for example a PHP based website has been produced.

Authentication and Authorisation (Server side) (10 Marks)

Mark	Criteria
9-10	Outstanding work, exceeding what is taught on the module. A username and password can be posted to an appropriate API endpoint. An external JWT class is used appropriately to encode a JSON web token which is returned to the client. The token includes appropriate information including an expiry date and time, and the admin status of the user. User credentials including the password are not included in token. The JSON web token is accepted by any endpoint that requires authorisation, and the expiry and user status are validated before any restricted actions can take place. The token has a short time to expiry (no more than a few hours) and these are handled gracefully.
7-8	Excellent work. A username and password can be posted to an appropriate API endpoint. A JWT class is used appropriately to encode a JSON web token which is returned to the client. The token includes appropriate information including an expiry date and time, and the admin status of the user. User credentials such as the password are not included. The JSON web token is accepted by any endpoint that requires authorisation, and the user status is validated before any restricted actions can take place.
5-6	A good/ very good attempt. A username and password can be provided to an API endpoint, but potentially in a flawed or insecure way. A JWT class is used to encode a JSON web token which is returned to the client. The token may not contain necessary or appropriate information and/or may have some inappropriate information. There is an attempt to validate user status before restricted actions take place.
4	There is an acceptable attempt. A JSON web token is generated, but not validated before restricted actions take place. The logic of issuing the token may have flaws, or the token itself may be invalid or have significant flaws.
0-3	Little or no attempt at using JSON web tokens. A token may not be properly issued or checked. An appropriate JWT class has not be used, or has been inappropriately modified such as removing the original authors' names or the licence information. PHP sessions are used.

Part 2: 50%

Please note that you must meet all of the criteria within any band in order to gain a mark within that band. The first band (e.g. 18-20) represents outstanding work that exceeds the expectations of the module. The second band is first class work (i.e. worth 70%+).

Functionality (20 marks)

Mark	Criteria
18-20	The client application is outstanding, meeting and exceeding the specification. The application communicates with the API (part 1) to present schedule information about the conference, allowing the user to view full details of the conference timeslots and sessions, including but not limited to: full details of the presentations, authors, the session chairs and session locations. The application has a page presenting detailed information about the authors at the conference and allowing for the names to be easily searched with fast, detailed and clear results (getting these directly from the API rather than using a client-side filter). Comprehensive details of the authors' presentations can be viewed. An outstanding user interface has been provided. The interface is well structured, consistent and easily navigable. An original design and colour scheme has been created. The functionality of the site is easy to identify and use. The site meets and exceeds accessibility recommendations. The app has a responsive design using media queries, with a design that is highly suitable for mobile browsers as well as desktop computers.
14-17	The client application is excellent, meeting the specification. It communicates with the API (part 1) to present detailed schedule information about the conference, allowing the user to view information about each timeslot and session, including details of the presentations, authors, the session chair and session location. The application has a page presenting information about authors at the conference, allowing for the names to be searched (using a client-side filter or the Search endpoint of the API). Details of the authors' presentations can be viewed, such as the titles of the presentations, and what session these are in. The interface is well structured, consistent and navigable. The functionality of the site is easy to identify and use. The app meets accessibility recommendations. The app has a fluid design making it suitable for use on mobile browsers as well as desktop computers.
10-13	The client application is good/very good, mostly meeting the specification. It communicates with the API (part 1) to present schedule information about the conference, allowing the user to view most of the information about each timeslot and session, including most details of the presentations, authors, the session chair and session location. The application has a page presenting information about authors at the conference, but these are not searchable and/or limited information about each author is given. The interface is appropriate but might be difficult to navigate and/or inconsistent. The app meets most accessibility recommendations. The app is suitable for desktop computers or mobile browsers but not both.
8-9	The client application is acceptable, meeting several parts of the specification. It communicates with the API (part 1) to present some schedule information about the conference, allowing the user to view some of the information about each timeslot and session. Some details of the presentations and authors are given but other information is missing or inaccurate. The application does not have a separate page/route listing authors. The interface is acceptable but is difficult to navigate and little attempt has been made to style the page, or the styling is poor. The app meets basic accessibility recommendations.
0-7	The work is not acceptable. No or very limited information is loaded from the API. Conference information is very limited or is displayed with major inaccuracies. The is little or no attempt to style information loaded from the API. There is some attempt at building a navigable website. For 0-3: Little to no attempt. No data is loaded from the API. The functionality provided does not use React or AngularJS (e.g. it is a PHP website). Logos or conference information have been used without written permission from the copyright holders, or information about the authors or their presentations is misrepresented (for example by changes to the database).

Code (20 marks)

Mark	Criteria
18-20	The client side code is outstanding, meeting and exceeding the recommendations on the module. The code is written in either React or AngularJS and does not attempt to combine the two. The code is highly modular, using well designed components or templates as appropriate. The increased size of the code base represents extra functionality introduced. The quality of the code, including indenting, naming and commenting is consistently outstanding. CSS is used in an outstanding way in conjunction with React or AngularJS. A router is used appropriately to allow navigation between several pages. A limited number of other packages may also be in use. State or scope is managed in an outstanding way.
14-17	The client side code is excellent, meeting the recommendations on the module. The code is written in either React or AngularJS and does not attempt to combine the two. The code is modular, using well designed components or templates as appropriate. The quality of the code, including indenting, naming and commenting meets the recommendations of the module. CSS is used in an appropriate way in conjunction with React or AngularJS. A router is used appropriately to allow navigation between pages.
10-13	The client side code is good/very good, meeting many of the recommendations on the module. The code is written in either React or AngularJS and does not attempt to combine the two. The code is somewhat modular, using some components or templates as appropriate. The quality of the code, including indenting, naming and commenting meets many of the recommendations of the module but is inconsistent of has problems such as unused components and commented out code. CSS is used in conjunction with React or AngularJS to style the page. A router is used to allow navigation between pages.
8-9	The client side code is acceptable, meeting some of the recommendations on the module. The code is written in either React or AngularJS and does not attempt to combine the two. The code is not very modular, holding most functionality in a small number of components or templates. The quality of the code, including indenting, naming and commenting is inconsistent, has problems such as unused components and commented out code. Not very much code has been written, or much of the code does not offer useful functionality. Some CSS is used in conjunction with React or AngularJS to style the page. There is only one route.
0-7	The work is not acceptable. The code is very basic. it cannot display data from the API, either because it does not communicate with it or because the retrieved data is not visualised. The CSS is sparse or is very poorly constructed. For 0-3: The code is not written in React or AngularJS (v1.x), or combines the two. The code is written using another JavaScript library/framework such as Angular (2+) rather than AngularJS. Alternatively, the build but not the source code has been submitted.

Authentication and Authorisation (Client Side) (10 marks)

Mark	Criteria
9-10	Outstanding, meeting and exceeding the specification. There is an admin page and a login component allowing the user to authenticate with the API. Once they are authenticated, any user can view a list of conference session names and details, and an admin user can update the name of a conference session. A non-admin user cannot update session names. A JSON web token is stored appropriately and is posted to the API when needed. The token is destroyed when the use logs out. Issues such as expired tokens are dealt with gracefully. Functionality exceeds the specification in a significant way such as enabling new user registration.
7-8	User authentication is excellent, meeting the specification. There is an admin page or component with a login form allowing the user to authenticate with the API. Once they are authenticated, any user can view a list of conference session names and details, and an admin user can update the name of a conference session. A non-admin user cannot update session names. A JSON web token is stored appropriately and is posted to the API when needed. The token is destroyed when the use logs out.
5-6	User authentication is good/very good, mostly meeting the specification. There is an admin page or component with a login form allowing the user to communicate with the API. There may be

	some problems with authentication such as limited functionality or some security issues. Once they are authenticated, any user can view a list of conference session names and details. It might not be possible to update data, or this functionality might not be restricted to admin users. A JSON web token may be returned but is not stored appropriately. There is no meaningful logout option.
4	There is an attempt at this functionality, such as a login form and an admin page with relevant features.
0-3	There is no meaningful attempt at this functionality, or the attempt is very poor.

ASSESSMENT REGULATIONS

You are advised to read the guidance for students regarding assessment policies. They are available online here.

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.

These provisions apply to all assessments, including those assessed on a Pass/Fail basis.

The full policy can be found via the student portal.

Academic misconduct

The Assessment Regulations for Taught Awards (ARTA) contain the ***Regulations and procedures applying to cheating, plagiarism and other forms of academic misconduct***.

The full policy is available via the student portal.

You are reminded that plagiarism, collusion and other forms of academic misconduct as referred to in the Academic Misconduct procedure of the assessment regulations are taken very seriously. Assignments in which evidence of plagiarism or other forms of academic misconduct is found may receive a mark of zero.