INFS3202/7202 Project Specification

Version: 1.3.1

Overview

The goal of this assignment is for you to implement a Web Information System. Your project can be different to your proposal though this is not recommended. You may do this assignment by yourself or in a group consisting up to three students (including yourself). It is highly recommended that you do not disband your team unless you have a good reason. You may use any software you wish as long as it is not banned (*see Ban List*) and meets the hurdle requirements of this assignment (*see Hurdles*). If you are in a group, please make sure everyone partakes in the development across all components of the 3-tier architecture, especially server-side programming. Groups of 2 or 3 members are required to complete a PAF survey after the assignment. Please take your time to fully read and understand this specification. There are many components to marking and calculations. Though PHP is not compulsory for your back-end development, your capability of using PHP will be assessed in the final exam.
The total score of the project is 100 marks, which is worth 35% of your grade

Submission

You are required to submit all necessary files related to the implementation of your project to blackboard in a .zip archive. The submission must take place prior to your demonstration.

You must submit all client-side code, all server-side code. If you use any libraries or packages, you may or may not need to include them depending on how your code imports said libraries. You also must submit an export of your database (of appropriate size) as a text file.

You must submit your project with the requested files otherwise you will not receive any marks for this assignment.

In a file somewhere within the same submission, include a reference of all libraries you have used. You do not have to reference code if you have used to only assist you in understanding how to achieve some functionality. You MUST reference code if you have copied it into your project

You do not submit your reflection (i.e., individual final report) with the project (this assignment). The specification for the reflection will be released later.

The teaching team reserves the final right to interpret the content of the specification.

Demonstration

Should not be confused with Outstanding Projects Showcase. As part of marking, you are required to demonstrate, in person, to teaching staff your project. Be prepared to demonstrate functionality that teaching staff request, show any source code and answer questions related to the implementation of your project. Demonstrations will take place in Week 13 and you are required to register a time for your demo session.

You must demonstrate your project otherwise you will not receive any marks for this assignment.

Outstanding Projects Showcase

Should not be confused with Demonstration. Projects that are compelling will be nominated by teaching staff for presentation in the week 13 lecture. More details will be released with an update to this spec or in a future blackboard announcement.

Hurdles

As mentioned within the proposal, trivially implied and explicitly mentioned, some functionality must be achieved in order to qualify full marks. You must pass these to avoid a penalty cap. The implementation quality is not assessed in this part, rather if there has been a valid attempt on the components.

Penalty caps are additively applied. Your final score is calculated as follows:

$$marks = \max(\min(\text{score}_{\text{total}}, 100 - \text{penalty}_{\text{total}}), 0)$$

For example:

If you achieve a score of 78 but you don’t have users then the highest possible mark you can get is max(min(78, 100-50) , 0) = max (50, 0) = 50.

Task	**Description**	**Penalty Cap**	**Detail**
Supports users	The Web Information System must somehow support users.	50	• The Web Information System must support user authentication and user authorization. ○ The means in which user authentication and user authorization is accomplished **is compulsory** to this hurdle.
3-Tier Architecture	You must use 3-Tier Architecture of some sort	100	• You must use a 3-tier architecture. • The 3-tier architecture: ○ You must have client that can access your WIS and communicate with your WIS’s server

			○ You must have a server that can communication with your client and database ▪ You may use Serverless Code services (AWS Lambda) ○ You must have a database that stores data in a persistent manner, and it must communicate with your server. ▪ **In-Memory Databases are not permitted (Redis).** • Submission must contain: ○ Client-side code that was demonstrated during marking ○ **For Mobile Application Clients:** When you are about to submit your code to Blackboard, add an additional attachment to the same submission containing the application archive file (APK file for Android and IPA file for iOS). ○ Server-side code that was demonstrated during marking ○ DevOps, operations and provisioning scripts are not to be included and will not be marked (unless it is used to generate or compile database or client-side source code). ○ Database code that was demonstrated during marking. ▪ If you used phpMyAdmin, the instructions for exporting your database will be released later on Blackboard. ▪ If you are using a NoSQL database, include as little rows as possible to demonstrate your schema. (Do not submit your database with no rows). ▪ If you used ORM (Object Relational Mapping), you must include all code relevant. ▪ If you have any grants, they must be included.
Source Code Walkthrough	Whilst demonstrating your project you should have a copy of the source code.	1, 2, 5, 10, 25 50	• You must be able to explain your source code during your demonstration. • Each occurrence that you are not able to explain source code will attract a higher penalty cap. ○ For example, if you fail to explain your code three times, you will receive a 5-mark penalty cap. The questions can be answered by any group members. • All the group members should attend the demo session.

Deployment	Your system must be deployed to a public cloud	40	• The project should be deployed on UQ Zone or any other public cloud platforms (e.g. AWS, Azure). • DevOps, operations and provisioning scripts are not to be included and willnot be marked (unless it is used to generate or compile database or client-side source code).

1. Database

Task	**Description**	**Mark**	**Detail**
Database Usage	A database has been used	4	• Can be relational or NoSQL.
Schema Size and Complexity	Database consist of at least 3 tables	6	• Your database must have at least 3 tables. • Tables should be populated with some data prior to demonstration. • CRUD Create, Read, Update, Delete) operations have been used.

2. Web Server (Backend)

Use of a server-side language such as PHP to achieve functionality such as, but not limited to: session control, login methods and database connections.

Task	**Description**	**Mark**	**Detail**
Server Features	The server must have sophisticated functionality	27	• The server must have a moderate amount of sophisticated functionality • Complex functionality is regarded as functionality beyond CRUD operations and code used to communicate between 3-tier architecture hierarchies (both client-server and server-database). • This could be for example (but not limited to): calculations, file manipulation, third-party API integration, integrations with other services, sending emails, RSS feeds and web sockets. • Functionality pertaining to user authorization and authentication does not apply here but applies to *Security*. • For any server features to count, it must work (not cause an exception), it must be correct (values and output of feature is correct) and be of academic merit (a feature that is substantially more in-depth and sophisticated than: a test, a proof of concept or a “Hello World”). • **See Server Features**

Server-Client Communication	The server and client serialize data with a standardized protocol.	3	• Communication between Server and Client should use JSON or XML technologies.
CRUD Functionality and Database Connection	The server must have CRUD functionality (beyond user login and registration).	10	• Your backend must support database CRUD operations. • The connection to the database must be used as part of the database CRUD functionality. • CRUD operations must use a database connection.
Error Handling and Bugs	The server should handle faulty data and input appropriately.	10	• The server should reject invalid input and respond accordingly ○ The sever shouldn't crash nor should it raise an unhandled exception. • The sever should accept valid input and respond where appropriate. • The server should not allow invalid input and should not reject valid input.
Code Modularity	Common code should be modularized into reusable parts. There should be little to none code duplication	5	• Code should be modularized into reusable parts. ○ A common function or method used by other code. • There should be little to none significant code duplication. (0 - 4 duplications) ○ Duplications will be identified with the tool: PMD (pmd.github.io) ○ PMD (CPD mode) will run with a minimum 300 tokens. Duplication of 300 tokens would be considered significant code duplication with a very high possibility that the offending code could be made modular. ○ The marking tutor has discretion to ignore instances of duplication where appropriate. ▪ False positives ▪ External library code.

2.1 Server Features

You are expected to implement several features for your WIS. The marking criteria for each major feature is defined as follows. **You can achieve up to a maximum of 27 marks. At the marking tutor's discretion, features can be promoted in complexity where credit is due. (when they deserve it)**

For any server features to count, it must work (not cause an exception), it must be correct (values and output of feature is correct) and be of academic merit (a feature that is substantially more in-depth and sophisticated than: a test, a proof of concept or a "Hello World").

Multiple features that use the same technology (or same set of API's in the case of CRUD operations with a third-party service) are not counted twice.

Warning!
Examples are subject to change and not definitive that they fall under a certain complexity.

Complexity	Simple (5 marks)	Intermediate (8 Marks)	Advanced (13 Marks)
Description (Generalisation and not completely definitive)	Generally: (turn-key) • A small amount or trivial code written by you. Potentially with a library doing most of the work **OR** • Using the code written by INFS3202/7202 teaching staff **OR** • CRUD operation with a third-party service	Generally: (libraries and plug-ins) (cannot be code written by INFS3202/7202 teaching staff) • A small to medium amount of code written by you. Potentially with a library as means **OR** • Requires shallow understanding of esoteric knowledge beyond the scope of this course	Generally: (libraries) • A medium to a large amount of code written by you **AND** • Requires succinct understanding of esoteric knowledge beyond the scope of this course. **AND** • Cannot be code written by INFS3202/7202 teaching staff

Examples	• Sending Email from own web server • Using third-party service integration ○ YouTube Data API (Getting uploaded videos) ○ Google Calendar (Getting next 10 events) ○ SMS as a service (A company sends email on behalf of our website) ○ Payment Providers (Paypal) • Simple Image Manipulation ○ Compression ○ Cropping ○ Resizing • Simple Video manipulation ○ Video Thumbnail generation	• Non-text file generation and manipulation ○ PDF file generation ○ Sound file manipulation • Web Push Notifications (Server Code) • Advanced non-trivial algorithms without a library. • Web sockets (server-side) (incompatible with UQ Cloud) • Intermediate Image manipulation ○ Intermediate Filters (Grayscale)	• Advanced Image manipulation ○ Advanced Filters (e.g. Red-eye Removal) • Web Scraping • Computer Vision • Data mining (e.g. Using Collaborative-Filtering Model for goods recommendation on an online shopping website) • Machine Learning (pretrained) • Artificial Intelligence (pretrained)

If you have any questions about this scheme or if you would like us to assess a complexity, please contact teaching staff.

2.2 Bonus Sophistication

Any usage of the following will be awarded marks for each occurrence.

For any bonus sophistication of server features to count, it must work (not cause an exception), it must be correct (values and output of feature is correct) and be of academic merit (a feature that is substantially more in-depth and sophisticated than: a test, a proof of concept or a "Hello World").

Description	Examples	Bonus
One or more usages of MVC, REST, other service oriented architecture frameworks to drive server-client and database-server communication. Some frameworks may provide more than one of applicable technologies. In this case it will still count however only. **For a framework to be of academic merit: it must be used to drive a significant amount of CRUD (Create, Read, Update, Delete) operations.**	PHP: • CodeIgniter • Laravel ASP.NET: • ASP.NET MVC • ASP.NET Web API	5 Marks

3 Front-end

3.1 Desktop Web Site:

If you choose to have a desktop web client, you cannot have a mobile client.

Task	Description	Mark	Detail
Client Features	The web page needs to present sophisticated user interaction	7	• The combination (up to 7 marks) of: ○ Third-party widgets ○ Web Interface powered by the 'deep' use of JavaScript ○ Complex Client-side coding • For any client features to count, it must work (not cause an exception), it must be correct (values and output of feature is correct) and be of academic merit (a feature that is substantially more in-depth and sophisticated than: a test, a proof of concept or a "Hello World"). • **See Client Features at the Bottom**
Intuitive Navigation	The design of the web page needs to have intuitive navigation	2	• The website must be intuitive to navigate for a person from a non-IT background who is competent at using a web browser and computer but has never seen or used your website before. ○ No broken links ○ All elements that look actionable should have action. • Appropriate amount of user prompts, instructions, notifications and message.
Neat Presentation	A clear interface and the website must have a neat and sensible design	4	• Consistent colour palette • Sensible layout and navigation • The visual style for the website is appropriate for the theme and audience
Agnostic Browser Support and Responsive to Screen Size	The Website must be supported across commonly used browsers and Web Sites responds to varying common screen sizes.	4	• The desktop website must support the following modern web browsers. ○ Google Chrome (Version 64.0.3282.167 or greater) ○ Firefox (Version 53.0.3 or greater) • Web site features should not be specific and applicable to a browser. • All features should be on parity across browsers. • Your website must fit and look appropriate to resolution (inclusively) between 1280x720 pixels and 1920x1080 pixels even during resize. (Simulated at 90-110 DPI) • The website should not need to refresh upon resize.

AJAX and Client-Server Communication	Your website must make use of AJAX technology	8	• You must have at least two different uses of AJAX on the website.
Bonus: Trendy and Modern Design	Your website looks beautiful	(6)	• This is a subjective and as such a bonus mark. You can get full marks on the assignment without getting a single mark in this criterion. • It only counts towards the summation of Desktop Client Side marks • Tutors cannot provide feedback on this criterion • **This mark cannot be argued. Teaching staff decisions are final.** • In general: the visual design is engaging

3.2 Mobile client

If you choose to have a mobile client, you cannot have a desktop web client.

We cannot supply phones for you to develop, test and demonstrate on. We have made available Android Studio 3 on all lab computers including the Android Emulator. If you choose to create an Android application, it is highly recommended you use a real mobile device or the Android Emulator to develop and demonstrate your application.

We cannot supply Apple devices for developing iOS mobile applications. If you choose to develop for iOS, we are assuming you have your own Mac.

Task	**Description**	**Mark**	**Detail**
Client Features	The mobile app needs to present sophisticated user interaction	7	• The combination (up to 7 marks): ○ Third-party widgets ○ Deep integration with OS-level features. ○ Complex Client-side coding • For any client features to count, it must work (not cause an exception), it must be correct (values and output of feature is correct) and be of academic merit (a feature that is substantially more in-depth and sophisticated than: a test, a proof of concept or a “Hello World”). • **See Client Features at the Bottom**

Intuitive Navigation and Neat Presentation	The design of the mobile app needs to have intuitive navigation and a clear interface and the mobile app must have a neat and sensible design	6	• The mobile application must be intuitive to navigate for a person from a non-IT background who is competent at using the device you are targeting but has never seen or used your app before. ○ Gestures native to platform ▪ Left edge screen swipe to go back on iOS ▪ Back button on Android ○ Contextual Menus native to platform • Appropriate amount of user prompts, instructions, notifications and message. • Consistent colour palette • Sensible layout and navigation
Supporting different platform versions	Mobile app works on different versions of platform.	2	• Your mobile app should support different platform versions. ○ For IOS apps: you need to support at least IOS 10.0 or later. ○ For Android apps: you need to support at least Android 4.3 or later.
Supporting different devices scareen sizes	Mobile app works on different screen sizes.	2	• Your mobile app should support different screen sizes. ○ (E.g. On iOS: iPhone 8 and iPhone X screen sizes) ○ (E.g. 16:9 and 2:1 resolutions on Android)
Client-server Communication	Your app must be able to communicate with a server	8	• Your app must be able to retrieve and publish data with your server.
Bonus: Trendy and Modern Design	Your mobile app looks beautiful	(6)	• This is a subjective and as such a bonus mark. You can get full marks on the assignment without getting a single mark in this criterion. • It only counts towards the summation of Mobile client marks. • Tutors cannot provide feedback on this criterion. • **This mark cannot be argued. Teaching staff decisions are final.** • In general: the visual design is engaging

3.3 Client Features

There are 4 distinct categories (rather than levels of complexity) of sophisticated user interaction that count towards this criteria. Some apply to both desktop web and mobile application client-side. For a feature to count, it must correctly work. In the case of desktop web browser web clients, it must work on both Firefox and Chrome (versions are stated in *Agnostic Browser Support*).

For any client features to count, it must work (not cause an exception), it must be correct (values and output of feature is correct) and be of academic merit (a feature that is substantially more in-depth and sophisticated than: a test, a proof of concept or a "Hello World").

You can only obtain a maximum of 7 marks. Multiple features that use the same technology or same set of API's (in the case of CRUD operations with a third-party service) are not count twice.

3.3.1 Third-Party Widgets

This applies to both mobile and desktop

Client-side functionality that uses a library or script to integrate a single third-party (not authored by you) element to your webpage. Examples: Lightbox, Google Maps, jQwidgets. The work you must do to setup the third-party element, process data to display in the widget and configure it to work does not double up.

For each unique usage of a widget used successfully, Tutors will determine their complexity and award marks. Marks you get count towards Client Features.

Complexity	Simple (1 mark for each unique usage)	Advanced (2 marks for each unique usage)
Description (Generalisation and not completely definitive)	Configuration to use third-party element to work with the system is simplistic and only involves setting up variables. Code can be examples from teaching staff.	Configuration to use the third-party element to work with the system is involved and requires some scripting or processing authored by you. Code cannot be examples from teaching staff. Note: Whilst a third-party widget/element, with quickstart turn-key setup, would be regarded as simple, customisation of the widget/element can bring it up to advanced.
Examples	Lightbox	Sophisticated use of Google Maps

3.3.2 Web Interface

This only applies to desktop web browsers clients.

Tutors will assess the interface for the entire web site and determine the depth. You will receive marks in this Client Features category once. Marks you get count towards Client Features.

Depth	None (0 marks)	Shallow (1 marks)	Deep (2 marks)
Description	Navigation is primarily done with hyperlinks. JavaScript is not used or minimally used to drive the web site interface.	Navigation is achieved with simple menus and interface implemented with JavaScript however interactions with these interface elements is static and just hyperlinked.	Navigation is achieved with menus and interface elements with JavaScript. Elements provide some interaction such as hints, validation, autocomplete or drive some functionality.

3.3.3 Mobile Operating System Integration

This only applies to mobile clients.

Tutors will assess the interface for the entire app and determine the depth. You will receive marks in this Client Features category once. Marks you get count towards Client Features.

Depth	None (0 marks)	Shallow (1 marks)	Deep (2 marks)
Description	No integrations with the mobile operating system	Simple integrations with the mobile operating system usually involving simple configuration alterations.	Integration into mobile operating systems that require work beyond trivial configuration and simple function bindings.
Examples	Nothing.	Notifications	Android Widgets or Control Centre Widgets Chromecast

3.3.4 Scripting and Processing

This applies to both mobile and desktop.

Any form of complex client-side coding. This is similar to sophisticated server functionality but on the client-side. Marks you get count towards Client Features.

Complexity	Simple (1 mark)	Intermediate (2 marks)	Advanced (4 marks)
Description	Generally: (turn-key) • A small amount or trivial of code written by you. Potentially with a library doing most of the work or the library already setup **OR** • Code written by INFS3202/7202 teaching staff	Generally (cannot be code written by INFS3202/7202 teaching staff): • A small to medium amount of code written by you. Potentially with a library as means **OR** • Requires shallow understanding of esoteric knowledge beyond the scope of this course	Generally: • A medium to a large amount of code written by you **AND** • Requires succinct understanding of esoteric knowledge beyond the scope of this course. **AND** • Cannot be code written by INFS3202/7202 teaching staff
Examples		HTML 5 Canvas / Android Canvas	Web assembly

4. Security

Applies to both Desktop Web Browser Client and Mobile Application Client

Task	**Description**	**Mark**	**Detail**
Access Control	Account dependent functionality must be protected by user authorization.	2	• All private functionality that is limited to users must have proper authentication to use.
SQL Injection Mitigation	Prevent foreign SQL	2	• Some sufficient mechanism, system, method or design pattern to have a reasonable mitigation against SQL injection attacks. • Hint: Client-side data sanitation is not enough
SSL Encryption	Communication to the Web Information System must be done with SSL encryption.	2	• The client connection to the server must be encrypted using TLS/SSL. • The certificate must be valid on the day of your demonstration.
Password Protection	Any stored passwords are protected by sufficient cryptographically-secure means.	2	• If storing passwords, current best practices should be used. ○ You must provide reference or source as to how you are storing passwords. (How you store your password must also match the reference you provide) ○ This can include: web server documentation, Stack Overflow answers ○ Usage of hash stretching algorithms such as bcrypt or PKDBF2 • Your authentication system or authentication flow should use best practices.
Submission does not contain keys and passwords	The submitted project does not contain keys and passwords	2	• The submission made to Blackboard does not contain any passwords or API keys. • This includes any passwords of any kind. • Your submission may differ from the actual code running on the website. ○ If you have a hardcoded password string, you may change to an empty string. ○ If you set passwords by environment variables, you do not have to submit your profile (which contains environment variables). ○ If you submit a NoSQL database and store user credentials, omit passwords as part of the submission.

Appendix 1: Ban List

Using software from any of the listed categories are banned as they warrant no academic merit defeating the purpose of this assignment. Usage of these software will result in you receiving no marks for this assessment.

If you use software from this list, you are guaranteed to get 0. This list however is continuously updating and not exhaustive. We could deem your system without academic merit on the day of demonstration

In general, any software that builds a system for you is banned. If you are unsure, check your software past teaching staff and they will notify you if you can use it.

External systems like third party APIs or external systems are not part of the system and therefore do not count to this ban list.

Software Categories	**Examples**	**Reason**
Cloud-based web development platforms and Wikia	Wix, Squarespace, Wikia	Software from these categories are used to build websites quickly using a web interface. They warrant no academic merit since all client, server and database functionality is done on a website GUI. Such technology circumvents assessable demonstration of learning objectives.
Content Management Systems	Drupal, Wordpress, Joomla!	Many criteria can be easily achieved with in content management systems. Some are even turn-key solutions available in WordPress. Usage of such systems circumvents assessable demonstration of learning objectives.
Hosted Content Management System	Wordpress.com	See Content Management Systems.
Advertisement and Client-side Crypto Currency Mining.	Google Ads, Coinhive	We do not want you take advantage of our systems to gain financial revenue when we mark or visit your project.
Mobile operating systems not Android and iOS	Windows Phone, Blackberry OS, Ubuntu Mobile, Firefox OS, Tizen OS	These operating systems have a negligible market share if any. It will be too difficult to support and mark mobile application targeting these platforms.

Appendix 2: Special Rules

Some software or features have special rules to marking. These rules have higher precedence than the mark sheet. These rules are in place to overcome technical and practical limitations as well as bring fairness.

Software Categories	**Examples**	**Rule**	**Reason**
Payment Providers	Paypal, Masterpass, Visa Checkout	You must use sandboxed versions of integrations with third party payment providers as we will not be testing your integration with actual payment. The sandboxed version is not a version you create. It is rather an alternative service that the payment providers make for developers that allows mock transactions to take place without any financial consequence.	We will not be making any financial transactions during marking. All transactions are to be mocked.
Web Sockets	Web Sockets	Despite web-sockets being a client-server communication technology, it will be regarded as server features as well as client features	The amount of work to accomplish this is non-trivial from our baseline expectations.
Non-Mobile Hardware	Chromecast	We have no means of testing this and we cannot supply special hardware for development and demonstration purposes. If you wish to demonstrate such feature using non-mobile hardware, you'll have to supply it yourself. We are not liable for anything that happens to special hardware that you bring in.	It is not feasible for us to provide this to every student.
Emails used for verification purposes	Confirmation of user registration to WIS	This will count as a server functionality.	
Libraries that can extract prominent colours from images.	Google palette API,	If and only if colour extraction libraries are used for the purpose of setting the interface colour scheme, then the "Consistent colour pallet" clause in Neat Presentation (desktop) and Intuitive Navigation and Neat Presentation (mobile) will not apply to affected interface elements	Modern design trends are altering interface elements based on the content being displayed. To adapt for change, this exception is applied.

Updates:

Version 1.0 → Version 1.1:

- Removed Client, Server and Database hurdles in favour of one hurdle: “3-tier architecture”
  - Penalty cap worth 100 marks. (You will not get any marks if you don’t meet its requirement)
- Added “code cannot be examples from teaching staff” clause for Advanced third-party widgets from Client Features (formerly Sophisticated User Interaction or Sophisticated Client-side Coding).
- Added “cannot be code written by INFS3202/7202 teaching staff” to - Scripting and Processing Intermediate and Advanced categories of Client Features (formerly Sophisticated User Interaction or Sophisticated Client-side Coding).
- Added data mining example to Advanced Sophisticated Server Functionality.
- Added information about submission regarding submission not containing passwords and API keys.
- Added clause layout must look appropriate during resize on Desktop Web Clients.
- **Renamed ‘Sophisticated Server Functionality’ to ‘Server Features’**
- **Points system for Server Features (previously known as Sophisticated Server Functionality) removed in favour of actual marks.**
- **Marks for Server Features (formerly Sophisticated Server Functionality) have been rebalanced.**
- **Renamed ‘Sophisticated User Interaction or Sophisticated Client-side Coding’ to ‘Client Features’**
- **Points system for Client Features (formerly Sophisticated User Interaction or Sophisticated Client-side Coding) removed in favour of actual marks.**
- **Marks for Client Features (formerly Sophisticated User Interaction or Sophisticated Client-side Coding) have been rebalanced. See new marks**
- Reworked descriptions for Client Features in both Desktop Web and Mobile client criterion (formerly Sophisticated User Interaction or Sophisticated Client-side Coding). The clearer descriptions by themselves do not affect marking.
- Added clause about having a feature to be marked. For a feature to count, it must work (not cause an exception), it must be correct (values and output of feature is correct) and be of academic merit (a feature that is substantially more in-depth and sophisticated than: a test function, a proof of concept or a “Hello World”)
- Special rule added for emails being used for user verification.
- Added clause about double up on functionality.
- Minor typos that do not affect marking (appropriate spelling, incorrect usage of plurals or singular)
- Added clarification to hint about Hash and Salt in security

Version 1.1 → Version 1.2:

- Change demonstration requirements of password protection.
- Removed password storage hint
- Updated Server Feature example list

Version 1.2 → Version 1.3:

- Typos with homonyms
- In 3.1 Desktop Web site, moved “Appropriate amount of user prompts, instructions, notifications and message” bullet point from Neat Presentation to Intuitive Navigation
    - Mark and weight have not changed
- Defined which tool will be used to measure code quality and the threshold to obtain to full marks in this criterion.
- Reworked Bonus Sophistication in Server Features:
    - Included working, working correctly and academic merit clause. Added clarification to framework
    - Changed bonus mark scheme for MVC frameworks to also include REST and other service oriented architecture frameworks
        - Extended provision to REST and service oriented architecture.
        - It is still worth the same 5 marks
        - Included explicit working, working correctly and academic merit clause to bonus sophistication. Specific language added for MVC/Rest frameworks.
- Clarified Error Handling and Bugs in Sever criteria
    - Added clause that invalid data doesn’t cause a crash nor should it cause an unhandled exception.
- Clarified contradiction in Client-Server Communication criterion in Front-end (Mobile)
    - There must be bi-direction communication between Front-end (mobile) and Server
- New Colour Palette special rule
- More details in password storage pertaining to how passwords should be stored
- Added details to submission process. Must reference external libraries or code copied.
- **Changed code modularity marking method to be objective.**
- **Added rule that features can be promoted in complexity at tutor’s discretion.**

Version 1.3 → Version 1.3.1

- Added simple, intermediate image manipulation
- Advanced image manipulation clarified.
- Added UQ Cloud compatibility issue with Web sockets
- Added SMS as a service
- Added Simple Video manipulation
- Added PayPal
- Demonstration date update.
- Made deployment requirement explicit