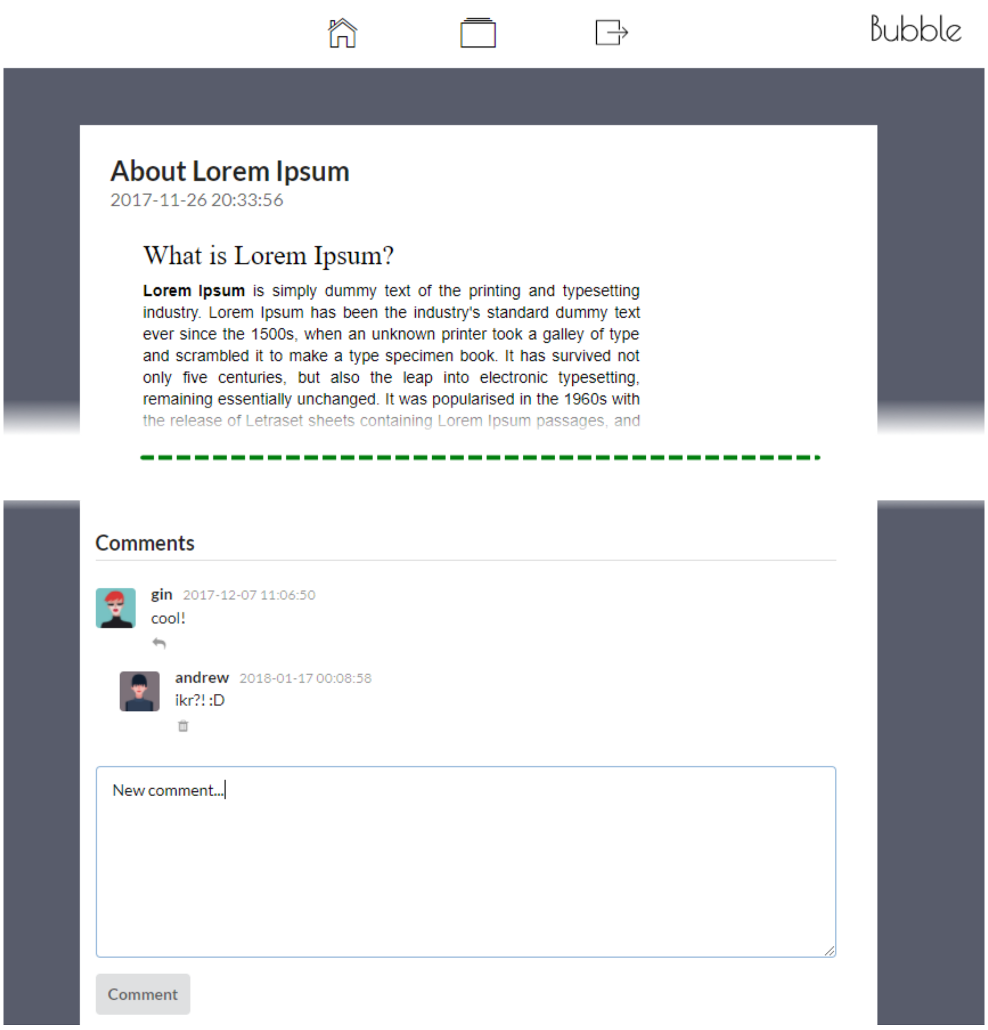
PGCert in Information Technology

Final Project - A Personal Blogging System



Bubble

About Lorem Ipsum
2017-11-26 20:33:56

What is Lorem Ipsum?

Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and

Comments

gin 2017-12-07 11:06:50
cool!

andrew 2018-01-17 00:08:58
ikr?! :D

New comment...

Comment

Fig1: Demo blogging platform. Credit: "Bubble Blog" (Emma Zhao, Tongxin Xie, Qian Peng, Xingjian Che)

Key Dates

- **18th of May:** Project Introduction
- **1st of June:** Group project source code due
- **2nd of June:** Group presentations
- **6th of June - 12th of June:** Working on individual project
- **12th of June:** Individual project code due
- **15th of June:** Individual report due

Introduction

In this project you will develop a blogging website using the skills and technologies you have learnt through the *Programming for Industry* and *Programming with Web Technology* courses. The project will also give you the opportunity to show how you can use online resources to discover and apply content *not* taught within the course. The snapshot shown above is taken from a previous PGCert IT student group's submission.

Through the web site, users can register for an account, which is needed to be able to post articles and to leave comments on others. When logged in, they have full control of the content they have authored: creating, updating and deleting their content and comments.

In addition to the website you build for the group project, a Swing-based administrative interface will also be created as an individual project, allowing admins to view and delete user accounts. You will work on the individual project after the group project is completed.

You are given a list of requirements for the blogging system. Your group will also have the opportunity to customise aspects of the project, within the broader requirements.

The project will give you the opportunity to work as a team on a larger-scale project than you've had experience with previously in the course. It will also allow you to showcase your individual software development skills.

Project Overview

The final project is worth **30%** of your total grade. It is broken down into three parts, as follows:

- Group implementation and presentation: **15%**
- Individual implementation: **10%**
- Individual report: **5%**

Group implementation and presentation **(15%)**

The team portion of this project consists of **100 marks total**, and is worth **15%** of your final grade for both the *Industry* and *Web* papers in the certificate.

While it is a group project, individual group member's marks are dependent on participation in the group, contribution to the project and attendance at daily standup meetings. We do put careful consideration into marking to make sure groups are not disadvantaged in the case that a member is not able to participate for any reason. We do not expect everybody to have the same technical ability but participation and collaboration are essential. Planning goals and the contributions of each group member in daily standup is essential. It is also important to get to know the members of your group so that you can collaboratively plan tasks that each member can contribute to.

The marks for the group project are comprised as follows:

	Marks
Feature implementation	*75 marks*

Code quality	*10 marks*
Daily stand-ups / project management	*10 marks*
Presentation & demonstration	*5 marks*

Feature implementation **(75 marks)**

There are several requirements for each team's blogging system. These are given in the **Group Requirements** section below. All requirements must be implemented to a high standard to receive full marks for feature implementation.

Code quality **(10 marks)**

Your team's code must be easily understandable by third parties, and conform to best practices. This includes the use of appropriate variable and identifier names, sufficient commenting, and breaking your code up into appropriate modules, amongst other considerations. It should be written in a way that would make it easy for other people to understand and modify.

You should use code organisation and quality techniques covered in the course content where possible and appropriate. This would include use of modules, modular routing, middleware, design patterns, refactoring, and testing where possible. Also consider these general principles when structuring your CSS and HTML etc. This helps ensure that all students have an equal opportunity to participate in the project and understand the code group members are producing. Consider carefully how you can structure things like DAOs, CSS, Handlebars, JavaScript files etc so that you can effectively share and reuse code.

The group project is designed to provide an opportunity to apply the technologies and concepts taught in CS719. It is expected that you will use the general approaches and principles taught in CS719 to approach the implementation. While you may integrate some code from other sources for specific components like the WYSIWYG you should not use any premade templates or frameworks for larger parts of the project. If you have doubts about what tools are appropriate to use, ask one of your lecturers.

Daily stand-ups / project management **(10 marks)**

Each weekday during the project, your team will be required to report progress to the lecturer or tutor. Each team member must be present at every meeting. The meetings will be short - approximately five minutes per team per meeting. Evidence of good teamwork is required at these meetings - the instructor should be able to see and agree upon a fair workload allocation for each team member. Evidence of good project planning should also be visible in these stand-ups, via a team Trello board.

For more information regarding important project management aspects to consider, see the **Project Management** section of this document.

Presentation & demonstration **(5 marks)**

On the presentation date (see **Key Dates** on the front page of this document), each team will get the opportunity to present their project to the rest of the class, and other staff and guests. This is your chance to show off all your hard work and it should be a fun session.

Each team will be allowed 15 minutes to speak, followed by some questions. Presentations should be professional, be slideshow-based (i.e. attempting to do a live demo is *not* allowed), and focus on the overall system architecture and features of your web app, rather than delving into low-level implementation details. Each member of the team is expected to speak during the presentation, as well as answer questions during Q&A. After the presentations, we encourage groups to have a go at using and testing each other's projects on the computers in the lab.

Individual implementation **(10%)**

After your group has completed the group project, you will implement a Java Swing-based application which communicates with the backend of your group project implementation in order to provide some administrative user management features.

The individual implementation will be built to connect to your group project and we will be providing some resources to help with the individual implementation when the group project is submitted so it is suggested you start the individual implementation then.

75% of the marks for this implementation will be awarded for completing the requirements set out in the **Individual Requirements** section below, while the remaining 25% will be awarded for good code style and appropriate use of applicable design patterns.

Individual report **(5%)**

In addition to the group and individual implementation, each **individual** must submit a written report on or before the report due date (see **Key Dates** on the front page of this document).

It is suggested that you work on the report throughout the group and individual project and use it as an opportunity to do some extra reading related to the principles and technologies you are using. The report must cover the following topics:

- In your own words, explain how the system as a whole has been designed; the overall system architecture and how various components fit together and communicate to form the whole.
    - Brief overview of the system and architecture design.
    - Highlight targeted problems and solutions.
- Detail your particular contributions to the team and project; you are expected to have worked on the front-end and back-end so outline what you did on both.
- Detail your work on your individual project.
- Detail which topics, taught in class, have been used within the project, and where.
- Detail any topics or technologies, which were not taught in class, that you have used, and where.
- Describe the lessons you have learned from working in a team. What are the benefits, and are there any drawbacks or difficulties? How have those been overcome in your team?

The length of the report should be *approximately* **four pages** in **IEEE two-column format** (though there is no minimum or maximum limit). Word and LaTeX templates for the report are available here.

The report should be written as a structured piece of academic writing; you should follow the formatting and conventions of the IEEE format as linked above. You should use well written paragraphs as much as possible and only use diagrams and bullet points where appropriate. You should introduce and conclude sections of the report appropriately along with using clear subtitles for the different sections of the report. If you have not done much academic writing before, you should read through the University's resources related to academic writing and related skills here.

Group Requirements

Your team must implement the following requirements. These are organised into **website requirements** and **API requirements**.

Website requirements

The following requirements form the core functionality of your team's blogging system:

User accounts

1. Users must be able to create new accounts. Each new user should be able to choose a username (which must be unique) and a password. At minimum, a user's real name and date of birth should also be recorded, along with a brief description about themselves.

2. When selecting a username while creating an account, users should be *immediately* informed if the given username is already taken. Users should *not* have to submit a form to discover whether their chosen username is taken so you will have to investigate how to use AJAX/Fetch for this.

3. When selecting a password while creating an account, users should be presented with *two* password textboxes (e.g. "Choose password", and "re-enter password"). They must type the same password in each box in order to proceed. If the user didn't enter the same password in both textboxes, they should not be allowed to submit the form. Ideally, a visual notification message, such as ("passwords do not match"), should also be displayed.

4. Users' passwords should not be stored in plaintext - they should be appropriately hashed and salted. You will need to research hashing and salting; there are NPM packages that can be used for hashing and salting passwords that you can research and implement.

5. When creating an account, users must be able to choose from amongst a set of predefined "avatar" icons to represent themselves; you can choose exactly where in the interface the user avatars appear, but you should make sure that users can see other users avatars in relevant parts of the page; e.g., you may wish to display a user's avatar and username beside comments they have made.

6. Once a user has created an account, they must be able to log in and log out; make sure to use an authentication system that allows users to stay logged and choose to log out when they wish to.

7. Users must be able to edit any of their account information (*including* their username), and also be able to delete their account. If a user deletes their account, all of their articles and comments (see below) should also be deleted.

Articles

8. Users must be able to browse a list of all articles, regardless of whether they are logged in or not. If logged in, they should additionally be able to browse a list of their own articles.

9. When viewing the lists of articles identified above, users should be able to sort article lists by *article title*, *username*, and *date* (but only one at a time). Aim to follow UI/UX conventions for user friendly sorting functionality. You may want to investigate how it has been implemented in similar interfaces. It is expected that the usability of your sorting options is intuitive and shows good interface design.

10. When logged in, users must be able to add new articles, and edit or delete existing articles which they have authored.

11. When logged in, users must be able to like articles. An individual user should only be able to like the same article once. Once a user has liked an article, they should be able to see that they have already liked that article. The total number of likes from all users should be displayed somewhere so users can see how many likes each article has. Users should also be able to remove their like from the article in case they like it accidentally.

12. When creating or editing articles, users should be presented with a WYSIWYG (what you see is what you get) editor. The WYSIWYG editor should allow users to edit the formatting of an article without having to edit the HTML markup. There are a variety of styles of WYSIWYG editors and you may code your own from scratch or integrate an existing WYSIWYG library; there are a variety available online but you should research the range of options available. Investigate WYSIWYG options carefully as it is better to do a more robust implementation of a simple editor that fits with the style of your site and how articles should display than integrating a WYSIWYG editor that will allow a user to create content that will break your site or display incorrectly. The editor should *(at minimum)* allow users to:
    - Add headings (or titles and subtitles)
    - Make text bold, italic and underline
    - Add bulleted and numbered lists

13. When creating new articles, users must be able to add an image to that article (if they choose - whether or not a user adds an image is up to them). When editing articles, users must be able to change, add, or remove this image. It is not compulsory to design it so users can add more than one image or add images inline with text. Implementing file uploads for multiple images can add complexity so it is suggested that you consider the implementation of image uploads carefully so that uploaded images of varying sizes will display appropriately; you may even want to consider having some form of resizing or validation to ensure images display well.

Comments

14. When logged in, users must be able to comment on articles. When viewing articles, comments associated with that article should also be viewable.

15. Comments must show the username of the commenter, and the timer & date the comment was made, in addition to the comment itself.

16. Users should be able to comment on comments up to at least two levels of nesting (i.e. comments on comments on comments). Any comment should be able to be replied to up to

two levels of nesting. Comments should be listed chronologically below the article or comment they are replying to. Comments that are replies to comments should be indented and directly below the comment they are in reply to.

An example of what two levels of nesting would look like is included below.

- comment…
- comment…
    - comment…
    - comment…
        - comment…
        - comment…
    - comment…

- comment…
    - comment…
    - comment…

17. Commenters should be able to delete their own comments; article authors should be able to delete any comments on their own articles. Make sure that you think through how you implement comment deletion and replies to deleted comments as there are a few different ways this can be implemented.
18. Users should be able to show or hide comments for articles they're reading.

Subscription and Social Network:

19. The website must have subscription functionality, which allows one user to receive the latest updates of another user after subscribing to that user. (Hereafter, we will address the user subscribing as "subscribers" and the user being subscribed to as "authors"). This feature should be supported and managed with relevant user interfaces that facilitate subscribe/unsubscribe actions of users (e.g., a subscribe/unsubscribe button). The user interface should include:

    - At an appropriate location(e.g., Under the author icon), a "subscribe" button/clickable should be available to allow one user to subscribe to an author.

    - Suppose the target author already has been subscribed by the user, in that case, an "unsubscribe" button/clickable should be available instead and facilitate the action of removing the subscription of the target author.

    - There should be an admin interface which displays a list of authors' profiles that the user is currently subscribing to and another list of subscribers' profiles that are subscribing the current user.

    - For each profile displayed in the list, the user should be able to navigate to their individual profile page by clicking the profile. In addition, there should also be an "unsubscribe/remove" button to unsubscribe/remove either subscription or subscriber through the admin interface.

20. Notification is an essential feature to nudge subscribers about new updates from the subscribed author. Please design a solution that will notify all subscribers when:

- The author creates a new article.
- The author makes/replies to a comment.
- A news subscriber starts following you.

21. There should be a notification badge displayed as a small icon in the navigation bar (or at the top of the page). with a special effect that indicates the number of unread notifications (see the Fig.2 example below).

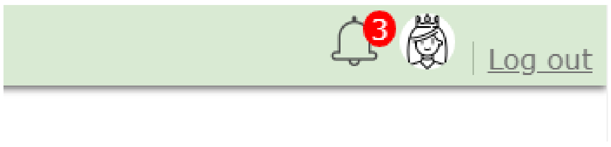
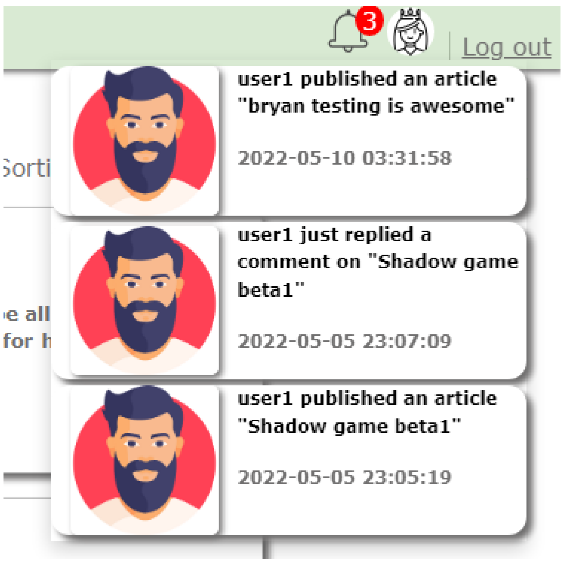


Fig.2 : Demo notification badge.

22. When the badge is clicked or when the mouse hovers over the badge, it should display a list of notifications, which are ordered by time and differentiate the read/unread status by colour or other visual aids. Relevant and succinct information should be included in each of the notifications (see the Fig3 example below), including:
    - the auth's icon
    - the author's username
    - a short message
    - a timestamp
    - if a notification is clicked, the user should be navigated to an appropriate page (e.g., the article mentioned in the notification message).



user1 published an article "bryan testing is awesome"

2022-05-10 03:31:58

user1 just replied a comment on "Shadow game beta1"

2022-05-05 23:07:09

user1 published an article "Shadow game beta1"

2022-05-05 23:05:19

Fig3: Demo notification message list display.

Analytics:

23. Analytics dashboards are often provided for authors to monitor the popularity of their articles and other metrics relating to engagement with their content. You are required to design and develop a simple dashboard page that will show some basic key metrics, the top three most popular articles, and a histogram chart:

    - Key summarised performance metrics:

        i. Total number of followers for the currently logged in user.

        ii. Total number of comments on the articles of the currently logged in user.

        iii. Total number of likes on articles belonging to the currently logged in user.

    - Display the top 3 most "popular" articles:

        i. Define your own formula for "popularity", which must involve at least two independent values e.g., "popularity = comments * 2 + total likes".

        ii. Rank all the articles of the currently logged in user based on their popularity, and only display the top 3.

        iii. For each article being displayed, show the rank, the article thumbnail image, the article title, the number of comments, the number of likes, the popularity, and the creation time.

    - A histogram chart showing number of daily total comments for each day over time

        i. Think carefully about the labelling on each axis and how the chart displays; you must ensure that the chart will display effectively when larger amounts of data are added. You may wish to add some maximum amount of days that are displayed if you feel that is appropriate. Your chart should be able to display up to at least 10 days of data clearly, but you may choose to allow it to display more than that.

        ii. It is suggested not to attempt to complete the chart by pure CSS. You should explore and consider some of the existing solutions. W3Schools can be a good starting point to look at how to make graphs/charts in web pages.

        iii. You should design your graph/chart carefully so that the labels on the axis and the range of data displayed make sense. You must add enough default data to at least one of the user accounts to showcase how the chart displays when there are at least 10 days of data. You should have the necessary SQL scripts included with your project to generate this default data along with the default user accounts so that other group members and markers can use the site with your default data.

Usability & Interface Design

24. The website must have a consistent look and feel, and must be responsive. The interface should respond well to the resizing of the screen and be usable throughout the range of common screen-widths. Consider how your group can have a structured and systematic approach to creating CSS.

25. The website must be user friendly and have good usability. When adding the features, consider how such features have been implemented in other websites you've used before. What did you like about those websites? What could use improvement?

    It is suggested you investigate usability and review resources such as Neilsen's heuristics and / or PACMAD.

26. It is suggested that you have a structured approach to interface design and consider creating 'wireframe' outlines of your pages so your group can discuss and plan what elements need to be on each page and how they are positioned. You may wish to create hand-drawn wireframe designs or investigate using a tool like Figma.

API requirements

The following requirements necessitate that your backend includes some routes (endpoints) that accept JSON and return JSON and / or various HTTP status codes and / or cookies. You will also need to research and understand the various request methods along with how to set the status codes of responses as not all of these were taught in the course. These routes should be created as part of your group project but will be accessed by the Java Swing-based administrative interface you'll be developing individually (see **Individual Requirements** below):

1. When a **POST** request is made to **/api/login**, a user should be authenticated using the username and password supplied *as JSON* in the request body

    a. If authentication is successful, a **204** response should be returned, along with information that can be used to identify the authenticated user in the future (presumably some kind of *authentication token*).

    b. If unsuccessful, instead a **401** response should be returned.

2. When a **GET** request is made to **/api/logout**, a user should be logged out (presumably by deleting their authentication token that was created above). Then, a **204** response should be returned.

3. When a **GET** request is made to **/api/users**

    a. If the requestor is authenticated **as an admin**, an array of all users should be returned, as JSON. For each user, all of their profile information should be included, along with the number of articles that user has authored.

    b. If the requestor is unauthenticated, or is authenticated but not as an admin, instead a **401** response should be returned.

    Hint: How will you determine if a particular user is an admin? Perhaps an extra column in the database?

4. When a **DELETE** request is made to **/api/users/:id** (where :id is a user id):

    a. If the requestor is authenticated as an admin, the user with the given id (if any), along with all of their articles and comments, should be deleted. Then, a **204** response should be returned.

    b. If the requestor is unauthenticated, or is authenticated but not as an admin, instead a **401** response should be returned (and nothing should be deleted).

    c. Note: there is something new in this step which you did **not** use in the labs. This step refers to using "path parameters" also known as "route parameters" for the id; "**/api/users/:id"** means that your route handler function will receive the parameter for the id in the path. For example, if you wish to get the user with the id '314', then the path for the request from the client would be "./api/user/314". Note that this is different to using query parameters which is what you have used in the labs. It was covered briefly in the lecture slides but it is suggested that you read a bit more about it online; here are some suggested links:

        - https://www.tutorialspoint.com/node-js-reading-path-parameters-from-url
        - https://www.geeksforgeeks.org/reading-path-parameters-in-node-js/

Individual Requirements

You must **individually** implement the following requirements, **after your team has completed the group requirements**. These requirements form a basic administrative user interface, built using Java Swing. The requirements listed here involve interaction with your group project's backend. To do this, make appropriate HTTP requests to your API endpoints, as created in the **API requirements** section above.

1. A Java Swing application must be created, consisting of a single window (i.e. JFrame).
2. The UI must include two text boxes - one each for allowing a user to enter their username and password.
3. The UI must include two buttons - one for logging in, and the other for logging out.
4. The UI must include a JTable to display user data.
5. The UI must include one further button, for deleting the currently selected user.
6. When a user clicks the login button, they should be authenticated using the username / password they entered.
7. If the user is authenticated as an admin above, then a list of users should be obtained from the backend and displayed in the JTable. If the user was not authenticated, or was authenticated but not as an admin, instead an appropriate error message should be displayed in a dialog box, and the user should be logged out again.
8. When the user clicks the logout button, the JTable should be cleared and the user should be logged out.
9. While authenticated, when the user selects a row in the JTable then clicks the "delete user" button, the selected user should be deleted from the backend (and the associated row should be removed from the JTable).
10. All buttons and other UI elements should only be enabled if it is currently possible to carry out their functionality. For example, the login button shouldn't be enabled if the user is already logged in, and the "delete user" button shouldn't be enabled if there is no row selected in the JTable.
11. Your program must use appropriate design patterns (e.g. observer / adapter / MVC for displaying items in the JTable) and apply the concepts taught in CS718.
12. The individual implementation is designed to provide an opportunity to apply the technologies and concepts taught in CS718. It is expected that you will use the general approaches and principles taught in CS718 to approach the implementation. You should not use any premade templates or GUI builders. If you have doubts about what tools are appropriate to use, ask one of your lecturers.

Notes

- The individual project will require you to investigate how to send HTTP requests, and to convert between JSON and standard Java objects, from within the Java language. A fully functional sample application will be provided to you, and a brief introduction will be given after your group implementation is complete, before you start the individual implementation. However, this will largely be a research task on your behalf.

- If your group is unable to complete the **API requirements** section above during the group implementation, you should inform us as we will need to make arrangements to ensure you are able to build the Java interface on top of a working NodeJS server with the required routes / endpoints.

Project Management

There are several project management considerations to keep in mind while working on this project.

Teamwork

For the group portion of this project, you will be expected to work with members of your team to effectively come up with a list of requirements, prioritize that list, and divide the work up amongst team members. Make sure to maintain good communication with your group and lecturers if anything affects your attendance and/or ability to contribute to the project. While working on the project, also keep in mind that each team member will need to demonstrate that they have worked on both the front-end (HTML / CSS / JavaScript) and back-end (node.js / SQL), and that they have contributed equally and fairly to the team; teams should support group members to work on a variety of parts. Make sure to outline your contributions to the frontend and backend in the final report.

Those considerations notwithstanding, how you divide the work amongst your teammates is entirely up to you, but **your scrum master (i.e. the lecturer) must agree on your team's proposed work breakdown.** Your team will lose marks if there is evidence of the group having poor teamwork or unfair task allocation; if there are issues with individuals this will be addressed on an individual basis.

It is important to communicate clearly throughout the development process. Do not be afraid to discuss ideas with your group if you are not sure what tasks you can be helping with. It is important that you communicate clearly to your group if you are unsure of things like who is doing what or how to approach a task you have been assigned. It can be helpful for groups to plan and assign tasks quite clearly so every group member understands who is editing different files at different times.

It is very important that you use the technologies taught in the course as the basis for your group's technology stack. All group members should have a fair chance to participate and apply the skills they have learnt in the course. If you have prior knowledge of other technologies, remember that other members of your group may not have this knowledge and it will make it difficult to participate in the project.

Pair Programming & Collaboration

We encourage pair programming and collaboration where possible although this will not always be possible with remote working. Consider how you may be able to work via' audio or video calls and screen sharing so you can review each other's code, help discuss problems and research next steps.

Communication

Remember to organise some easy way in which your group can communicate – especially if you're not working in the same location (though working in the provided lab space for the majority of the time is strongly encouraged). Facebook works well for this, but if you don't want the distraction, we recommend trying Slack. Or, feel free to use any other tool your group agrees upon.

Code Sharing

It is vital to properly utilise version control to reduce problems when working on the same codebase. The first thing your team should do is familiarise yourself with your GitHub repository, which you will all use collaboratively. You should work out who will be responsible for approving pull requests and communicating to other people in the group to update their master branches. Remember only one person can merge changes to the shared master at a time. We recommend the use of the Forking Workflow, as demonstrated in the git session; see details in the Canvas / Moodle module. Remember to work on small features and merge regularly as resolving merge conflicts can take up a lot of time if you do not do it regularly.

IMPORTANT: Your project should be set up with an appropriate '.gitignore' file. However, you should still look carefully at what files are being committed and merged and check that all members of the group have the same project setup.

Demonstrating Progress

You will be required to keep an up-to-date visualisation of your team's progress at all times. To do this, your team should set up a Trello board. This free online tool will allow your team to easily keep track of tasks, add them to different lists such as "To-Do", "Doing", "QA", and "Done", assign tasks to team members, and colour-code the tasks for ease of viewing. You will also add your scrum master / lecturer to your board so they may keep track of your progress easily.

During the regularly-scheduled class times, you will be required to make a short progress report to your Scrum master. This will involve a stand-up session where your team will briefly report on what you've done, what you're currently doing, and what you still need to do later.

Note: Keeping track of progress this way should not just be considered "busy-work" – it serves a real purpose, in that your team will be able to set clear goals and keep them in sight at all times!

Deliverables & Submission Instructions

Daily stand-ups

During each progress stand-up, the only thing you must specifically bring with you is the Trello board mentioned above, along with *all* of your team members at *every meeting*. If you have anything ready to demonstrate to the lecturer, feel free to do that also.

Group project source code

Your team git repositories serve as the submission for your project source code. Make sure that your team repo master branch is up-to-date on or before the due date above and make clear in your final commit message that it is the final commit (see **key dates**). Any commits after this deadline will be ignored by the markers.

In addition to your source code, your repo should also include the following:

1. An SQL script with your CREATE TABLE statements and any initialization data (db-init.sql);
2. A `README.md` file containing the following information:
    a. Team name
    b. Are there any special setup instructions, beyond initializing the database and running your project?
    c. At least one username / password combination for an existing user in your system with some already-published articles & comments
    d. Any other instructions / comments you wish to make to your markers

Presentation day

Your team must come to the presentation day with a functional version of your blogging system. Your team will be giving their presentation to the class.

Your team will have fifteen minutes to present, followed by a five-minute Q&A session. Each team member should speak during the presentation, as well as answer questions during Q&A.

Individual source code

Your individual implementation should be compressed into a single Zip archive, and the Zip should be uploaded to Canvas / Moodle on or before the due date.

Individual reports

Your individual reports should be submitted separately to Canvas / Moodle as a single PDF document, on or before the due date.