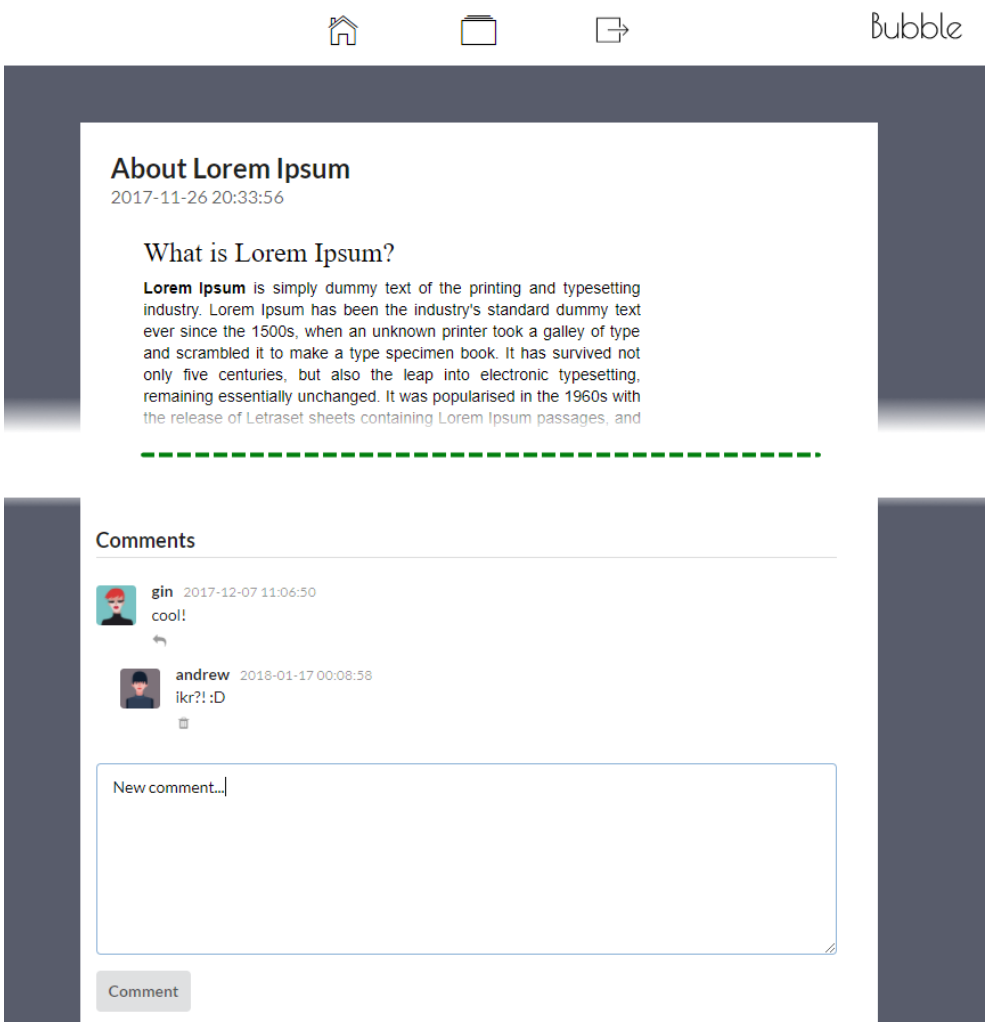
PGCert in Information Technology

Final Project - A Personal Blogging System



Credit: "Bubble Blog" (Emma Zhao, Tongxin Xie, Qian Peng, Xingjian Che)

Important Dates

- **Friday the 29th of January:** Project Introduction
- **Friday 12th of February:** Source code due
- **Friday 12th of February:** Reports due
- **Monday 15th of February:** Presentation day

Introduction

In this project you will develop a blogging website using the skills 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.

You are given a list of compulsory requirements for the blogging system, which must be implemented. You will also have the opportunity to show your creativity and / or technical skills by adding extra feature(s).

This project is a team project - each of you will be working in groups which will be assigned by your lecturer. Make sure to name your team something memorable!

Overall Breakdown

The final project is worth 30% of your total grade. It is broken down into 2 parts. The team deliverables is worth 20% and the individual reflective report is worth 10%

The two parts of the final assessment

- Group project team deliverables
- Group project individual reflective report

Group project team deliverables **(worth 20% overall)**

The team portion of this project consists of **100 marks total**, and is worth **20%** 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 are comprised as follows:

Deliverable	**Marks**
Implementation of compulsory features	*50 marks*
Implementation of research features	*30 marks*
Code style	*10 marks*
Daily stand-ups / project management	*10 marks*

Details of the individual deliverables are given in these subsections.

Implementation of compulsory features **(50 marks)**

There are a set of compulsory features which comprise the basic functionality for a blogging system. These are detailed in the "compulsory features" section below. Your team **must** implement **all** of these features adequately to receive full marks for this section.

Implementation of research features **(30 marks)**

Your group must implement the extra features listed in the research features section. These are a list of features that will involve a little bit of extra research to implement. Make sure to discuss these in your group early so you know how you will be researching each feature. You may wish to test some of the more complex parts in a smaller standalone project before integrating them.

Code style **(10 marks)**

Your 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, adherence to applicable patterns such as DAO and Web-MVC, amongst other considerations. It should be written in a way that would make it easy for other people to understand and modify.

Daily stand-ups **(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. Further details are available in the *Project Management* section below. The meetings will be short - approximately five minutes per team per meeting. Evidence of good teamwork is required at these meetings - your scrum master 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.

Presentation & demonstration (5 marks)

On **Monday the 15th of February**, 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 should be a fun session.

Each team will be allowed 20 minutes to speak, followed by a 5-minute Q&A session. Presentations should be professional, be slideshow-based (i.e. attempting to do a live demo is *not* recommended), 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.

Following the presentations, all attendees will get the opportunity to try out each team's project that will be deployed on trex-sandwich.com .

Individual Reflective Report **(worth 10% overall)**

In addition to the team component, you should submit an **individual** reflective report (each team member must write their own), which comprises **10%** of your final grade for the *Web* paper in the certificate. 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.
- Detail your particular contributions to the team and project; you are expected to have worked on front-end and back-end so outline what you did on both
- 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.

Compulsory Features (50 marks)

Your team must implement the following compulsory features, which comprise the basic functionality of a blogging system:

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.
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; we have some materials and files that will help with this.
5. When creating an account, users must be able to choose from amongst a set of predefined "avatar" icons to represent themselves. Users must also be able to upload an image from their computer that can be used as a custom avatar.
6. Once a user has created an account, they must be able to log in and log out.
7. 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.
8. When logged in, users must be able to add new articles, and edit or delete existing articles which they have authored.

9. When logged in, users must be able to comment on articles. When viewing articles, comments associated with that article should also be viewable (along with the username of the commenter and the date & time of the comment).
10. Commenters should be able to delete their own comments; article authors should be able to delete any comments on their own articles.
11. 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 should also be deleted.
12. The website must have a consistent look and feel, and must be responsive.
13. The website should be deployed to the trex-sandwich server before you submit the final code

Using libraries for CSS and JavaScript like Bootstrap, jQuery, w3.css or similar is ok, but do not use any commercial or freely available 'website templates' (e.g. Wordpress). If in doubt, *check with your lecturer first*.

Administrative interface

An administrative interface should be implemented using Swing that allows an administrative user to:

1. Add and remove users
2. Reset user passwords, by sending the corresponding user an email with a link to reset their password
3. Show and hide comments and articles
4. Search articles based on timeframes.
5. It should require username and password to connect and give the option to remember in future. [HINT: Preferences]
6. Import/Export list of articles and comments into a JSON file. [Prefer Jackson]

Research Features (30 marks)

In addition to the compulsory features above, your group should implement these more in-depth features. We've labelled these as research features as they are features that you should spend some time researching how to implement.

- **Advanced, nested comments**: users should be able to comment on comments up to at least 2 levels of nesting; i.e. comments on comments on comments. Any comment should be able to be replied to up to 2 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 2 levels of nesting would look like is included below.

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

- comment…
    - comment…
    - comment…

Comments should also have information about the time, date and author. Users should be able to show or hide comments.

Users who are logged in must be able to upvote or downvote individual comments. A user should only be able to either upvote or downvote an individual comment once. Users who are not logged in cannot upvote or downvote comments.

- **WYSIWYG editor**: add a WYSIWYG (what you see is what you get) editor that allows users to edit the articles they add. 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. The editor should allow users to add titles, subtitles, make text bold, italic and underline. The WYSIWYG editor should also allow users to add images inline; images added in the WYSIWYG should be uploaded as a fileupload properly images should not be stored as a 'blob' in the database. You should research which WYSIWYG editor carefully to make sure that your implementation can support file uploads.

- **Search and sort**: users should be able to **search** and **sort** article lists by *article title*, *username*, and *date*. Users should be able to use **search** and **sort** functions independently or jointly. For example, after searching for a list of articles for a given criteria, a user should be able to sort the results by title, date or author. A user should be able to search and sort articles without the browser window having to reload. Aim to follow conventions for user friendly searching and sorting functionality. You may want to investigate how it has been implemented in similar interfaces.

Project Management

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

Teamwork

With the exception of the reflective report, the rest of the project is a team 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 (NodeJS / Express / MariaDB), 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 front 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.

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 organize some easy way in which your group can communicate – especially as 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 Zoom, Discord or Slack. Or, feel free to use any other tool your group agrees upon.

Code Sharing

It is vital to properly utilize version control to reduce problems when working on the same codebase. The first thing your team should do is create a GitLab repository, which you'll all use collaboratively. We recommend the use of the Forking Workflow, as demonstrated in the "advanced git / mini project" lecture. A video from the University of Waikato is available here, which explains a little more about how to use git collaboratively in an effective manner. It is recommended that you investigate how to create a group in GitLab to house your shared

repository. 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: You must also setup your initial repository with an appropriate '.gitignore' file; you can use one of the '.gitignore' files from one of the lab projects or research how to write your own. 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 visualization 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 color-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.

Code submission

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. 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:

3. Team name
4. Website URL on the course server (trex-sandwich)
5. What are your extra features?
6. Are there any special setup instructions, beyond initializing the database and running your servlet project?
7. At least one username / password combination for an existing user in your system with some already-published articles & comments
8. Any other instructions / comments you wish to make to your markers

Individual reports **(12th of February @ 11.59pm)**

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

Presentation day (15th of February @ 1pm)

Your team must come to the presentation day with a functional version of your blogging system having already been deployed to the course server (*trex-sandwich*). Remember to deploy early and often, to avoid any last-minute issues.

Your team will be giving their presentation to the class.

Your team will have twenty 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.

Following the presentations, there will be an opportunity for attendees to trial all blogging systems through sharing the URLs of the deployed versions of the site.