Subject Number:	SWEN30006
Subject Name:	Software Modelling and Design
Submission Title & Part no:	Submission 3
Submitting for group:	Erwin Fernandez (567034) Erlangga Satria Gama (570748) Marisa A. Tjoe (566322)
Due Date:	4/05/2014

For Late Assignments Only

Has an extension been granted? Yes / No (circle)
A per-day late penalty may apply if you submit this assignment after the due date/extension. Please check with your Department/coordinator for further information.

Plagiarism

Plagiarism is the act of representing as one's own original work the creative works of another, without appropriate acknowledgment of the author or source.

Collusion

Collusion is the presentation by a student of an assignment, as his or her own which is in fact the result in whole or in part of unauthorized collaboration with another person or persons. Collusion involves the cooperation of two or more students in plagiarism or other forms of academic misconduct. Both collusion and plagiarism can occur in group work. For examples of plagiarism, collusion and academic misconduct in group work please see the University's policy on Academic Honesty and Plagiarism: http://academichonesty.unimelb.edu.au/

Plagiarism and collusion constitute cheating. Disciplinary action will be taken against students who engage in plagiarism and collusion as outlined in University policy. Proven involvement in plagiarism or collusion may be recorded on my academic file in accordance with Statute 13.1.18.

STUDENT/GROUP DECLARATION

Please sign below to indicate that you understand the following statements:

We declare that:

- This assignment is my/our own original work, except where appropriate citations mention the original source.
- This assignment has not previously been submitted for assessment in this or any other subject.

For the purposes of assessment, I/we give the assessor of this assignment the permission to:

- Reproduce this assignment and provide a copy to another member of staff; and
- Take steps to authenticate the assignment, including communicating a copy of this assignment to a checking service (which may retain a copy of the assignment on its database for future plagiarism checking).
- Indicate each group member's % collaboration to this project below.

Student 1: Erlangga Satria Gama % contribution 33.3%

Student 2: Erwin Fernandez% contribution 33.3%

Student 3: Marisa Tjoe% contribution 33.3%

Summary of Changes

There are a few changes that we made in the implementation phase in contrast to the initial design that we planned out. Some of these changes are meant to simplify several use-cases, and others to improve.

The most significant area of change is the model. When we drew out the UML diagram in submission 2, we were not aware of the relationships between models, and how to implement them in mySQL. Majority of models were connected by a many-to-many relationship, we realized later on that implementing a many-to-many relationship requires a mediator table. For instance, the Society model and the Event Model has a many-to-many relationship, hence we made another model called SocietyEvent that contains a society id and a model id to connect both models. Our UML diagram also lack a significant amount of Models, such as Network, University, and others. This is because we initially thought they are rather insignificant to be implemented as a model, we realized later on that these secondary models are essential for the system. The controllers in the UML we made also lack a significant amount of essential functions.

We also simplified several use-cases. One of which was the user/society/event registration system, where we actually do several changes. However, our system implementation were actually pretty much accurate to the the sequence diagram that we made in Submission 2.

Limited changes was made to the initial mockup that we made in Submission 1. There are several layouts that we changed in the society and event registration system. This changes was made based on the availability of APIs/plugins to better fulfill the function. For example, the method of adding admins to a society in it's registration page was rather difficult to implement elegantly. We couldn't find any web UI plugins that would fulfill our initial tabular design, so we ended up using a single text area field and add several open-source web in order to have a tag system for admins. This is similar to what we find in many popular website like Wordpress or GMail. Another change we made to the UI was ousting the asynchronous suggestion drop-down in the search bar. We realize that this is really hard to implement with our lack of understanding and experience of JQuery and AJAX. In additional, we finally decided to populate the category from the database that we have provided. At first, we thought the tag system (similar to Twitter's hashtags, but this is to identify society/event) will be great, however with the consideration to reduce the randomness of category we finally use drop down menu so that the use can choose the standard category.

Reflection and Critique

REFLECTION

The submissions of this project follows a top-down approach. In Submission 1, we made high-level designs, consisting for several mock-ups of our website, and use-cases of several functionalities. We also conducted surveys, aimed for both society admins and society members. In Submission 2, we drew out a UML class diagram that implements the system in MVC, and also visualize the user-system interaction by converting our use-cases to sequence diagrams. In contrast to the previous two submissions, submission 3, being the implementation phase, requires more technicality and less planning.

For most of us, this staged implementation of a project is a familiar but newly implemented concept. The norm was to implement straight away, with limited design decisions made beforehand, and change the design regularly based on the problems we face midway. After carrying out this project using a staged implementation, we felt that it is more effective and efficient, since the design and implementation are done separately, and hence we are able to better focus on a single task. This is especially true in the implementation phase, where the design is available on hand, and there are less design decisions or changes made midway, so we can focus on the technical difficulties we face.

The use of MVC leads to better abstraction and makes the project easier to plan and implement. Rails provide a very easy way for us to separate these different components and focus on them individually during implementation. It is also relatively easy to learn due to the reliable online documentation and large user base in online forums. All in all, it was a great experience using an MVC web framework to implement a website.

Given only three weeks to finish the implementation with limited knowledge of Ruby and the Rails framework, it is rather difficult to create a mature and feature-loaded system. There are several additional functionalities that we previously hoped to add, but had to let go due to time constraints. Therefore, there are a lot of limitations in this system. Despite that, we tried our best to provide a good UI that provides easy navigation for users and is nice to look at.

CRITIQUE ON PROCESS

When drawing out the UML class diagram, the Rails framework and even the MVC architecture was very vague to us, and hence we were not able to clearly divide the classes into Model-View-Controller. We should have tried implementing a simple project using Ruby on Rails before making the UML diagram. That would allow us to get familiar with the different classes in the framework, and the division of Model-View-Controller.

There should be more introduction in software architecture as well. It is difficult to look at the power of Model-View-Controller architecture since we lack of knowledge about the other features. Another problem would be the lack of clear textual distinction between Model-View-Controller and Model-View-Presenter. Since we found many suggestions from online and trusted offline sources that Rails is MVP instead of MVC since MVP can be a type of MVC. Indeed, we believe that the clear understanding of this architecture will lead to better result in our works. Another problem we faced is the lack of consistency in the explanation of several architectures and other important concepts. This is very confusing for us.

Since there is no introduction for the security system, what we did on this project cannot be fully implemented in real work. We need to know the power of Ruby on Rails in tackling the security issues that is a lot happening in the real world.

Several good features that is using client-side programming also cannot be done optimally due to the lack of client-side programming introduction. Many features in web development are using client-side programming like JavaScript or JQuery. However, we could not implement these features optimally since we have not much knowledge about how to implement it in terms of coding in our project.

CRITIQUE ON IMPLEMENTATION SYSTEM

One of the main limitation of our system is the lack of security system features, of which we were unfamiliar with. We realize that there are actually too many “holes” in this website that we are not be able to handle. For example, we do not have any filter system for our file and image uploader, there is no guarantee that the client will not be able to upload any kind of file that contains virus. Also, we lack of human verification, in the other hand is the use of captcha to make sure that the registration was done by human. However, other way we could improve on that would be to add a verification system for every user/society registration. For user registration, there could be a mailer that sends verification emails, containing the url to the verification page, like in other websites. For society registration, there is no measure whatsoever to verify that the society is an official society of their respective universities. This problem is a bit trickier to handle, as waiting for replies from universities for every society registration might be inefficient timewise, and getting an official database of societies that requires updating every time a new society registers might require agreement from Universities. Hence, the current system automatically registers the society without any form of verification. This problem will be better handled if

the system goes into real development and will be deployed for use. Rails also provide basic security measures, especially in forms, we tried to make use as much of that as possible.

We realized that the website is lacking several important or useful functionalities. One of the functionality that we wished to have is an integration to social media, for instance allowing users to sign up through their Social Media accounts without having to fill in other forms. Another feature that is possible by doing this is that relate the user interest-for example, from Facebook-with the club's categories so that it is much easier for user to find the suitable clubs. These feature would be easy to do with the APIs provided by the different social medias, but would require further research.

Another functionality that we planned to have and bailed in the implementation phase was the auto-complete search form. However, it requires a deep understanding of JQuery. This is challenging for us that is not experience yet with JQuery.

One of the important issue that we did not pay attention at is using cookies and session to store the users session in their browser. With the lack of experience in web development, we admit that we did not maximize the power of these two features in web development. However, several Ruby Gems that we used might use these features, such as Devise.

Overall, we found that having a staged planning and implementation to be better and more organized than that without any planning. Moreover, implementing the system using the Ruby on Rails framework has been a great learning experience for us, despite the confusion we experienced regarding the different architectures.