ELEC3609 Web Applications Group Development Project

As part of ELEC3609 you will form a group of 4 students and you will go through the software development cycle to produce a series of web applications.

The project is worth 40% of the course assessment.

The project is made up of 4 deliverables:

Deliverables:	
Deliverable 1 (due week 5 in tutorial and worth 10%):	Response to Request for Proposal with Requirements Analysis and Specification
Deliverable 2 (due week 7 and worth 5%):	Design Specifications
Deliverable 3 (due week 12 and worth 20%):	Implementation of Web Applications (Working Prototype)
Deliverable 4 (due week 13 and worth 5%):	Test results and user documentation

Each student from each group will have to sign the front page of the deliverables and state their contributions (which tasks they have done) to be properly graded.

Marks may differentiate for each member of a group if a certain student does not contribute equally to the group's deliverables.

You must be part of a group by the end of week 2. You must notify the lecturer Frank by email frank.moisiadis@sydney.edu.au or at frank@moisiadis.com and the tutor in the tutorial of the members of your group and their contact details (email and student no) by the end of week 2.

Each member of each group needs to provide a short description of their software development skills which they have with the first deliverable of their group.

Deliverable 1 (worth 10%): Due in week 5.

Response to Request for Proposal with Requirements Analysis and Specification

Request for Proposal:

You are asked to propose a solution to the following business problem:

ELECsociety is a new idea to develop a web portal (community site) for all ELEC students with some of the following web services:

- Hosting student websites,
- Providing a public forum or discussion board to help fellow students with academic issues.
- Providing a store for students to buy and sell items to and from each other
- Providing a facility to pay for items bought at the ELECsociety site
- Providing an award point system for people who use the site
- Presenting a newspaper for the society
- Placing advertisement banners for the site

Other ideas might be to provide banking facilities and foreign exchange facilities and to provide a research database for honours projects.

It should be a community site and there should be at least four sub-sites: For example, forums, bank accounts, a content management system (a newspaper), an advertisement (e.g. banners) module and a simple ecommerce module (e.g. to spend the points).

Consider the use of private and public web services with interfaces and databases for the portal.

You can implement the project using .Net, Java and XML.

Your group must decide what they can do and present it as a proposal for the ELEC society portal by week 5.

Deliverable 1 has two parts:

1. **Group Proposal**

A proposal for the software problem to be tackled by the group and why it is a good software project to develop (remember you only have 12 weeks so do not stretch yourself with a big project, you may even state which part of a large project you will focus on). Describe the scope of the web applications you are going to deal with and specify what part of the applications you will focus on. Include the programming languages you will choose to implement the web applications with (3 pages).

2. **Requirements Analysis and Specification document:**

A short Software Requirements Specification using template presented in lectures (10 pages).

Use Case Diagram (1 page).

Use Case Description for the three most important use cases (5 pages).

Analysis Class Diagram (1 page).

Sequence Diagram for the most important use cases (1 page).

State Diagram for one important object (1 page)

<u>Use Case Diagram:</u> This is a graphic model showing the actors, the use cases and the relationships between them. One page should be sufficient. As a rule of thumb – if the description of a use case is very short (e.g. one or two steps only) or very similar to another use case, then consider combining the use cases and describing the alternate courses of action within that use case. Structure the use cases into logical groupings. Remember it's from the users' point of view – not the developers.

<u>Use Case Description:</u> For each use case, there needs to be a corresponding textual description. Please use the template shown in lectures. Sub-use cases can be combined into one use case description. For example, if you have a use case Maintain Client with sub-use cases Adding Client, Deleting Client, Viewing Client or Modifying Client, you can just write one use case description

for Maintain Client and describe the differences by branches in the use case steps.

Analysis Class Diagram: At the analysis stage you may not have discovered methods, boundary or control classes. Make sure any classes or methods on your sequence diagrams have been included on the class diagram. Full method signatures do not need to be given, but you can specify them now if you chose. The diagram must include classes, attributes, and associations.

Sequence Diagram: One sequence diagram for one use case description. It should be possible to read the description and follow what is happening on the diagram. The objects and messages must be valid and shown on the Class Diagram.

State Diagram: A state diagram shows the life cycle of an object and thus all objects can be represented in a State Diagram. However, you are required to consider the life cycle of one object in your system and to submit diagrams for the states it may have. In most cases when an object is updated or printed (updated and printed can be states themselves but are generally not very meaningful) that will not change more interesting states such as paid/unpaid, married/single or for sale/sold.

Apart from signing the deliverable (or acknowledging their work) ALL team members MUST clearly specify their contribution to the deliverable (in a table or other format that is clear to the marker).

ALL DELVERABLES ARE EMAILED TO FRANK AND SUBMITTED ONLINE TO GRADE.

frank.moisiadis@sydney.edu.au

frank@moisiadis.com

DELIVERABLE 1 IS DUE IN WEEK 5 (SUNDAY MIDNIGHT).

Deliverable 2 (worth 5%): Due in week 7.

Deliverable 2 (due week 7 and is worth 5%): Design Specifications.

This design document must contain the following:

1. **System Design Document**
2. **User interface layouts**
3. **Program Navigation Diagram**
4. **Model View Controller diagram**
5. **Data definitions**
6. **Design Class Diagram**
7. **State Diagram for any important objects**
8. **List of design assumptions (if any)**

System Design Document: This will include the basic architecture of the system and the high level strategy decisions. You need to include a description of the:

a) system architecture

b) storage/persistent data strategy

c) trade-offs and choices

d) any concurrent processes and how they will be handled if any exist

e) Package diagram showing the subsystems you will use

User Interface Layouts: Use a drawing package or system builder like eg. Jbuilder or Visual Basic to show your proposed interface layouts.

Program Navigation Diagram: Show how the different screens will link together and what triggers a transition from one screen to another. Activity diagrams are recommended for this purpose.

Data Definitions: Create a table showing what data is needed to be stored in files/database. For each table/file show the name of the field, the primary key (if applicable), the field type and an example of data in this field.

Design Class Diagram: you should have discovered methods, boundary or control classes. Make sure any classes or methods on your sequence diagrams have been included on the class diagram. Method signatures should be given. The diagram must include:

classes, attributes, associations, inheritance and/or aggregation (if applicable), traversals, multiplicities.

State Diagram: A state diagram shows the life cycle of an object and thus all important objects can be represented in a State Diagram. However, you are required to consider the life cycle of each object in your system and to submit diagrams for those that have interesting states or complex behaviour. One way to measure if a state is interesting is to consider whether you need to test that state before performing a particular action or if the state changes after an action is performed. What is interesting will depend on the application. In most cases when an object is updated or printed (updated and printed can be states themselves but are generally not very meaningful) that will not change more interesting states such as paid/unpaid, married/single or for sale/sold.

Model View Controller diagram: refer to slides in week 4 lecture material

List of Assumptions: This will help the marker/review team to understand why you have done certain things. Please review the assumptions as a group before submission. A poor assumption will not be a valid reason for poor design decisions.

Apart from signing the deliverable (or acknowledging their work) ALL team members MUST clearly specify their contribution to the deliverable (in a table or other format that is clear to the tutor is marking the work

ALL DELVERABLES ARE EMAILED TO FRANK AND SUBMITTED ONLINE TO GRADE.

frank.moisiadis@sydney.edu.au

frank@moisiadis.com

DELIVERABLE 1 IS DUE IN WEEK 7 (SUNDAY MIDNIGHT).

Deliverable 3 (worth 20%): Due in week 12 in your tutorial class.

Deliverable 3 (worth 20% and due week 12 during your tutorial class):

Software Demonstration

Each group will be given 15-20 minutes in week 12 to demonstrate their web applications to the class and tutor.

A working prototype should be presented.

All member should be ready to answer any implementation questions.

The general marking scheme for deliverable 3 will be as follows:

1. Correctness (50%, 10 Marks) - whether deliverable 3 works and functions according what was presented in deliverable 1 and 2.

2. Coherence (10%, 2 Marks) - whether deliverable 3 is coherent with what was presented in deliverable 1 and 2.

3. Completeness (10%, 2 Marks) - whether deliverable 3 is an accurate reflection of what was presented in deliverable 1 and 2.

4. Complexity (15%, 3 Marks) - the standard to which deliverable 3 demonstrates technological complexity and the application of the technology taught in lectures.

5. Creativity (15% 3 Marks) - the standard to which deliverable 3 demonstrates creativity in delivering the solution to the problem proposed.

Deliverable 4 (worth 5%): Due in week 13 in your tutorial class.

Deliverable 4 (worth 5% and due week 13 during your tutorial class): Test results and user documentation.

Test Specifications: This document must not exceed ten (10) A4 sides and should contain the following:

1. Test-case specifications, made up of test-case identifiers, test data - input specifications and output specifications.

2. Test plans, including a detailed test schedule, testing resources assigned, testing milestones and test deliverables.

Acceptable documentation for this deliverable should include:

1. Test case specifications:

a) Identifier.

b) Test description.

c) Input specifications.

d) Output specifications.

2. Test plans, covering scheduling and resourcing of all testing processes.

User Documentation

The User Documentation should not exceed ten (10) A4 sides, and should enable a moderately computer-literate user, initially completely unfamiliar with the system, to understand and fully utilise its functionality

Apart from signing the deliverable ALL team members MUST clearly specify their contribution to the deliverable (in a table or other format that is clear to the tutor is marking the work).