CITS4401 Software Requirements and Design

Software Design Assignment

Worth 30% of total assessment

Due: Friday, May 6th, 2016 @4pm

Aim: The aim of this assignment is to test your ability to design a medium sized software system and to demonstrate your ability to manage and present clearly a design document.

This assignment is designed for students working in pairs. You are allowed to work alone but improved group work is an aim of this unit and so is encouraged. Both partners are required to contribute equal effort to the assignment. See the Unit Coordinator as soon as possible (well before the deadline) if you believe this is not the case. Please feel free to post to **help4401** (the discussion forum) if you need to line up a partner.

If you choose to work on your own, the workload will be very high. The same marking criteria will be used whether the assignment is done collaboratively or individually.

Task: Your task for this assignment is to develop a system design for a computer system to support the membership management for a community organization with a central headquarters but a dynamic range of local branches. The system will be called the **Dynamic Branch Membership System** (**DBMS**). The requirements definition for this system is given in the DBMS Requirements Definition reproduced below. As well as the analysis for the software design, you should document the rationale for the design decisions you make and also keep a **log** (listing date, task, people involved, time taken) for all work undertaken on this project.

Registration: Whether working in a group or by yourself, you must register on the following website:

http://teaching.csse.uwa.edu.au/units/CITS4401/OGRES/public_html/index.php

Please note down the group number generated by OGRES. On the first page of your design document, you should clearly show your group number, student number(s), family and given names of member(s) in the group.

Project Planning: You are expected to spend 40 to 50 hours on this assignment, including background reading. The work you submit for this assignment must be your own (either the designated pair or the individual). In accordance with the School of CSSE Plagiarism policy "Any contribution from others … *must* be acknowledged as part of the submitted work. Students must inform the Unit Coordinator if their work is done jointly or borrows heavily from others. Failure to do so is **plagiarism**." This includes the use of templates, textbook examples and all information from the web. For the full policy see

http://www.governance.uwa.edu.au/procedures/policies/policies-and-procedures?policy=UP07%2F21

Submission: Please ensure that you keep a backup copy of your system design document for yourself before submission. If your diagrams are hand-drawn, then you can scan all the pages of your assignment into a PDF file using any photocopier in a UWA library. Please ensure that your diagrams and handwriting are neat and legible. You can also investigate for a software tool that supports the drawing of UML diagrams. Your diagrams and descriptions can be put together as a Word document; however, you must convert the document into **PDF** for submission.

Your document should be in PDF and should be submitted to the *cssubmit* website:
https://secure.csse.uwa.edu.au/run/cssubmit
If you work in pairs, only one student should log onto *cssubmit* to do the submission. Please discuss between yourselves who should handle the submission and ensure that the right version of the document is submitted. All late submissions will be penalized in accordance with the School of CSSE policy on late submission. For the full policy see
http://web.csse.uwa.edu.au/students/assessments/late-submissions

Your system design document (80%): Your design document for the DBMS should comprise the parts described below. For parts a to g, you should supply rationale for your design decisions; however, you do not need to structure your arguments in rhetorical steps (such as *issues*, *proposals*, *criteria*, …, as given in lecture 13). There is no minimum or maximum page limit on your design document. You should target at covering all the parts below and adopt a common document format (font: Times, size: 12, margins: 2.5-3cm).

a. **Functional modelling.** Provide a use case diagram containing appropriate choice of actors and use cases which covers all the functionality of the given requirements definition but does not invent any new functionality. In the diagram, you should use the client's language and not new or ambiguous terms. Boundary between this system and external systems should be clearly identified and justified. For each use case in your diagram, you should provide a table that describes the use case (see the lecture note for the template).

b. **Object modelling.** Provide a class diagram to show the significant relationships between objects identified in the DBMS. Reasonable roles and multiplicities should be given in your diagram. For each class in your class diagram, supply the attributes (including the types) and major operations. Again, use the client's language and do not invent new or ambiguous terms. Names of attributes and operations should be explained. To avoid putting too much information into a single diagram, it is recommended that your main class diagram for the DBMS should contain only the names of all the classes. For each class in the main class diagram, you can then draw a separate class diagram showing the attributes and operations.

c. **Dynamic modelling part a.** Provide 5 sequence diagrams chosen from any 5 use cases in your use case diagram above. Identify in each sequence diagram the participating actors and objects. Explain the type (*entity*, *boundary*, or *control*) of each object. Clearly label the messages sent between the participating objects. Your sequence diagram should be consistent with your class diagrams.

d. Design constraints. List 5 design constraints for the DBMS and prioritize them. Give sufficient explanation for each design constraint and its priority.

e. Subsystem decomposition. Describe in detail how you would decompose the DBMS into subsystems. Describe the services provided by each subsystem. Justification of your subsystem decomposition (such as coupling and cohesion) should be given. Are there other ways to decompose the system? Include diagrams alongside your description.

f. Dynamic modelling part b. Supply 4 statechart diagrams to describe the dynamic behaviour of the overall DBMS and/or certain parts of the DBMS. Give a detailed description for each diagram. Choose meaningful names for the states and transitions. Again, use the client's terminology.

g. Design patterns. List 3 design patterns that would be appropriate to be used in the system. Describe which part of the system that each pattern is suitable and justify why you chose these patterns.

h. System design rationale. Now think about implementing the DBMS. Your aim is to develop a working system that is efficient, reliable, and secure. Although you would, hypothetically, be free to use any programming language (you are not required to write any code in this assignment), one constraint is that the DBMS must work over the internet. Before beginning any coding, you would need to consider about hardware and software components that would fit together and work with your design. In your system design document, identify 3 issues that may have two or more possible solutions and that require you to resolve for the best solution for each issue. Follow the structured steps (*issues, proposals, criteria, …*) for the design rationale as shown in lecture 13 to address these issues and your decision rationale.

i. Log. Attach at the end of your system design document a log, which should list the date, task, people involved, time taken, for all the work undertaken on this project. You can supply your log in a table form.

Seminar presentation (20%): You must give a seminar to explain your design for DBMS. If you work in a group, both group members must be present and participate in the talk. Each seminar will be approximately 15-20 minutes long (including question time). There is no need to create fancy Powerpoint slides. However, diagrams/text shown on the slides should be clear and legible. In the seminar, you should focus on the important aspects of your design, as there may not be sufficient time to go through all the fine details of your design.

More details about the seminar timetable will be given later.

DBMS: Dynamic Branch Membership System

(Requirement Definition Document)

1 System Objectives

The Dynamic Branch Membership System (DBMS) is a software application to assist in the task of managing the membership of a community organization which has a national HQ but a variable number of local branches.

2 System Context

1. DBMS will provide a central server program to assist with managing branch membership for a community organization "Wetland Watch" (WW). DBMS helps with the membership fee records, allocation of new members to branches, and the formation of new branches.
2. The system communicates with four separate existing systems: a commercially run credit card payment service; a commercially run online Australian address, postcode and distance service; (via a local network) the current national membership records database server for "Wetlands Watch"; an email server.
3. The existing membership database contains contact information, membership payment records and branch details. However, it is not accessible from the Internet.
4. Administration staff are able to set up DBMS, connect it to the other server programs and input the current branch details.
5. Each branch has a secretary (who is a member of WW) and each member of WW will be (eventually) allocated to a branch. There is also national secretary of WW who needs to be informed (via email) of certain membership decisions.
6. Branch secretaries, ordinary members, and prospective members will only access DBMS through a secure web facility.

3 Functional Requirements

3.1. New Member

1) A public web site allows interested members of the public to choose to become members of the WW.
2) The user selects the "become a member" option and goes through a process of entering personal information followed by a credit card payment process. There is only one standard amount to be charged which is annual membership for an individual.
3) Credit card membership payments are handled by a separate online service provided by a bank. DBMS informs the service of the amount to charge and then allows the user to interact with the service directly (and securely). If the amount is charged successfully then this service informs DBMS and allows DBMS to

continue interacting with the user.

4) Membership details are recorded in the membership database.
5) The new member is welcomed to WW and then informed that they are expected to sign up to belong to local branch. DBMS uses a separate commercially available geographic online service to display to the new member a list of the 10 local branches of WW that are closest in distance to the user's home address.
6) The user is asked to select one branch to belong to. They are informed (on the web page) that the branch secretary will contact them in the next few days. The member is also sent a confirmation email, which also gives them a temporary password to use to allow log onto the DBMS as a member.
7) An email is automatically sent to the selected branch secretary informing him/her of the existence of the new member. Contact details of the new member are sent to the secretary.
8) Reminder emails are sent every week to the branch secretary about the need to contact these "pending branch members" until the branch secretary logs onto DMBS to confirm that he/she has contacted and accepted the new member.
9) The new member is then sent an email confirming that they now belong to that branch. Alternatively, if the branch secretary reports that the branch is not happy to accept the new member then an email to that effect is sent. An email is also sent to the national secretary of WW.

3.2. Edit existing member

1) At any time an existing member of WW can log onto the DBMS (via a web page) to view or edit their contact details. Alternatively, an admin staff member can log onto edit the details of any member.
2) They can also search, browse and view alternative branches to which they could belong.
3) If desired, they can request to be transferred to another branch. In that case, their current branch allocation is cancelled. The member is added to the list of pending branch members for the other branch and (as above) will wait for contact and confirmation from the branch secretary.

3.3. Create new branch

1) A member can log onto DBMS to start a new branch. Alternatively, administrative staff can do this for them.
2) The member specifies the name and location of the branch. The member him/herself will be the new branch secretary.
3) An email is sent to the national secretary requesting approval of the formation of the new branch.
4) Approval (of the set up of the new branch) is indicated to DBMS by administrative staff.
5) If approved, the new branch secretary and members in nearby branches are

informed of the creation of the new branch by email (automatically generated by DBMS).

3.4. Edit branch

1) The branch secretary can log in at any time to edit details of the branch.
2) S/he can change branch location or change the branch secretary to be another member of the branch. Branch members and the national secretary are informed by email.
3) The branch secretary can view and then accept or deny any pending branch members. They will be informed by DMBS via email.
4) The branch can be closed down. Members are informed by email and asked to log in at their convenience to choose a new branch.

3.5. Renewal Reminders

1) DBMS will automatically send emails to members to remind them a few weeks before membership renewal is due.

4 Quality Requirements

4.1 Performance

4.1.1 DBMS shall show no visible deterioration in response time as the number of users of the system increases.
4.1.2 DBMS shall require a reasonably small amount of memory so that enough of it is permanently resident on the server to provide quick service.
4.1.3 DBMS shall load as quickly as comparable productivity tools on whatever environment it is running in.
4.1.4 DBMS will respond to client web activity in a timely and convenient way.

4.2 Reliability

4.2.1 DBMS shall be available for use as much as comparable productivity tools.

4.3 Usability

4.3.1 DBMS shall provide a standard style of user interface so that users do not have to learn a new style of interaction.
4.3.2 Users will be able to understand the layout and options of the DBMS user interface.
4.3.3 Notification and email messages generated by DBMS shall be clear, succinct, and polite and free of jargon.

4.4 Portability

4.4.1 DBMS will be implemented on a platform that allows easy re-hosting on different hardware and OS.

4.5 Modifiability

4.5.1 DBMS will be implemented using modern programming practices that maximize the maintainability and reusability of designs and code.
4.5.2 DBMS will be implemented in such a way that alternative authentication, geographical, payment, database and email server programs could be used easily without affecting the logic of the design.

5 Future Requirements

5.1 Facility for branches to use DBMS to coordinate their group meetings.
5.2 Support for SMS and other means of contact besides email.