Module CSC3054/7054

Web and Mobile App Development: Team based project

Submission date: completed software and project report should be submitted on or before (no later than) 16:00, 26th March, 2015

1. Aims

This team based project is assessed. This is an open-ended project that provides students with an opportunity to increase their knowledge and understanding, and to enhance the skills in mobile and web app development. This project also allows students to consolidate and practise the knowledge, understanding and application that they have gained in the classes. **Students are permitted to take any project as long as they can justify that the project to be taken is closely relevant to the contents of the course module, e.g. Android app or mobile web development**.

2. Learning outcomes

- Knowledge
    - A range of IT tools and software packages required for the project
- Skills
    - To work in a task oriented team
    - To use computers properly and develop appropriate software for the specified task
    - To apply suitable human computer interface techniques
    - To prepare and deliver documentations for the project/task
- Understanding
    - Analysis of a real-world problem
    - Development of a fully working software solution
    - Collecting materials and references for real-world problem solving

3. Key indicators

- A team of 5 or less students
- Use of object oriented programming languages, such as Java or C++
- Provision of a working software solution with appropriate documentations
- Peer assessment to your team members
    - To provide a score, for example, 20 out of 30
    - To explain in detail why this score is given
    - To report the disagreement if there is any

4. Suggested procedure of conducting a team based project

- To form a group or team of 5 or less people and then select a team leader
- To generate a project title and related project description
- To plan project activities and organise activities as appropriately
- To allocate tasks to individuals in the group

- To provide feedback to each other in the team
- To organise system integration for the system delivery
- To revise and improve the prototype
- To write up a report to explain how the project is handled, how the codes are implemented, and what is the outcome of the project (using images to illustrate, etc.)
- To submit the project software with its technical report
- To submit peer assessment results of each team member according to their contributions to the project design and implementation

5. Project submission

Please try to attend the allocated practical sessions over the coming weeks and use these time slots for the project implementation. Before submitting your work, please test the generated project on an Android platform (e.g. Android tablet or mobile phone) thoroughly and add appropriate comments to the programs. Please note that if you are using an Apple device, please make sure the developed application has also been tested on an Android device.

The leader of each team will be required to (1) submit the project title, project description and team members to the email address: h.zhou@qub.ac.uk before 16:00, 24th February, 2015, and (2) submit the entire project codes, a consolidated peer assessment report and a word document to explain what has been done in the programs in a zip file to the email address: h.zhou@qub.ac.uk before 16:00 on Tuesday, 26th March, 2015. An oral presentation/demo will be arranged as soon as you have submitted your project (details will be coming soon).

Project submission emails should be conducted **through your QUB account and from on the campus network**. Any submissions received through other email accounts will NOT be accepted. For project submission emails, the subject file must commence with: **CSC3054/7054 - project**. The first line of the body of the email should include your name, student number, and module code.

Check that it is submitted on, or before, the specified or agreed due date. Late submissions will only be accepted in exceptional circumstances or where a deferment has been granted in advance in line with the University policies and regulations; otherwise, score deduction will be applied.

Please note that if you want to do something similar to another project (e.g. a final year project or a project of another course module), make sure that there is at least 50% difference between the technical contribution of the current project and that of the other project. Failing to distinguish this difference will result in significant score reduction of the current project. On the other hand, any plagiarism case will lead to a zero mark for the project, following the University policies and regulations.

6. Marking scheme

Your score depends on how well your software solution meets the following requirements:

- A working and maintainable software for your project – 20 marks

- To evaluate whether or not your program is good enough, please see: http://www.dwheeler.com/oss_fs_eval.html
- Functionality: does the program do what you want it to do?
- Cost: does your program need too many commercialised software tools to support?
- Maintenance: is your program easy to be maintained or updated?
- Reliability: is there any leak in the program that causes potential problems in program execution?
- Performance: try different circumstance and see if this works.
- Scalability: does your program handle different size of data?
- Usability: is your application useful to the community?
- Flexibility/customisability: can your program handle an unusual circumstance that is not expected in an ideal environment?
- Easy to operate and friendly graphic interface

- A clearly and well written document to explain the above programs – 5 marks
  - Maximum pages: 10 (from introduction to references). Font: Times New Roman, 11 or larger.
  - Contents of the document: please follow the style/format given in Appendix 1 that has been given on: http://www.ece.rutgers.edu/~marsic/Teaching/SE1/report1.html. Please note that due to the space limit, you will have to use a compact version of "report 1.html" shown in Appendix 1 and generate your own document.
  - An exemplar report is enclosed in the same folder.

- Extra contributions – 5 marks
  - Students can choose whatever they like and then implement it for their project. Note that the extra contributions must demonstrate their relevance to the module and appear to be competent and beyond the state of the art technology.

Appendix 1: Document requirements extracted from "Report.html"

``Part 1:

1. *Customer Statement of Requirements (CSR)*
   a. *Problem Statement*
      A minimum 3-page high-level narrative about your project. The narrative should not be written from the developer's perspective, describing the features of the planned system.
      Rather, put yourself into a customer's role, and write your CSR as if your imagined customer would write it! —Describe the problem that your customer is facing and his or her suggestions about how a software system could help.
      Your CSR should be based on your project proposal, revised and improved as necessary.
      If you're working on an existing project idea, then summarize and rephrase the description given therein.
      You are welcome to borrow anything and everything from the past student projects

posted there; just make sure that you describe explicitly how novel or different your extensions will be compared to the past projects.

   b. *Glossary of Terms*
      List important terms and their definitions to ensure consistency and avoid ambiguity in the system specification. Use the language of the application domain and avoid uncommon terms or define these as well.
      It is helpful to illustrate the complex terms by providing images and graphics to help reader's understanding.
      Another option is to provide web links where to find more complete definitions of your terms.

2. *System Requirements*
   Note: Instead of system requirements, you may wish to write User Stories (write one or the other, but not both).
   a. *Enumerated Functional Requirements*
      Extract the requirements from the customer's narrative and list them in a table, one row per requirement.
   b. *Enumerated Nonfunctional Requirements*
      List, prioritize, and describe the requirements. The non-functional requirements numbering should continue the functional requirements list.
   c. *On-Screen Appearance Requirements*
      For projects that are heavy on graphics (such as a video game) the on-screen appearance makes up the majority of the requirements. Again list, prioritize, and describe the on-screen appearance requirements, but also include a graphic illustrating the requirement. You may find images on the Web or make hand-drawn sketches on paper, then scan them and insert as images into your report.
      Do not spend time polishing these graphics, because polishing is part of Section 4 User Interface Specification (below)

- *Project Management (described in Section 6 below) and References (described in Section 7 below)*

Part 2:

3. *Functional Requirements Specification*
   Derive the use cases based on the requirements from Section 1 and Section 2 above.
   a. *Stakeholders*
      Identify anyone and everyone who has interest in this system (users, managers, sponsors, etc.). Stakeholders should be humans or human organizations.
   b. *Actors and Goals*
      Identify the roles of people or devices that will directly interact with the system, their types (initiating vs. participating) and the goals of the initiating actors.
   c. *Use Cases*
      i. *Casual Description*
         For all use cases that you can think of (based on your System Requirements), write a brief or casual text description. List explicitly the requirements that each use case responds to.
      ii. *Use Case Diagram*
         Draw the use case diagram with all the use cases. Indicate the relationships, such as <<include>> and <<extend>>.
      iii. *Traceability Matrix*
         Show how your system requirements map to your use cases. Calculate the

priority weights of your use cases. The use cases with the highest priority should be elaborated and planned.

- iv. *Fully-Dressed Description*
  Select a few most important use cases and provide detailed ("fully dressed") description. The "most important" use cases are indicated by your traceability matrix.
  Your event flows must show step-by-step every action that the initiating actor ("user") can take while running the given use case.

d. *System Sequence Diagrams*
   Draw the system sequence diagrams for the few most important use cases selected above.

4. *User Interface Specification*
   (Note: If your system prints some forms or generates periodic reports, this is also considered part of the user interface and the format of forms/reports must be specified in this section.)
   The user interface should be specified only for the use cases elaborated in the previous section ("fully dressed" use cases).
   a. *Preliminary Design*
      For a given use case, show step-by-step how the user enters information and how the results appear on the screen.
      Use screen mock-ups and describe exactly what fields the user enters and buttons the user presses. Describe navigational paths that the user will follow.
      In case you are developing a graphics-heavy application, such as a video game, this is one of the most important sections of your report.
   b. *User Effort Estimation*
      Select several typical usage scenarios and, as you walk through the flow of events, count and report the number of mouse clicks and/or keystrokes that are needed to accomplish the task. What fraction of these goes to user-interface navigation vs. clerical data entry?

- *Project Management (described in Section 6 below) and References (described in Section 7 below)*

Part 3:

5. *Domain Analysis*
   a. *Domain Model*
      Show the process of deriving the domain model and then draw the diagram. Provide text description of:
      - i. *Concept definitions*
      - ii. *Association definitions*
      - iii. *Attribute definitions*
      - iv. *Traceability matrix - show how your use cases map to your domain concepts.*
   b. *System Operation Contracts*
      Should be provided only for the operations of the fully-dressed use cases elaborated in Section 3.c), for their system operations identified in Section 3.d).
   c. *Mathematical Model*
      Do you use any mathematical models? E.g., you may use a statistical model for stock price prediction, or a geometric model for computing the trajectories for animate figures in a video game.
      If NO, skip to the next item;
      If YES, describe precisely your model.

6. *Plan of Work*
   Describe what your group is planning to do after submitting report#1 until the end of the semester. Show the roadmap with projected milestones and dates by which you plan to accomplish them. Of course, your plans for the short term (next few weeks) should be much more detailed than further in the future.
   Preferably, you should use Gantt charts for planning and scheduling your project.
   Also include the product ownership description from your project proposal, and provide the breakdown of responsibilities: what each member did so far, is currently doing, will do in the future, including management and coordination activities.
7. *References*
   The list of references should contain exact references and URLs of any material that is used in the project and doesn't come from the textbook. If a reference is listed but not cited/mentioned in the main text, explain briefly in what way it was used."

Appendix 2: some of the MSc students' Group Projects created during 2012/2014:

1. Game App: Battleships
2. Belfast Event Finder App
3. Harry Potter Quiz App
4. HSC: Transforming Your Care App
5. Teacher/Pupil Quiz App
6. Danske Bank Premiership App
7. Game App: LOTS-O-SLOTS