CSCI 3100 Software Engineering
Project Requirement Specification

1. Objective:

The objective of this course project is to practice what you are learning in this CSCI3100 Software Engineering course by designing, implementing, testing, and documenting a Web application. The particular project serves as a vehicle to sharpen your knowledge in Software Engineering and to develop your relevant skills. This course project also introduces students to teamwork, which is a key for successful large-scale software development. Besides, you can take the project opportunity to develop a tech-savvy, customer-driven, and eye-catching software product, as if you are in a start-up IT company.

The project consists of four phases: (1) software initial design, (2) initial coding, (3) completed coding and demo, and (4) final report and documented code. After the submission of completed code, the project team will need to make a demonstration and answer questions regarding the project during the demo on the Demo Day.

2. Project Grouping:

Each project group is composed of 5 students for the whole duration of the project. Some group may have 4 members, but no group should have 6 or more members. All students in a group work together on the same project based on the project requirements defined below. By now you may have chosen group members by yourselves. If so, please visit the front page of the course website, where there is a link for Project Group Sign-up. Sign up your project group accordingly. Note that you can sign up a group only when you have 3 or more members in the group already. At the end of project grouping, we will randomly assign remaining students to form additional groups, or assign them to existing groups which include less than 5 (i.e., 3 or 4) members. Once the group is assigned, you should remain in the group and work with your team members closely for the entire project duration.

3. Project Requirements:

The goal of the project is to develop an advanced Web application with an arbitrary topic, defined by your group. You can image that this project is the killer application of a start-up company created by your team. You are required to use techniques (e.g., requirement analysis, DFD, UML) taught in the lectures in your project development process. In this project, you can try to design and implement as many fancy features as you like, making use of the full power of software engineering techniques, under your topic. You are also recommended to employ as many software engineering techniques taught in the lectures as possible. However, the followings are the basic requirements that your application *must* fulfill:

- Web based access

The application must be actively accessible over the Web, using a Web browser (e.g., Chrome). Your application will provide useful and friendly functionality/service to the human users via Webpage interfaces. There is no particular technological requirement on Web access. However, we encourage you to employ modern Web technologies, such as Node.js, HTML5, CSS3.0, JQuery.ajax, WebSocket, AngularJS and React.

- Database

Either SQL database (e.g., MySQL, or Sqlite) or NoSQL database (e.g., MongoDB, or Redis) must be employed by your application for storing data.

- DFD (Data Flow Diagram)

Requirement specification is fundamental to your project. You are expected to submit your project requirement specification by English description and DFD techniques taught in the class, which explain the key components of your project. You are recommended to employ DFD as detailed as you like in your final report, with proper revision to reflect your actual system specification.

- UML

UML must be employed to describe the key components of your project design. This will also be described in the class. You need to include UML to describe some major components in your initial design. You are also recommended to employ UML as detailed as you like in your final report, so as to reflect the final detailed design of your system.

- Test Cases

You must design some test cases to test the key functionalities of your project. The design of test cases includes black box testing (required) and white box testing (optional), which will be covered in the class.

The followings are some optional features for your references when designing your application:

- **Analysis**. Such as data search (discovering specific services/products among those available), sorting, regression analysis, other statistical analysis, etc. trivial stats (e.g., average, std) will not count.

- **Display**. Visualization of the analysis results using graphs and charts.

- **Access control mechanism** that takes into account the different types/roles of users (customer, system administrator, etc.).

- Support for **managing users** and maintaining user state (e.g., profiles).

- Tailored **mobile** support that can suit at least one mobile platform (e.g., iOS or Android), which can be achieved via standalone mobile application or Web with a specially tailored user interface for mobile devices (e.g., via Bootstrap techniques). Refer to Gmail desktop

version and Gmail mobile version for an excellent example.

- **Finite State Machine** and **Petri Net** can be included to describe non-trivial system behaviors. These are requirement specification techniques, which will be covered in the class. Trivial system behavior, such as the typical login process, will not count. You are encouraged to specify the Finite State Machine and Petri Net of your system in the final report, as a way to describe your system behavior in detail.

When designing your Web application, the most important project feature to keep in mind is that the software product you will develop should require a reasonable programming effort. In order to learn good design, the process must include actual implementation of the design. Therefore, the project must demonstrate a decent programming exercise. Please note that designing passive HTML W pages with no communications between server and clients is not programming. Although your project can include Web page design, that is not enough. On the other hand, designing active Web pages using JQuery.ajax or other advanced frameworks like AngularJS and React for dynamic pages is programming and is perfectly acceptable for the class project. In addition to the programming effort, keep in mind that one key purpose of this course is that you learn how to do modular design of software and how to document the design using symbolic representations, i.e., UML diagrams.

No joint work over any technical aspects of the project is allowed between any two groups/teams. Any problem about the project requirements should be directed to the tutors through electronic mails, newsgroup discussions, or tutorial sessions. The reason for this policy is to enforce team separation for proper credits. This project should be considered as if there is only one single team, namely your team, responsible for your whole project development. No plagiarism is allowed regarding any aspect of the project. Reusing existing designs or codes as part of your project (such as those from the open source) is allowed, but you need to document the reused code clearly. We will also employ professional tools to verify the percentage of existing codes in your project. Note that you should not reuse existing code excessively. Your project mark will be deduced significantly if the percentage of reused existing code exceeds a certain threshold.

4. Project Phases:

There are four project phases described as follows:

(1) Initial Design Phase (5 weeks)
In this phase, each project group will prepare and submit an initial design document to provide high-level descriptions on functionalities, features, and architectural design of your application. Project background, architecture diagrams and brief descriptions of the key system components should be provided. You should also represent these key components of your project using UML (including use-case diagram, class diagram, and sequence diagram). See Appendix 1 for guidance. Feedbacks will be provided on your initial design, and you should consider and possibly revise the project goals before going on to the second phase. You are required submit your project design report by midnight of **February 23, 2018 (Friday)**. You should then continue to develop the

initial design of your project into a detailed design, preparing for the next phase. We don't require you to submit your detailed design, but it should be included in your Final Report as a comprehensive documentation of your project.

(2) Initial Code (3 weeks)
In this phase, you will work as the programmers to implement your own design, and are required to submit your initial code by midnight of **March 18, 2018 (Sunday)**. The objective of this phase is to evaluate your effort in designing the whole project, including architectural design and detailed design, and to make sure the progress of your project is in the right direction. The initial code may consist of ALL major class definitions, interfaces and member function prototypes, reflecting your UML design diagrams. The implementation does not need to be completed or even compliable in this phase; however, you should have at least partial implementation included in your submission. You are required to use **git** version control system for all your coding efforts, as described later in Section 6.2.

(3) Final Code and Demonstration (3 weeks)
In this phase, you are completing your project and are required to submit your final code of the project. Your final code should be self-contained and working. You will need to make a demonstration after the submission of the final code. Project Demo Day is scheduled on **April 4, 2018 (Wednesday)**. Detailed project demo arrangement will be announced in the course website, and the demo schedule will be signed up accordingly (please note the news on the course website). Your final code for demo should be submitted by midnight of **April 3, 2018 (Tuesday)**.

(4) Final Report and Commented Code (4 weeks)
After the demonstration, you are required to prepare and submit a final report and commented code of your project by midnight of **May 4, 2018 (Friday)**. In additional to reporting your comprehensive project, which includes your finalized detailed design, the final report should also show what software engineering techniques/tools you have applied in the project, possible future enhancement of your project, and whatever lessons you have learned. Detailed requirements of the final report are provided in Appendix 1. The revised final code should be commented as detailed as possible, with all known bugs removed.

5. Grading Criteria:

The followings are the project schedule of different phases:

Phase Deliverables	Weightings	Durations	Due Date
0. Project Assignment Team Formulation	--	--	**19 Jan (23:59:59 pm)**
1. Project Design Document	**20%**	**5 weeks**	**23 Feb (23:59:59 pm)**
2. Initial Code	**10%**	**3 weeks**	**18 Mar (23:59:59 pm)**
3. Completed Code and Demo	**50%**	**3 weeks**	**Code submission: 3 Apr (23:59:59pm)** **Demo Day: 4 Apr (full day)**

4. Final Report and Commented Code	**20%**	**4 weeks**	**4 May (23:59:59 pm)**
Total	**100%**	**15 weeks**	

Grading is market-based—whoever offers the most-impressive system receives the highest grade. The reports will be graded based upon the technical content and the clarity of the presentation, and the final revised code will be graded according to its modular structure, comments and cleanness. However, it is not enough to meet all the listed requirements to receive the maximum grade. For example, having a perfect report for a trivial project will result in a very low overall grade. Thus, the overall quality and functionality of the project is the key scaling factor for all other aspects of the grade. Moreover, the techniques and tools you used to develop the project and the test cases and scenarios you designed to verify the system are also important factors considered in your project grade.

Although generally the project grade will be based for the whole team and will not be assigned individually to the members, each team member must be aware that a major part of his or her final project grade depends on the team work. Failures to cooperate with other team members and to invest equitable amount of effort can lead to undesirable outcomes, particularly when other team members raise complaints about the non-participating members.

6. Submission

There are two report submissions (i.e., Initial Design Documentation and Final Report) and two code submissions (i.e., Initial Code and Final Commented Code). The submissions need to meet the following requirements:

6.1. Report submission

Each project group should Email the softcopy of the report to csci3100@cse.cuhk.edu.hk before the deadlines. The followings are the required Email titles and names of the attached documents of different phases:

"Group** Project Initial Design Report"

"Group** Final Report"

Please replace the "**" with your group ID.

6.2. **Code submission.**

ALL your project stuff (including source code, images, flashes, databases files, etc.) should be conducted by **git**. Tutors will set up a git server for hosting your project before the coding phases start. Further information will be provided in the related emails or information on the website.

Git actions play an important role in evaluating your project coding phases. You should take advantage of the version control system to support the development and documentation of your project. You MUST submit your project to the git server, and faithfully record your coding activities. We will NOT accept any code submissions via other approaches. Moreover, the tutors will check your version control logs when marking your coding efforts.

Appendix 1 Guidance for documentation

A total of two reports (i.e., Initial Design Documentation and Final Report) will be submitted by each project group. The reports should be submitted by the whole team (one report per team) and the report will be graded as a whole for the team members.

Initial Design Documentation focuses on the high-level system functionality and architectural design. In typical software engineering project, there should be a detailed design document which include detailed classes diagrams, component descriptions, pseudocodes, and programmers should be able to implement the system based on this design document. We skip this for reducing this project development workload (except for DFD/UMLs to show your detailed design), and defer the detailed design to the final report. In the final report, besides the design details, the work assignment of project members, code statistics (e.g., lines of code, number of functions, etc), project highlights, test case design and results, and lessons learned should be reported.

Each document should contain three parts: cover page, table of contents, and detailed contents.
The Cover page must contain the following information

- Name of Document
- Project Title (You can create your own project name under the assigned topic)
- Document Version Number
- Printing Date
- Group ID
- member names and SID
- Department & University

The followings are the detailed outlines for the two reports:

1.1 Initial design outline (5-10 pages)

The initial design document is mainly focused on the purpose of the product you are designing, and its high-level descriptions. You should describe the objective, expected customers and market, features, and architectural design of your Web application project. Moreover, the project background, architecture diagram and brief descriptions of the system components should be provided. You are also encouraged to apply DFD to specify your product and UML to depict your initial design, so that the architectural and major components of your product can be clearly described. The recommended outline is listed as follows.

1 INTRODUCTION
- 1.1 Project Overview
- 1.2 Objective
- 1.3 Expected Customers and Market
- 1.4 System Features

2 BACKGROUND

Emphasize why your design the product, and its most attractive functionality/features.

3 SPECIFICATION (e.g., DFD)

4 SYSTEM ARCHITECTURE

3.1 Architecture Diagram

3.2 System Components

3.3 Description of Major System Components by UML

1.2 Final report outline (30 or more pages)

Your final report should include five sections (introduction, system architectural design, detailed description of components, user interface design, and test plans/cases). The detailed requirement of each section is listed below. A recommended report structure is attached at the end of this document. You are not restricted to the recommended report structure, but your own structure should contain AT LEAST the following listed sections and sub-sections.

Introduction

In this section, you should describe the general overview of your project. Contents that you should report in this section include background, group members and the workload of each group member, the problem to be solved, your motivation, project key features, etc.

In the *highlights* sub-section, you can include major attractive functionality and advanced features in your project. The expected customers and users can also be included, and you can elaborate why they will enjoy your project.

In the *project statistics* sub-section, you should indicate the LOC (lines of code) and McCabe's number (to be taught in class) of your project, including the main function and other major functions. These metrics can be as extensive as possible, and down to module level. Present your project statistics clearly, such as using a tabular format.

System Specification

In this section, you should describe the system specification of your project. You have to include the DFD diagrams to introduce the operational specification of your project. Besides, you are encouraged to use FSM and Petri Net to explain non-trivial system operational behaviors, for the whole or some selected parts of the system.

System architectural design

In this section, you should describe the architectural design of your project. Contents that you should concern in this section include: whole system architecture overview, key components, key interface description etc. Moreover, you should indicate the connections among different key components.

Detailed description of components by UML

In this section, you should use UML to provide the detailed description of at least the key components. Contents that you should concern in this section: UML diagrams (including use-case diagram, class diagram, and sequence diagram) of the major class, functionality of the component, list of major function, etc.

User interface (UI) design

In this section, you should present the UI of your project. This section could be like a user manual of your project. You should teach and guild the users by walking through your UI operations. The use of screenshots is needed to indicate your UI overview and the result after certain user actions.

Testing

In this section, you should present the test plan, test case design, and test result of your project.

In the *test overview and test plan* sub-section, you should contain your test approach, features to be tested, plan for testing, testing tools if any, and the testing environment.

In the *Case-N* sub-section, you should concern the objective of each test case, input, expected outputs, Pass/Fail criteria, and so on.

Lessons learned

In this section, please describe what lessons you have learned from the software engineering exercise of this project, what would you do differently if you can re-do the project again, and any positive and negative experience you have with the project.

Recommended final report structure

Cover Page
Table of Contents
1 INTRODUCTION
    1.1 Project Overview
    1.2 Objective
    1.3 Highlights
    1.4 Project Statistics
2 SYSTEM ARCHITECTURAL DESIGN by DFD
    2.1 System Architecture
    2.2 DFDs
3 DETAILED DESCRIPTION OF COMPONENTS by UML
    3.n Component-n
        3.n.1 Structural Diagram
        3.n.2 UMLs
        3.n.3 Functionality
        3.n.4 Procedures and Functions
4 USER INTERFACE DESIGN
    4.1 Description of the User Interface