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 modern mobile-enabled 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 team work, which is a must for large-scale software development. Besides, you can take the project opportunity to develop a tech-savvy, customer-driven, eye-catching software product, as if you are in a start-up IT company.

The project consists of four phases: (1) software design, (2) initial coding, (3) completed coding 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.

2. Project Group:

Each project group is composed of 4-5 students for the whole duration of the project. All students in a group work together on the same project based on the project requirements defined below. By now you should have chosen group members by yourselves and registered in our grouping system, which is also a demo project for this course. 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 easy mobile access, such as online book/music/video store, online backup service, online collaboration service, social app, etc. (use your imagination). You can image that this project is the killer application of a start-up company created by your team. The most important objective to keep attention is that how to serve as many people as possible, and amuse your customers. Some previous sample projects are listed in Appendix for your reference. 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. However, the followings are the basic requirements that your application *must* fulfill:

- Web based access.

The application must be 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. In order to learn modern Web technology and drop the old pattern of Web development, the server-side program must be built on Node.js, which will be introduced in the tutorial section with details and demos. This is a technological requirement and no other somewhat obsolete Web

framework (like PHP, or Django) is allowed.

- Database

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

- Tailored mobile support

Your application should suit at least one mobile platform. Android is recommended; however, iOS is also acceptable if you already know how about it. This can be achieved via either standalone application or Web with a specially tailored user interface for mobile devices Specifically, your UI should take the limited screen size into account and make use of the available space wisely and efficiently. The place of navigation should suit the convention of mobile application usage. The design should be visually appealing and user-friendly. Refer to Gmail desktop and Gmail mobile for an excellent example.

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.

- **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).

- **Online help** capability and error reporting/tracking capability.

Although not obligatory, you are encouraged to employ various well-known technologies and tools (e.g., AJAX, WebSocket, Jade etc.) in your project. You are also encouraged to explore more advanced Web frameworks built on the top of Node.js. For example, Primus and DataChannel.io are two powerful tools to build real-time Web application and KnockoutJS will help you to implement fancy user interface.

The tutorials will cover related techniques and tools, such as JavaScript, CSS3, HTML5, Node.js, Express Framework, Git, etc. In the tutorials, a simple Web application designed for managing tutorial sessions and project grouping will be demonstrated to show you how to use these useful tools and techniques to built your Web application based on Node.js.

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 include a decent programming exercise. Please note that designing passive HTML

web pages is not programming. Although your project can include Web page design, that is not enough. On the other hand, designing active Web pages using AJAX or any other techniques 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.

Four Web application examples are provided in Appendix 2 for your reference. However, you are encouraged to design your own project by including considerable enhancement over these examples or by choosing completely different applications of your own. It is advisable to come up with something fresh and innovative to keep yourself interested in its realization and to make good impression on the project markers (i.e., the lecturer and the tutors).

No joint work over any technical aspects of the project is allowed between any two teams. Any problem about the project 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 should document that properly.

4. Project Phases:

(1) 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 architecture design of your application. Project background, architecture diagram and brief descriptions of the system components should be provided. See Appendix 1 for guidance. Feedbacks will be provided on your initial design and the students should reconsider and possibly revise the project goals before going on to the second phase.

(2) Initial Code (4 week)
In this phase, you will work as the programmers to implement your own design, and are required to submit your initial code. The objective of this phase is to evaluate your effort in designing the whole project architecture 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. The implementation is NOT the primary concern in this phase; however, you should have at least partial implementation included in your submission. You are required to use version control system for all your coding efforts, as described later in Section 6.2.

(4) 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 is scheduled on April 14, 2016 (Thursday). Detailed demonstration 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).

(5) Final Report and Commented Code (3 weeks)
After the demonstration, you are required to prepare and submit a final report and commented code of your project. In additional to reporting your comprehensive project, the final report should also show what software engineering techniques/tools you have applied in the project, and what 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	--	--	**18 Jan (on Web)**
1. Project Design Document	**10%**	**5 weeks**	**24 Feb (midnight)**
2. Initial Code	**5%**	**4 weeks**	**23 Mar (midnight)**
3. Final Code and Demo	**60%**	**3 weeks**	**14 Apr (full day)**
4. Final Report and Commented Code	**25%**	**3 weeks**	**6 May (midnight)**
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 that influence your project grade.

Although generally the project grade will be based for the whole team and will not be assigned individually to members, each student must be aware that a major part of his or her final grade depends on the team project. Failures to cooperate with other team members and to invest equitable amount of effort can lead to undesirable outcomes, particularly when 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** Initial Design"

"Group** Final Report"

Please replace the "**" with your group ID. Each project group is required to submit one initial design softcopy, one final design softcopy, and one final report softcopy to csci3100@cse.cuhk.edu.hk before the specified deadlines.

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 tutorials.

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 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 architecture 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, 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
- 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 architecture design of your Web application project. Moreover, the project background, architecture diagram and brief descriptions of the system components should be provided.

1 INTRODUCTION
    1.1 Project Overview
    1.2 Objective
    1.3 Expected Customers and Market1.4 System Features
2 BACKGROUND
    Emphasize why your design the product, and its most attractive functionality/features.
3 SYSTEM ARCHITECTURE
    3.1 Architecture Diagram
    3.2 System Components

1.2 Final report outline (30-50 pages)

Your final report should include five sections (introduction, system architectural design, detailed description of components, user interface design, and test). 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) of your project and the number of functions, etc. 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 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, data/control flow diagram, key interface description etc. Moreover, you should indicate the connections among different key components.

Detailed description of components

In this section, you should describe the detailed description of each component. Contents that you should concern in this section: UML diagram of the major class, functionality of the component, list of major function, implementation (pseudo-code), interfaces, 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.

Test

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

In the *test overview* sub-section, you should contain your test approach, features to be tested, 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 test coverage.

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
    2.1 System Architecture
    2.2 System Interface Description
3 DETAILED DESCRIPTION OF COMPONENTS
    3.n Component-n
        3.n.1 Structural Diagram
        3.n.2 Functionality
        3.n.3 Procedures and Functions
4 USER INTERFACE DESIGN
    4.1 Description of the User Interface
    4.2 Screen Images
    4.3 Objects and Actions
5 TEST
    5.1 Test overview
    5.n Case-n
        5.n.1 Purpose
        5.n.2 Inputs
        5.n.3 Expected Outputs & Pass/Fail Criteria &Coverage
6 LESSONS LEARNED
7 CONCLUSION

Appendix 2: Sample Projects:

Online Calendar

Description

This project aims to provide a powerful online calendar. Besides recording the events and notifying the user, the online calendar could synchronize events from multi-sources (e.g., mobile, PC, pad). The system should also provide simple storage functions to store event related files.

Optional Functionality

You are recommended to implement the following functionality. It is not necessary to implement all of them, though. It is also possible to add new functionality by yourself.

Calendar: The user could add/drop/update events on the system.
Cooperation: The users could share the calendar with colleagues. One event could involve several people as long as they accept the event.
Notification and alarm: The system should provide various kinds of notifications (e.g., full screen figure, ring). The system could provide wake up alarm as a plus. The system may also send SMS to notify users the events.
Storage Database: The users could upload and download files related to the events. The files are stored on the servers and users could download the files everywhere.
Calendar Export: Users may have a way to download/export the online calendar into a CSV file for other use.
Adaptation for Mobile: A tailored UI for mobile can also be implemented. The designer should modify the grid style size, layout, notification tools, font size and etc. to give a good user experience on the mobile phone browsers.

Online Estate Sale/Rental Platform

Description

This application provides an effective web-based platform for sharing estate. Landlords and house holders can publish their estate for sale or for rental easily. Also, tenants and house buyers can search for their ideal choices with the on-site search engine. Useful estate statistics such as average estate price in specific region can also be provided to users as reference.

Optional Functionality

You are recommended to implement the following functionality. It is not necessary to implement all of them, though. It is also possible to add new functionality by yourself.

Login System: User can register for an account with different identities such as landlords and tenants. The login sessions and cookies should be carefully managed so that people only need to login once every time they use the platform. Also, their personal information should be stored on the server with encryption.

Advertisement Board: To allow the landlords to publish their rental or sale post, the advertisement board should be implemented for displaying those posts. The information shown in the board is based on the database which stores estates informations like the price, owner, location, pictures, etc. An information filter or search engine should also be included for tenants and buyers to search for estates they want.

Advertisement Management: Landlords should also have an interface to edit or take down their existing posts, or publish new post. Web forms should be filled by the landlords for any edition. Previous information should also be shown in the form for the reason that users usually keep most information unchanged.

Estate Statistics: The platform can also provide some useful statistics for users. It may include estate price in different regions, the price history in recent days, confidence evaluation of each user, distribution of estates on the map, etc.

Adaptation for Mobile: A tailored UI for mobile can also be implemented. The designer should modify the form style, font size, photo displaying layout, information arrangement and etc. to give a good user experience on the mobile phone browsers.

Online 2D/3D Shooting Game Platform

Description

This platform is an online game platform for shooting game with friends. You may also develop your own game platform, e.g. board game, chess game or strategy game. It's important to make it robust and efficient.

Optional Functionality

You are recommended to implement the following functionality. It is not necessary to implement all of them, though. It is also possible to add new functionality by yourself.

Login System: Users need to sign in before entering the gaming system. The system may require a password for security reasons. Login session should be stored in database to avoid duplicate login.

Game Engine: This engine should provide a well-designed game interface, which connects client machines with web sockets or other P2P techniques on web. A host machine is recommended for computing the whole game logic and ensuring that all machines are still online and follows the same logic and pace.

Chat Room: You can also develop a chatting room in the game interface to allow people to perform group chatting when playing game. The implementation of chatting room is similar with the game engine.

Waiting Room: You can develop a waiting room engine to help users to group together before starting the game.

Robustness Handling: You need to consider and handle unexpected situations to make your system robust. For example, some player may quit game or close the browser, you need to consider what other players' program should response in this kind of situation.

Adaptation for Mobile: A tailored UI for mobile can also be implemented. The designer should modify the web socket implementation, game board, buttons, controls, robustness for phone call interruption and etc. to give a good user experience on the mobile phone browsers.

Online Air Ticket Booking System

Description

The goal of this project is to implement an air ticket booking service. The system will query different airlines, sort and merge the results. Provide the service of querying, ordering air ticket and booking hotel.

Optional Functionality

You are recommended to implement the following functionality. It is not necessary to implement all of them, though. It is also possible to add new functionality by yourself.

Data Collection: The system should look up several airline databases to collect the data.
Search Engine: The system should provide flexible routines that some users would prefer cheaper routines while other users may prefer shorter time. The search engine could also provide transfer of different airlines.
Hotel booking: The system should allow user to book hotels. Better to give some discounts if the user purchases the air ticket together.
Personalized recommendation: Record the user booking histories in a database and learn the user preference. Recommend airlines to users according to the user habits.
Travel Advices: The system could provide travel advices if the users are planning a trip. Users could also share their travel experience with others.
Online Payment: The system should provide various ways of payment and guarantee the security.
Adaptation for Mobile: A tailored UI for mobile can also be implemented. The designer should modify the form style, font size, information layout, notification, chart/graphics displaying and etc. to give a good user experience on the mobile phone browsers.