Southeast Missouri State University

CS 631
Advanced Software Engineering
Summer 2021

Project Manual
(v1.0)

Instructor: Dr. Wee Wee Sim

Overview

1. *INTRODUCTION*

Students in this course will undertake group projects to solve real-world problem. They will solve the problem(s) with tools and practices commonly used in the industry. Students will traverse the entire Software Development Life Cycle (SDLC) to deliver a medium-size project in a team. Students will contribute to different types of SDLC work products e.g. requirements, design, code, test cases. The project is to be scoped such that it can be completed by a group of 3-4 students within 6 weeks.

The "CS631 Project Manual" document provides students guidance about when to do what by setting requirements for each of your project deliverables. But you must organize your own work and manage your own time in the project. The project manual will be updated as the semester proceeds. The class will be notified via announcement when there is a new updated version.

2. *ROADMAP*

There are **THREE** project deliverables (PDs) to be completed and submitted by each group during the course. The three deliverables are highlighted in Table 1.

Table 1. Project Deliverables

PD#	Focus	Deliverables	Due Date

PD1	Requirements Elicitation	• Software Requirements Specification (SRS)
		  ○ Functional and Non-Functional Requirements
		  ○ Use Case Model
		  ○ Use Case Diagram
		  ○ Use Case Descriptions
		  ○ Data Dictionary
		• Project Plan (using Microsoft Project)
		  ○ Work Breakdown Structure (WBS)
		  ○ Gantt Chart
		
		Note: the WBS and Chart can be included in the SRS document.
			5/31/2021
			5:00 PM

PD2	Requirements Analysis and Modeling, Architectural Design	• Context Model
		• Class Diagram
		• Dynamic Model
		  ○ Sequence Diagrams (of all the use cases)
		  ○ State-Chart Diagram
		• Architectural Diagram
		• User Interface Design
		
		Note: the model/diagrams can be included in the SRS document.
			6/14/2021
			5:00 PM

PD3	Software Testing	• Test Cases and Testing Results
		  ○ Black Box Testing (Use Cases)
		  ○ White Box Testing (Source Code)
		
		Note: the test cases and testing results can be included in the SRS document.
			6/21/2021
			5:00 PM

The **Final Exam** group project required documents are listed in Table 2.

Table 2. Final Exam–Final Group Project Requirements

	Requirements	Due Date

Final Exam	• Complete Report (SRS document)
	  ○ Functional and Non-Functional requirement statements.
	  ○ Use Case Model (use case diagram and use case descriptions) – *refer to PD1 and PD2 sections in this document*
	  ○ All required test cases – *refer to PD3 section in this document.*
	• Implementation
	  ○ Complete source code
	• 15-20 minutes Recorded Video Presentation
		6/24/2021
		5:00 PM

3. PROJECT GROUP MEMBERS ASSIGNMENT

Typically, in a software development work environment, you do not have much liberty to choose who you want to work with. The main goal of working in a group is to produce the product successfully as required by the client.

Therefore, for the group project, all groups' assignments are pre-arranged (randomly) prior to the beginning of week 1. Each group consists of 3-4 students (depending on final enrollment of the course) working together to complete the software application. It is not recommended to change to another group due to the tight timeframe for this short 6-week summer session.

The instructor will post a list of students and the students' groups on the course page on Canvas in the beginning of week 1. Students from each group will communicate with his/her teammates, appoint a group leader, and decide on what software application to work on. These activities must be done as early as possible during week 1.

4. ASSESSMENT

For the assessment and grading scale of the Final Exam project group and project deliverables, please refer to the syllabus posted on the course page on Canvas.

Project Description

1. *INTRODUCTION*

Students in this course will work in groups of 3-4 students to develop some working software applications to address a real-world business need. Students will put into practice concepts and skills learnt in this course as well as in the past related courses, in the context of a real-world problem. They will solve the given problem(s) with tools and practices commonly used in the industry. Students will traverse the entire Software Development Life Cycle (SDLC) to deliver a medium-size project in a group. Students will contribute to different types of SDLC work products e.g. requirements, design, code, test cases.

Students can choose any type of software applications they will like to develop. Examples of software applications are video store application, music store application, book store application, travel guide application, real estate application, and health guide application.

The requirements for the software application to developed for this course group project are the following:

- The software applications can be developed for regular desktop, mobile, or web platform.
- The software applications can be developed using any type of programming languages of their choice.
- The final software applications must be able to run (i.e. a working prototype) on regular desktop, mobile, or web platform.
- The final software applications must be free of virus/malware.
- Students can use any UML modeling tools/IDEs to create different kinds of model as required in the deliverables. Examples of modeling tools are Visual Paradigm, Microsoft Visio, Sparx Systems, MagicDraw UML, IBM Rational or Rational Rose, Netbeans IDE, Altova Umodel, etc.

Project Deliverable PD1

1. ELICIT AND DOCUMENT REQUIREMENTS

1.1. Determine the **target users** of your application. Elicit **functional** and **non-functional** requirements of the application. Note that in doing this you may wish to have some of the team members act as customers and the rest of the team be the project development team. This is an artificial way of defining the requirements but within the constraints of this course project it is a practical way to proceed.

1.1.1. A **REQUIRED** functional requirement is the **User Account Registration** requirement; specifically,

- User account password: 8 characters.
- User account password: combination of lower case, upper case, and special characters.
- Three invalid entry attempts of user account password: account blocked.

1.2. Document all the requirements in appropriate and **proper technical format** (i.e. proper requirement statements format). The requirements should clearly state who performs what system functionality, taking what input and producing what output.

1.3. **Atomise** the requirements such that they are verifiable and traceable. You will be writing test cases in project deliverable 3 to verify the requirements. You will also have to demonstrate traceability from requirements to the final product.

1.4. Document important terms of your application (e.g., user, device, input, output) in a **data dictionary**. Explain each term with a brief description. Identify attributes of each term and relationships between terms.

2. VISUALIZE AND REFINE REQUIREMENTS WITH USE CASE MODEL

2.1. From the set of functional requirements, identify the preliminary **Use Cases**. Depict them on a **Use Case diagram** using the UML modeling tool.

2.1.1. A **REQUIRED** use case for your project is the **User Account Registration** use case.

2.2. Due to the short timeframe for the summer session, you are not required to create an extensive (or complete) use cases for your application. A minimum of **FOUR** use cases are required for the project. One of the four use cases **MUST** include the **User Account Registration** use case.

2.3. For each Use Case, start writing the **Use Case Description** about how the user interacts with the system to carry out the system functionality. As a rule-of-thumb, each Use Case should have a maximum of 6~7 steps in its flow of events. A small (1~2 steps) Use Case indicates that the functionality has been sliced too finely; a large Use Case can be further broken down.

2.3.1. A **REQUIRED** use case description for your project is the **User Account Registration** use case description. The requirements for the User Account Registration are stated in 1.1.1.

2.4. Iterate over your Use Case diagram to identify **included** Use Cases, **extended** Use Cases, and **generalization relationships**, if any.

2.5. A **Use Care Description template** is available to provide you a guide in creating the use case descriptions for your project. The template is posted in Week 1 module on the course page on Canvas.

3. DOCUMENT REQUIREMENTS IN SOFTWARE REQUIREMENT SPECIFICATION (SRS)

3.1. All proper requirement statements should be documented using the provided **Software Requirement Specification (SRS)** document. An SRS template is available and posted in Week 1 module on the course page on Canvas.

4. CREATE PROJECT PLAN

4.1. Develop the project **Work Breakdown Structure (WBS)**, which is a visual and hierarchical decomposition of your project into manageable chunks. Your WBS shows all the project parts in an organized chart.

4.2. Create a **Gantt chart** from the Work Breakdown Structure (WBS) and track the tasks across time. Your Gantt chart should show the start and finish date of each task, their dependencies, and their relationship to each other in terms of sequencing.

4.3. There are a number of project tools you can use to create the WBS and Gantt Chart. Once such tool is the **Microsoft Project**.

5. DELIVERABLES

- **Software Requirement Specification (SRS)**
  - Documentation of functional and non-functional requirements
- **Use Case Model** (Initial)
  - Use Case Diagram
  - Use Case Descriptions
- **Data Dictionary**
- **Project Plan**
  - Work Breakdown Structure (WBS)
  - Gantt Chart

Submit all the required documents and diagrams on Canvas by **5/31/2021 5:00 PM**.

Note that the initial Use Case Model will be refined and elaborated as your application evolves during the course of the semester.

Project Deliverable PD2

1. REFINE USE CASE MODEL

1.1. Finalize your Use Case Diagram with clear definition of actors and use cases.

1.2. Keep refining the Use Case Descriptions. Describe clearly use case precondition, flow of events, and alternative flows.

1.3. Expect to re-visit your Use Case Diagram and User Case Descriptions to clarify and add details as you build Context Model and Dynamic Model.

2. BUILD THE CONTEXT MODEL AND THE CLASS DIAGRAM

2.1. Develop a **Context Diagram** of your application that defines the boundary between your application (or system) and its surrounding environment, showing the entities that interact with your application (or system).

2.2. From the Use Case Descriptions and the terms in the Data Dictionary developed in PD1, identify the classes, their attributes, and operations, and create the **Class Diagram**.

2.3. In the Class Diagram, depict the classes and the relationships between them, e.g. association, generalization, aggregation (if appropriate).

3. BUILD THE DYNAMIC MODEL AND THE USER INTERFACE (UI) DESIGN MODEL

3.1. For **each** use case in your Use Case Diagram, model the interaction between the classes to enact the flow of events in the Use Case Descriptions. Specify the messages and parameters passed between classes in the **Sequence Diagrams**. Typically, at least one sequence diagram is created for each use case.

3.2. Model the system behavior in a **State-Chart Diagram**. The State-Chart Diagram will be based on the **User Interface (UI) Design** that you will also need to create for your application. Refine and further specify your user interface design.

4. DETERMINE APPROPRIATE ARCHITECTURAL MODEL

4.1. Determine the structural design of your application (or system) and construct (draw) the chosen **Architectural Model**. The architectural model could be either software architecture or system architecture. Examples are layered architecture, service-oriented architecture, Model-View-Controller architecture, and client-server architecture.

5. DELIVERABLES

- **Context Model**
- **Class Diagram**
- **Dynamic Model**
  - Sequence Diagrams (of all your use cases in the Use Case Diagram)
  - State-Chart Diagram
- **User Interface (UI) Design**
- **Architectural Diagram**

Submit all the required diagrams on Canvas by **6/14/2021 5:00 PM**.

Project Deliverable PD3

1. DESIGN TEST CASES

1.1. Design test cases using **Black Box** and **White Box Testing** techniques.

- Black Box Testing: test requirements and specifications.
- White Box Testing: test implementation details (source code).

1.1.1. REQUIRED Black Box Testing test cases: **User Account Registration** use case.

1.2. Techniques you can use for Black Box testing are: **Equivalence Class** and **Boundary Value Testing** techniques.

1.3. Design **White Box Testing** test cases using **Basis Path Testing** technique.

1.3.1. REQUIRED White Box Testing test cases: **ONE** important method that implements some complex application logic.

1.4. Execute your test cases and document the testing results. Note that you can develop automatic test cases using proper automatic testing framework, for example, JUnit.
Or you can perform manual testing guided by your test cases.

1.5. Your test cases should include the following components (in table format):

- Test Case ID
- Test Case Description
- Tester
- Test Date
- Test Input
- Expected Output
- Actual Output

2. DELIVERABLES

- **Test Cases and Testing Results**
  - Black Box Testing
  - White Box Testing

Submit all the test cases and test results on Canvas by **6/21/2021 5:00 PM**.

Final Exam (Final Project Submission)

The final exam will be the final project submission from each project group. Each project group will submit a final **REPORT** that includes the **documentation** and **implementation (source code)** of the software application developed during the semester. The documentation includes the final documents, diagrams, and test cases and test results from all the three deliverables.

In addition to the final report, each project group will also submit a **15–20 minutes** recorded video presentation of the software project.

The deadline to submit the final exam group project report and recorded video: **THURSDAY 24 June 5:00 PM**.

Please refer to the syllabus "(18) Final Exam" section for information regarding the final exam, including the **peer evaluation** requirement.