TORRENS UNIVERSITY AUSTRALIA

ASSESSMENT 2 BRIEF	
Subject Code and Title	SEP401 Software Engineering Principles
Assessment	Assessment 2: Problem Analysis – Software Design Specification Document
Individual/Group	Individual/Collaborative
Length	Maximum 50 Pages
Learning Outcomes	This assessment addresses the Subject Learning Outcomes outlined at the bottom of this document.
Submission	Due by 11:55pm AEDT Sunday of Module 9 (Week 9).
Weighting	35%
Total Marks	100 marks

Context

The Software Design Specification (SDS) document is a written description of the design of the software product that a software designer provides the software development team. It is used for recording design information and communicating that design information to key design stakeholders.

As part of the software development team, software designers and software architects make design choices, document and communicate these choices to their stakeholders based on the approved software requirements.

This assessment will assess your understanding of software design and will further develop your skills and knowledge in designing the entire software, identifying the system architecture, each individual high-level component, and creating the detailed design.

Instructions

1. Revise your Software Requirements Specification (SRS) document based on the feedback provided by your facilitator.
2. Using the revised SRS document, develop an SDS document.
3. Use the Software Design Specification Document Template provided: You can add/remove items in the template depending on the applicability on your project.

4. You can use any drawing tool in creating your requirements modelling diagrams.

5. You will be assessed on the correctness and completeness of your document:

    a. Revised SRS Document
    b. Introduction and Design Considerations
    c. User Interface Design
    d. System Architecture
    e. Detailed Design

Submission Instructions

1. You will be submitting **two** documents. Your revised **SRS** and **SDS** document.
2. Submit the two documents in pdf format via the Assessment link in the main navigation menu in SEP104 Software Engineering Principles. The Learning Facilitator will provide feedback via the Grade Centre in the LMS portal. Feedback can be viewed in My Grades.

Assessment 2 Rubric

Assessment Attributes	**Fail (Unacceptable)** **0-49%**	**Pass** **(Functional)** **50-64%**	**Credit** **(Proficient)** **65-74%**	**Distinction** **(Advanced)** **75 -84%**	**High Distinction** **(Exceptional)** **85-100%**
Revised SRS Document **10%**	No changes were made based on the feedback provided.	Minimal changes were made, further revisions are needed.	Some changes were made with 25% of the work still needs improvement.	All changes were made based on the feedback provided but did not make changes on the affected sections.	All changes were made based on the feedback provided and revised all other sections that were affected by these changes.
Introduction and Design Considerations **10%**	Introduction and overall description were inadequate. The system and/or its purpose were explained. Design considerations were not outlined.	Introduced the project topic but not clearly presented. Presented limited understanding of context and/or purpose of the software project. Discussion of design considerations were limited.	Introduced the project topic and explained the significance of the project. Discussion of design considerations were outlined, and some explanations were provided.	The project topic and purpose of the project were described in detail. Discussion of design considerations were outlined in detail including reasons and explanations. Diagrams were sometimes used to support the explanation.	Exceptional quality and completeness of presentation of introduction and purpose of the project. Detailed and clear discussions of design considerations, providing pros and cons and reasons for the decisions. Diagrams were used appropriately to support the explanations.

User Interface Design **20%**	No element of the interface design fit the purpose of the project, or elements of the user interface design were poorly designed that they don't work together for the intended purpose.	Minimal level of creativity shown in the user interface design. Elements of the user interface design were not cohesive. Some of the parts work together but it is not intuitive. Navigation design is inconsistent. Showed some application of user interface design principles but showed limited understanding of its application.	Demonstrated lower level of creativity in the user interface design process. Satisfactory user interface design with some parts needing improvement. Navigation design is somewhat consistent. Most of the elements in the user interface design work together and were somewhat intuitive. Showed application of user interface design principles with some incorrect application. Demonstrated some understanding of the application of the design principles.	Demonstrated high-level of creativity in most parts of the user interface design. Well-developed user interface design with minimal corrections needed. Navigation design is consistent. Most of the elements in the user interface design work together and were intuitive. Showed mostly correct application of user interface design principles. Demonstrated understanding of the application of the design principles.	Highly sophisticated and creative user interface design was presented. Thoroughly developed layout and user interface that were consistent across the design. User interface design elements used enhanced the aesthetics and/or functionality of the application. Showed correct application of user interface design principles. Demonstrated a deep understanding of the application of the design principles.
System Architecture **30%**	System architecture design was poorly developed and incorrect. High-level design diagrams were not appropriately used and/or diagrams were highly redundant to the point that they are difficult to comprehend	Basic design solutions were presented with limited exploration of techniques and application of principles. Assumptions and/or rational were lacking and key trade-offs were missing. System architecture was basic, and most parts were	Fundamentally sound design solution with moderately creative use of concept, fundamentally appropriate technique, and adequate application of principles. Some assumptions and rationale lacking. System architecture plan was adequate with some parts	Interesting design solution showing consistently creative development of concept, original development of technique and original application of principles. Key assumptions and rationale were discussed. Key trade-offs were presented.	Compelling design solution showing highly original creative development of concept, innovative application of techniques and exemplary use of principles. Key assumptions and rationale were discussed. Key trade-offs were successfully analysed and defended.

	Diagrams were not well labelled and were not at an appropriate level of abstraction.	inconsistent with what was stated in the requirements document. Subsystem interfaces were not designed in detail. Interfaces surrounding the environment were not thoroughly considered. High-level design diagrams somewhat used appropriately. Diagrams included some unhelpful redundancy, but the general representations were still readily comprehensible. Labels in the diagrams were often cryptic and makes it difficult for the reader to understand.	inconsistent with what was stated in the requirements document. Subsystem interfaces were somewhat designed in detail. Interfaces surrounding the environment were considered but lacked elaboration. High-level design diagrams somewhat used appropriately. Diagrams appropriately used some of the abstraction features of the notation to minimize useless redundancy. Diagrams were mostly well labelled, with no more than 25% cryptic labels. Diagrams were generally at an appropriate level of abstraction, though a stakeholder familiar with the problem domain might need some guidance to understand them.	System architecture was complete with some minor incorrect details and/or inconsistencies with what was required but did not affect the overall quality of the design. Subsystem interfaces were designed in detail. Interfaces surrounding the environment were considered and elaborated. High-level design diagrams were used appropriately. Diagrams appropriately used the abstraction features of the notation to minimize useless redundancy. Diagrams were mostly well labelled, with no more than 10% cryptic labels. Diagrams are generally at an appropriate level of abstraction, a stakeholder familiar with the problem should be able to understand them.	Exemplary design of system architecture. It was thorough, consistent, complete and correct. Subsystem interfaces were designed in detailed and elaborated. Demonstrated application of critical analysis when considering interface surrounding the environment. High-level design diagrams were used appropriately and successfully analysed and defended the use of diagrams. Diagrams consistently used the appropriate abstraction features of the notation to minimize useless redundancy. The diagrams demonstrated a deep understanding of the solution to the problem. Diagrams were well labelled and at an appropriate level of abstraction so that stakeholders familiar with the problem domain could readily understand them.

Detailed Design **30%**	Poor detailed design and/or did not meet the software requirements. Diagrams were not provided or were too limited.	Acceptable detailed design. It focused on minimally meeting the functional requirements and lacked elaboration. Took into account some key constraints but design is not flexible and were not built to evolve. Described some of the necessary algorithms but not did not address most of the significant requirements. Detailed-design diagrams somewhat used appropriately. Diagrams included some unhelpful redundancy, but the general representations were still readily comprehensible. Labels in the diagrams were often cryptic and makes it difficult for the reader to understand.	Satisfactory design. Detailed design addressed most functional requirements. Functionality was somewhat described but lacks details. Took into account most key constraints. Design showed some flexibility to evolve. Described most of the necessary algorithms but missing for some significant requirements. Detailed design diagrams somewhat used appropriately. Diagrams appropriately used some of the abstraction features of the notation to minimize useless redundancy. Diagrams were mostly well labelled, with no more than 25% cryptic labels. Diagrams were generally at an appropriate level of abstraction, though a stakeholder familiar with the problem domain might need some guidance to understand them.	Good and flexible design meeting all functional requirements. Considered several important constraints and design showed flexibility built-in to evolve. Described all of the necessary algorithms for all significant requirements. Detailed design diagrams were used appropriately. Diagrams appropriately used the abstraction features of the notation to minimize useless redundancy. Diagrams were mostly well labelled, with no more than 10% cryptic labels. Diagrams are generally at an appropriate level of abstraction; a stakeholder familiar with the problem should be able to understand them.	Excellent design. Detailed Design addressed all functional requirements and were designed to accommodate potential future changes. Algorithms were designed in detail and efficiently for all significant requirements. Detailed design diagrams were used appropriately and successfully analysed and defended the use of diagrams. Diagrams consistently used the appropriate abstraction features of the notation to minimize useless redundancy. The diagrams demonstrated a deep

The following Subject Learning Outcomes are addressed in this assessment	
SLO a)	Demonstrate different software engineering principles and techniques.
SLO b)	Author documents required for the software development process e.g.: formal specifications, requirements document, test plan.
SLO c)	Design, develop, maintain and evaluate software systems.