Task 4 – Software Requirement Specification

Task:

Prepare *Software Requirement Specification* document of your proposed project.

Add at least 2 requirement modeling techniques (DFD, decision table, swimlane etc.) discussed in the class.

The SRS template is attached for your reference.

You may just give reference to your previous documents (vision and scope or use case) instead of copying the content.

The detailed description of each section is available in lectures and other reference material for your guidance.

Note: There will be only one submission per group.

Software Requirements Specification
for
<Project>

Version 1.0 approved

Prepared by <author>

<organization>

<date created>

Table of Contents

Revision History

Name	Date	Reason For Changes	Version

1. Introduction

<The introduction presents an overview to help the reader understand how the SRS is organized and how to use it.>

1.1. Purpose

<Identify the product whose software requirements are specified in this document, including the revision or release number. Describe the different types of reader that the document is intended for, such as developers, project managers, marketing staff, users, testers, and documentation writers.>

1.2. Document Conventions

<Describe any standards or typographical conventions used, including the meaning of specific text styles, highlighting, or notations. If you are manually labeling unique requirement identifiers, you might specify the format here for anyone who needs to add one later.>

1.3. Project Scope

<Provide a short description of the software being specified and its purpose. Relate the software to user or corporate goals and to business objectives and strategies. If a separate vision and scope or similar document is available, refer to it rather than duplicating its contents here. An SRS that specifies an incremental release of an evolving product should contain its own scope statement as a subset of the long-term strategic product vision. You might provide a high-level summary of the major features the release contains or the significant functions that it performs.>

1.4. References

<List any documents or other resources to which this SRS refers. Include hyperlinks to them if they are in a persistent location. These might include user interface style guides, contracts, standards, system requirements specifications, interface specifications, or the SRS for a related product. Provide enough information so that the reader can access each reference, including its title, author, version number, date, and source, storage location, or URL.>

2. Overall Description

<This section presents a high-level overview of the product and the environment in which it will be used, the anticipated users, and known constraints, assumptions, and dependencies.>

2.1. Product Perspective

<Describe the product's context and origin. Is it the next member of a growing product line, the next version of a mature system, a replacement for an existing application, or an entirely new product? If this SRS defines a component of a larger system, state how this software relates to the overall system and identify major interfaces between the two. Consider including visual

models such as a context diagram or ecosystem map to show the product's relationship to other systems.>

2.2. User Classes and Characteristics

<Identify the various user classes that you anticipate will use this product and describe their pertinent characteristics. Some requirements might pertain only to certain user classes. Identify the favored user classes. User classes represent a subset of the stakeholders described in the vision and scope document. User class descriptions are a reusable resource. If available, you can incorporate user class descriptions by simply pointing to them in a master user class catalog instead of duplicating information here.>

2.3. Operating Environment

<Describe the environment in which the software will operate, including the hardware platform; operating systems and versions; geographical locations of users, servers, and databases; and organizations that host the related databases, servers, and websites. List any other software components or applications with which the system must peacefully coexist. If extensive technical infrastructure work needs to be performed in conjunction with developing the new system, consider creating a separate infrastructure requirements specification to detail that work.>

2.4. Design and Implementation Constraints

<Describe any factors that will limit the options available to the developers. These might include: corporate or regulatory policies; hardware limitations (timing or memory requirements); interfaces to other applications; specific technologies, tools, and databases to be used; programming language requirements or restrictions.>

2.5. Assumptions and Dependencies

<List any assumed factors (as opposed to known facts) that could affect the requirements stated in the SRS. These could include third-party or commercial components that you plan to use, reuse expectations, issues around the development or operating environment, or constraints. The project could be affected if these assumptions are incorrect, are not shared, or change. Also identify any dependencies the project has on external factors outside its control.>

3. System Features

<This template illustrates organizing the functional requirements for the product by system features, the major services provided by the product. You may prefer to organize this section by use case, mode of operation, user class, object class, functional hierarchy, stimulus, response, or combinations of these, whatever makes the most logical sense for your product.>

3.1. System Feature 1

<Don't really say "System Feature 1." State the feature name in just a few words.>

3.1.1 Description

<Provide a short description of the feature and indicate whether it is of High, Medium, or Low priority.>

3.1.2 Stimulus/Response Sequences

<List the sequences of user actions and system responses that stimulate the behavior defined for this feature. These will correspond to the dialog elements associated with use cases.>

3.1.3 Functional Requirements

<Itemize the specific functional requirements associated with this feature. These are the software capabilities that must be implemented for the user to carry out the feature's services or to perform a use case. Describe how the product should respond to anticipated error conditions. Use "TBD" as a placeholder to indicate when necessary information is not yet available.>

3.2. System Feature 2 (and so on)

4. Data Requirements

<This section describes various aspects of the data that the system will consume as inputs, process in some fashion, or create as outputs.>

4.1. Logical Data Model

<A data model is a visual representation of the data objects and collections the system will process and the relationships between them. Include a data model for the business operations being addressed by the system, or a logical representation for the data that the system itself will manipulate. Data models are most commonly created as an entity-relationship diagram.>

4.2. Data Dictionary

<The data dictionary defines the composition of data structures and the meaning, data type, length, format, and allowed values for the data elements that make up those structures. In many cases, you're better off storing the data dictionary as a separate artifact, rather than embedding it in the middle of an SRS. That also increases its reusability potential in other projects.>

4.3. Reports

<If your application will generate any reports, identify them here and describe their characteristics. If a report must conform to a specific predefined layout you can specify that here as a constraint, perhaps with an example. Otherwise, focus on the logical descriptions of the report content, sort sequence, totaling levels, and so forth, deferring the detailed report layout to the design stage.>

4.4. Data Acquisition, Integrity, Retention, and Disposal

<If relevant, describe how data is acquired and maintained. State any requirements regarding the need to protect the integrity of the system's data. Identify any specific techniques that are necessary, such as backups, checkpointing, mirroring, or data accuracy verification. State policies the system must enforce for either retaining or disposing of data, including temporary data, metadata, residual data (such as deleted records), cached data, local copies, archives, and interim backups.>

5. External Interface Requirements

<This section provides information to ensure that the system will communicate properly with users and with external hardware or software elements.>

5.1. User Interfaces

<Describe the logical characteristics of each interface between the software product and the users. This may include sample screen images, any GUI standards or product family style guides that are to be followed, screen layout constraints, standard buttons and functions (e.g., help) that will appear on every screen, keyboard shortcuts, error message display standards, and so on. Define the software components for which a user interface is needed. Details of the user interface design should be documented in a separate user interface specification.>

5.2. Software Interfaces

<Describe the connections between this product and other software components (identified by name and version), including other applications, databases, operating systems, tools, libraries, websites, and integrated commercial components. State the purpose, formats, and contents of the messages, data, and control values exchanged between the software components. Specify the mappings of input and output data between the systems and any translations that need to be made for the data to get from one system to the other. Describe the services needed by or from external software components and the nature of the intercomponent communications. Identify data that will be exchanged between or shared across software components. Specify nonfunctional requirements affecting the interface, such as service levels for responses times and frequencies, or security controls and restrictions.>

5.3. Hardware Interfaces

<Describe the characteristics of each interface between the software and hardware (if any) components of the system. This description might include the supported device types, the data and control interactions between the software and the hardware, and the communication protocols to be used. List the inputs and outputs, their formats, their valid values or ranges, and any timing issues developers need to be aware of. If this information is extensive, consider creating a separate interface specification document.>

5.4. Communications Interfaces

<State the requirements for any communication functions the product will use, including e-mail, Web browser, network protocols, and electronic forms. Define any pertinent message

formatting. Specify communication security or encryption issues, data transfer rates, handshaking, and synchronization mechanisms. State any constraints around these interfaces, such as whether e-mail attachments are acceptable or not.>

6. Quality Attributes

6.1. Usability

<Specify any requirements regarding characteristics that will make the software appear to be "user-friendly." Usability encompasses ease of use, ease of learning; memorability; error avoidance, handling, and recovery; efficiency of interactions; accessibility; and ergonomics. Sometimes these can conflict with each other, as with ease of use over ease of learning. Indicate any user interface design standards or guidelines to which the application must conform.>

6.2. Performance

<State specific performance requirements for various system operations. If different functional requirements or features have different performance requirements, it's appropriate to specify those performance goals right with the corresponding functional requirements, rather than collecting them in this section.>

6.3. Security

<Specify any requirements regarding security or privacy issues that restrict access to or use of the product. These could refer to physical, data, or software security. Security requirements often originate in business rules, so identify any security or privacy policies or regulations to which the product must conform. If these are documented in a business rules repository, just refer to them.>

6.4. Safety

<Specify requirements that are concerned with possible loss, damage, or harm that could result from use of the product. Define any safeguards or actions that must be taken, as well as potentially dangerous actions that must be prevented. Identify any safety certifications, policies, or regulations to which the product must conform.>

6.5. [Others as relevant]

<Create a separate section in the SRS for each additional product quality attribute to describe characteristics that will be important to either customers or developers. Possibilities include availability, efficiency, installability, integrity, interoperability, modifiability, portability, reliability, reusability, robustness, scalability, and verifiability. Write these to be specific, quantitative, and verifiable. Clarify the relative priorities for various attributes, such as security over performance.>

7. Internationalization and Localization Requirements

<Internationalization and localization requirements ensure that the product will be suitable for use in nations, cultures, and geographic locations other than those in which it was created. Such requirements might address differences in: currency; formatting of dates, numbers, addresses, and telephone numbers; language, including national spelling conventions within the same language (such as American versus British English), symbols used, and character sets; given name and family name order; time zones; international regulations and laws; cultural and political issues; paper sizes used; weights and measures; electrical voltages and plug shapes; and many others.>

8. Other Requirements

<Examples are: legal, regulatory or financial compliance, and standards requirements; requirements for product installation, configuration, startup, and shutdown; and logging, monitoring and audit trail requirements. Instead of just combining these all under "Other," add any new sections to the template that are pertinent to your project. Omit this section if all your requirements are accommodated in other sections. >

Appendix A: Glossary

<Define any specialized terms that a reader needs to know to understand the SRS, including acronyms and abbreviations. Spell out each acronym and provide its definition. Consider building a reusable enterprise-level glossary that spans multiple projects and incorporating by reference any terms that pertain to this project.>

Appendix B: Analysis Models

<This optional section includes or points to pertinent analysis models such as data flow diagrams, feature trees, state-transition diagrams, or entity-relationship diagrams. You might prefer to insert certain models into the relevant sections of the specification instead of collecting them at the end.>