Software Requirements Specification (SRS) & Project Management Plan (PMP)

for

<ProjectName>

Version 1.0 approved

Prepared by
<author>
<author>
<author>
<author>
<author>

<Australian Institute of Higher Education>

<date created>

Table of Contents

Revision History

Name	Date	Reason for changes	Version

1. Introduction

1.1 Purpose

<Identify the product whose software requirements are specified in this document, including the revision or release number. Describe the scope of the product that is covered by this SRS and PMP, particularly if this SRS describes only part of the system or a single subsystem.>

1.2 Project Overview

<A simple overview of the project should be described in this section.>

1.3 Intended Audience and Reading Suggestions

<Describe the different types of reader that the document is intended for, such as developers, project managers, marketing staff, users, testers, and documentation writers. Describe what the rest of this SRS contains and how it is organized. Suggest a sequence for reading the document, beginning with the overview sections and proceeding through the sections that are most pertinent to each reader type.>

1.4 Project Deliverables

<Describe project deliverables. Example: In Assignment 2: SRS and PMP and SDD in Assignment 2. Also, include phases, deliverables and due dates. Click here to see example.>

1.5 References

<List any other documents or Web addresses to which this SRS refers, if any. These may include user interface style guides, contracts, standards, system requirements specifications, use case documents, or a vision and scope document. Provide enough information so that the reader could access a copy of each reference, including title, author, version number, date, and source or location.>

1.6 Definitions and Acronyms

<Provide definitions and acronyms if necessary. Example: PMP – Project Management Plan; SRS – Software Requirements Specification; SDD – Software Design Document>

2. Project Organisation

2.1 Process Model: Agile

<Discuss agile in just a few words. Click here to see example.>

2.1.1 Project Planning

<Discuss project plan>

2.1.2 Requirement Analysis

<Use SCRUM features such as Product Backlog Items written in User Story form, Sprint Tasks (this section should be in detail as it is the SRS. Each item and sub item must be indexed so that it can be referenced again in Assignment 3 (SDD) while Software Design Documentation).>

2.1.3 Software Design

< Architecture of web application, GUI design, process & workflow design, database design (mention this will be delivered as a milestone in Assessment 3 (SDD), so do not provide in-depth details here).>

2.1.4 Analysis Review

<Analysis Review>

2.1.5 Prototype: Front end prototype

<Create front end prototypes of your system you are after. You can use Mockflow (free), Mogups (trial), Balsamiq (trial)>

2.2 Organisational Structure

2.2.1 Project Manager and Tasks

<Project Manager and Tasks. Click here to see example.>

2.3 Organisational Boundaries and Interfaces

2.3.1 GitHub Communication

<Overall Description of how GitHub will be used to control versions and as a part of collaboration during development phase>

2.3.2 Meeting Times

<Discuss meeting times here>

2.4 Project Responsibilities

<Project Management Plan (PMP) Starts>

2.4.1 Project Management

<Discuss how the project will be managed. Click here to see example.>

2.4.2 Project Sponsor

<In case of real-world projects>

2.4.3 Liaison Manager

<In case of real-world projects>

2.4.4 Document Editor

<Discuss who will be editing documentations and how>

3. Managerial Process

3.1 Management Objectives and Priorities

<Discuss Management Objectives and Priorities. Click here to see example.>

3.2 Assumptions, Dependencies and Constraints

3.2.1 Assumptions

<Discuss assumptions. Click here to see example.>

3.2.2 Dependencies

<Discuss dependencies. Click here to see example.>

3.2.3 Constraints

<Discuss Constraints. Click here to see example.>

3.3 Risk Management

<Details of individual team risk management assessments should be provided. Click here to see example.>

3.3.1 Analysts Perspective

<Risk Management assessment from Analyst perspective>

3.3.2 Design perspective: Front end, process, database

<Risk Management assessment from design perspective>

3.3.3 Project Management Perspective

<Risk Management assessment from project management perspective>

3.4 Monitoring and Controlling Mechanism

<*Click here to see example.*>

4. Technical Process

4.1 Methods, Tools and Techniques

<*Click here to see example.*>

4.2 Software Documentation

<*Click here to see example.*>

4.3 Project Support Functions

<*Click here to see example.*>

4.4 Work Elements, Schedule

<Do not include budget. Click here to see example.>

5. Overall Description

<Software Requirements Specification (SRS) Starts. Click here to see example SRS>

5.1 Product Perspective

<Describe the context and origin of the product being specified in this SRS. For example, state whether this product is a follow-on member of a product family, a replacement for certain existing systems, or a new, self-contained product. If the SRS defines a component of a larger system, relate the requirements of the larger system to the functionality of this software and identify interfaces between the two. A simple diagram that shows the major components of the overall system, subsystem interconnections, and external interfaces can be helpful.>

5.2 Product Functions

<Summarize the major functions the product must perform or must let the user perform. Details will be provided in Section 3, so only a high-level summary (such as a bullet list) is needed here. Organize the functions to make them understandable to any reader of the SRS. A picture of the major groups of related requirements and how they relate, such as a top-level data flow diagram or object class diagram, is often effective.>

5.3 User Classes and Characteristics

<Identify the various user classes that you anticipate will use this product. User classes may be differentiated based on frequency of use, subset of product functions used, technical expertise, security or privilege levels, educational level, or experience. Describe the pertinent characteristics of each user class. Certain requirements may pertain only to certain user classes. Distinguish the most important user classes for this product from those who are less important to satisfy.>

5.4 Operating Environment

<Describe the environment in which the software will operate, including the hardware platform, operating system and versions, and any other software components or applications with which it must peacefully coexist.>

5.5 Design and Implementation Constraints

<Describe any items or issues that will limit the options available to the developers. These might include: corporate or regulatory policies; hardware limitations (timing requirements, memory requirements); interfaces to other applications; specific technologies, tools, and databases to be used; parallel operations; language requirements; communications protocols; security considerations; design conventions or programming standards (for example, if the customer's organization will be responsible for maintaining the delivered software).>

5.6 User Documentation

<List the user documentation components (such as user manuals, on-line help, and tutorials) that will be delivered along with the software. Identify any known user documentation delivery formats or standards.>

5.7 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, 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, such as software components that you intend to reuse from another project, unless they are already documented elsewhere (for example, in the vision and scope document or the project plan).>

6. External Interface Requirements

6.1 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.>

6.2 Hardware Interfaces

<Describe the logical and physical characteristics of each interface between the software product and the hardware components of the system. This may include the supported device types, the nature of the data and control interactions between the software and the hardware, and communication protocols to be used.>

6.3 Software Interfaces

<Describe the connections between this product and other specific software components (name and version), including databases, operating systems, tools, libraries, and integrated commercial components. Identify the data items or messages coming into the system and going out and describe the purpose of each. Describe the services needed and the nature of communications. Refer to documents that describe detailed application programming interface protocols. Identify data that will be shared across software components. If the data sharing mechanism must be implemented in a specific way (for example, use of a global data area in a multitasking operating system), specify this as an implementation constraint.>

6.4 Communications Interfaces

<Describe the requirements associated with any communications functions required by this product, including e-mail, web browser, network server communications protocols, electronic forms, and so on. Define any pertinent message formatting. Identify any communication standards that will be used, such as FTP or HTTP. Specify any communication security or encryption issues, data transfer rates, and synchronization mechanisms.>

7. 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, or combinations of these, whatever makes the most logical sense for your product.>

7.1 System Feature 1

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

4.1.1 Description and Priority

<Provide a short description of the feature and indicate whether it is of High, Medium, or Low priority. You could also include specific priority component ratings, such as benefit, penalty, cost, and risk (each rated on a relative scale from a low of 1 to a high of 9).>

4.1.2 Stimulus/Response Sequences

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

4.1.3 Functional Requirements

<Itemize the detailed functional requirements associated with this feature. These are the software capabilities that must be present in order for the user to carry out the services provided by the feature, or to execute the use case. Include how the product should respond to anticipated error conditions or invalid inputs. Requirements should be concise, complete, unambiguous, verifiable, and

necessary. Use "TBD" as a placeholder to indicate when necessary information is not yet available.>

<Each requirement should be uniquely identified with a sequence number or a meaningful tag of some kind.>

REQ-1:
REQ-2:

7.2 System Feature 2 (and so on)

8. Other Non-functional Requirements

8.1 Performance Requirements

<If there are performance requirements for the product under various circumstances, state them here and explain their rationale, to help the developers understand the intent and make suitable design choices. Specify the timing relationships for real time systems. Make such requirements as specific as possible. You may need to state performance requirements for individual functional requirements or features.>

8.2 Safety Requirements

<Specify those requirements that are concerned with possible loss, damage, or harm that could result from the use of the product. Define any safeguards or actions that must be taken, as well as actions that must be prevented. Refer to any external policies or regulations that state safety issues that affect the product's design or use. Define any safety certifications that must be satisfied.>

8.3 Security Requirements

<Specify any requirements regarding security or privacy issues surrounding use of the product or protection of the data used or created by the product. Define any user identity authentication requirements. Refer to any external policies or regulations containing security issues that affect the product. Define any security or privacy certifications that must be satisfied.>

8.4 Software Quality Attributes

<Specify any additional quality characteristics for the product that will be important to either the customers or the developers. Some to consider are: adaptability, availability, correctness, flexibility, interoperability, maintainability, portability, reliability, reusability, robustness, testability, and usability. Write these to be specific, quantitative, and verifiable when possible. At the least, clarify the relative preferences for various attributes, such as ease of use over ease of learning.>

8.5 Business Rules

<List any operating principles about the product, such as which individuals or roles can perform which functions under specific circumstances. These are not functional requirements in themselves, but they may imply certain functional requirements to enforce the rules.>

9. Other Requirements

<Define any other requirements not covered elsewhere in the SRS. This might include database requirements, internationalization requirements, legal requirements, reuse objectives for the project, and so on. Add any new sections that are pertinent to the project.>

Appendix A: Glossary

<Define all the terms necessary to properly interpret the SRS, including acronyms and abbreviations. You may wish to build a separate glossary that spans multiple projects or the entire organization, and just include terms specific to a single project in each SRS.>

Appendix B: Analysis Models

<Optionally, include any pertinent analysis models, such as data flow diagrams, class diagrams, state-transition diagrams, or entity-relationship diagrams.>

Appendix C: To Be Determined List

<Collect a numbered list of the TBD (to be determined) references that remain in the SRS so they can be tracked to closure.>