

Department of Information and Communication Technology

Faculty of Technology

University of Ruhuna

**Employee Management System**

**Software Requirement Specification**

Group Project (ICT3183)

Project ID: 07

Submitted by:

1. Index no. – (Name)

Submitted to:

(Supervisor’s signature)

…………………………..

Name of the supervisor: Ms.Malsha Prabuddhi

Date of submission

Table of Contents

Table of Contents i

Revision History i

1. Introduction 1

1.1 Purpose 1

1.2 Document Conventions 1

1.3 Intended Audience and Reading Suggestions 1

1.4 Product Scope 1

1.5 References 1

2. Overall Description 2

2.1 Product Perspective 2

2.2 Product Functions 2

2.3 User Classes and Characteristics 2

2.4 Operating Environment 2

2.5 Design and Implementation Constraints 2

2.6 Project Documentation 2

2.7 User Documentation 3

2.8 Assumptions and Dependencies 3

3. External Interface Requirements 4

3.1 User Interfaces 4

3.2 Hardware Interfaces 4

3.3 Software Interfaces 4

3.4 Communications Interfaces 4

4. System Features 5

4.1 System Feature 1 5

4.2 System Feature 2 5

4.3 System Feature 3 (and so on) 5

5. Other Nonfunctional Requirements 6

5.1 Performance Requirements 6

5.2 Safety Requirements 6

5.3 Security Requirements 6

5.4 Software Quality Attributes 6

5.5 Business Rules 6

6. Other Requirements 7

Appendix A: Glossary 8

Appendix B: Analysis Models 9

Appendix C: To Be Determined List 10

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason For Changes** | **Version** |
|  |  |  |  |
|  |  |  |  |

# Introduction

## Purpose

The Employee Management System (EMS) aims to streamline and automate various aspects of human resource management within an organization. This system is designed to enhance efficiency, accuracy, and transparency in handling employee-related information and processes. By providing a centralized platform, it enables HR professionals and organizational stakeholders to manage, monitor, and optimize workforce activities effectively.

Version 1.1

## Document Conventions

1. Document Title: The title of the document should be prominently displayed at the beginning, clearly indicating that it is an SRS.
2. Version Control: Include a version number and date to track the document's revisions. This helps readers identify the most recent version and understand changes made over time.
3. Table of Contents: Provide a table of contents to outline the structure of the document and help users navigate to specific sections.
4. Headers and Footers: Consistently use headers and footers to display the document title, section names, and page numbers. This maintains a professional appearance and aids in document navigation.
5. Formatting Styles: Use consistent formatting styles for text, such as font type, size, and color. Bold, italic, and underline can be used consistently to highlight important points.
6. Section Numbering: Number sections and subsections to create a logical hierarchy. This makes it easier for readers to locate specific information and understand the relationships between different sections.
7. Bullet Points and Numbered Lists: Use bullet points or numbered lists for clarity when presenting lists or sequences of information.
8. Acronyms and Abbreviations: Include a list of acronyms and abbreviations with their definitions to assist readers in understanding technical terms.
9. References: If external documents or sources are referenced, provide a bibliography or list of references for readers to access additional information.
10. Graphical Elements: Use consistent conventions for graphical elements, such as charts, diagrams, and tables. Include titles, labels, and legends to enhance comprehension.
11. Naming Conventions: If the document refers to specific entities (e.g., software components, modules), establish and adhere to naming conventions to maintain consistency.
12. Document Review and Approval: Include a section indicating who has reviewed and approved the document. This ensures accountability and provides context regarding the document's reliability.

1. Change History: Maintain a change history section that details revisions, including the version number, date, author, and a brief description of changes made. This helps users understand how the document has evolved.

## Intended Audience and Reading Suggestions

### Intended Audience:

* Developers: Individuals responsible for designing, coding, and testing the software.
* Project Managers: Individuals overseeing the software development project, responsible for planning, scheduling, and resource allocation.
* Testers: QA professionals responsible for testing the software to ensure quality and functionality.
* Documentation Writers: Individuals responsible for creating user manuals, help guides, and other documentation related to the software.
* Users: End-users who will interact with the software, including employees, administrators, or customers.
* Marketing Staff: Individuals involved in promoting and marketing the software to potential users or clients.

### Description of Document Contents and Organization

The SRS document provides a detailed description of the software requirements, outlining its functionality, features, constraints, and user interactions. It is organized into sections that cover various aspects of the software, including

1. Introduction: Provides an overview of the software, its purpose, scope, and intended audience.

2. General Description: Describes the context of the software, including its features, constraints, and assumptions.

3. Specific Requirements: Details the functional and non-functional requirements of the software, including user interactions, system behavior, and performance criteria.

4. External Interface Requirements: Describes the interfaces between the software and external systems, including hardware, software, and communication protocols.

5. System Features: Lists and describes the specific features and functionalities of the software.

6. Other Requirements: Covers additional requirements such as documentation, training, and support.

7. Appendices: Includes supplementary information such as glossary, references, and supporting documentation.

Reading Sequence Suggestions:

1. Begin with the Overview Sections:

- Start by reading the Introduction and General Description sections to understand the purpose, scope, and context of the software.

- Proceed to the System Features section to get an overview of the specific functionalities and features of the software.

2. Developer and Project Manager Focus:

- Developers and project managers should focus on the Specific Requirements section to understand the detailed functional and non-functional requirements of the software.

- They should also review the External Interface Requirements section to understand the integration points with external systems.

3. Tester Focus:

- Testers should pay close attention to the Specific Requirements section to understand the expected system behavior and performance criteria for testing purposes.

4. User Focus:

- Users should refer to the System Features section to understand the available functionalities and how they can interact with the software.

- They may also find the General Description section useful for understanding the context and constraints of the software.

5. Documentation Writer Focus:

- Documentation writers should review the Appendices section for supplementary information such as the glossary and references to support their documentation efforts.

By following this reading sequence, each stakeholder can focus on the sections most relevant to their role and gain a comprehensive understanding of the software requirements outlined in the SRS document.

## Product Scope

<Provide a short description of the software being specified and its purpose, including relevant benefits, objectives, and goals. Relate the software to corporate goals or business strategies. If a separate vision and scope document is available, refer to it rather than duplicating its contents here.>

## References

<List any other documents or Web addresses to which this SRS refers. 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.>

# Overall Description

## 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.>

## Product Functions

<Summarize the major functions the product must perform or must let the user perform. Details will be provided in Section 4, 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.>

## 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.>

## 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.>

## 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).>

## Project Documentation

<Write the project documentation components (such as Introduction) that will be delivered along with the software. Identify any known project documentation delivery formats or standards.>

## 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.>

## 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).>

# External Interface Requirements

## 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.>

## 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.>

## 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.>

## 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.>

# 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.>

## System Feature 1

## System Feature 2

## System Feature 3 (and so on)

**In your case you can write this whole section (Chapter 4 and its sub sections) based on a Use Case Diagrams, Use Case Scenarios, Initial Class Diagram, and Activity Diagram. (Initial Class Diagram and Activity Diagrams are optional) If you have a fairly large number of use cases you can make use of packages to group the use cases into multiple diagrams. Then for each package draw a separate use case diagram**

**Use Alistair Cockburn’s template from the unit Software Engineering II to document your use cases**.

# Other Nonfunctional Requirements

## 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.>

## 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.>

## 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.>

## 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.>

## 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.>

# 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.>