Software Requirements Specification (SRS) Template

The document in this file is an annotated outline for specifying software requirements, adapted from the IEEE Guide to Software Requirements Specifications.

(Remove this page from your submission)

New York University Abu Dhabi
CS-UH 2012: Software Engineering

(Group 5)

Software Requirements Specification for < *Dietade* >

Version: (n) **Date: (03/2/2023)**

Table of Contents

1. Introduction

1.1 Purpose

The purpose of this Software Requirements Specification (SRS) is to define the requirements and features of the Dietaide platform, which is a smart diet-building platform designed for hospital health professionals and patients. The SRS outlines the functional and non-functional requirements, user interfaces, system interfaces, performance requirements, and other specifications needed for the successful development and implementation of the software.

The intended audience of this SRS includes the development team, project stakeholders, and quality assurance team responsible for building and testing the software. This document serves as a reference guide for all parties involved in the software development lifecycle, ensuring that the final product meets the requirements of the hospital health professionals and patients who will use it.

By defining the scope and objectives of the Dietaide platform, this SRS will enable the development team to deliver a high-quality software solution that improves the efficiency and accuracy of diet planning and communication between health professionals and patients.

1.2 Scope

(1) The software product to be produced is Dietaide, a smart diet-building platform designed for hospital health professionals and patients.

(2) In general, Dietade *will*:

- Allow health professionals to provide and access appropriate diets for patients based on their medical diagnosis.
- Enable users to create customized diet plans and facilitate communication between health professionals and patients regarding dietary needs.

However, Dietade *will not*:

- Provide medical advice, diagnosis, or treatment recommendations.

(3) Dietade's Benefits: Improve patient outcomes by ensuring that patients receive tailored dietary recommendations that are aligned with their medical diagnosis

Dietade's Objectives: Aim to streamline the diet planning process, reduce manual effort, and improve accuracy.

Dietade's Goals: By providing an intuitive and efficient platform for diet planning and communication, Dietaide will enable health professionals to focus on delivering better care to patients.

1.3 Definitions, Acronyms, and Abbreviations.

Provide the definitions of all terms, acronyms, and abbreviations required to properly interpret the SRS. This information may be provided by reference to one or more appendices in the SRS or by reference to documents. This information may be provided by reference to an Appendix.

1.4 References

In this subsection:

(1) Provide a complete list of all documents referenced elsewhere in the SRS
(2) Identify each document by title, report number (if applicable), date, and publishing organization
(3) Specify the sources from which the references can be obtained.

This information can be provided by reference to an appendix or to another document. If your application uses specific protocols or RFC's, then reference them here so designers know where to find them.

1. IP/TCP Protocol Suite:
   Title: TCP/IP Illustrated, Volume 1: The Protocols Author: W. Richard Stevens
   Date: 1994
   Publishing Organization: Addison-Wesley Professional
   Source: https://www.amazon.com/TCP-Illustrated-Protocols-Addison-Wesley-Professional/dp/0201633469

2. HTTP/HTTPS Protocol:
   Title: Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing
   Report Number: RFC 7230
   Date: June 2014
   Publishing Organization: Internet Engineering Task Force (IETF)
   Source: https://tools.ietf.org/html/rfc7230

3. Amazon Web Services (AWS):
   Title: AWS Documentation
   Publishing Organization: Amazon Web Services, Inc.
   Source: https://aws.amazon.com/documentation/

4. Microsoft SQL Server:
   Title: Microsoft SQL Server Documentation
   Publishing Organization: Microsoft Corporation
   Source: https://docs.microsoft.com/en-us/sql/sql-server/?view=sql-server-ver15

5. NoSQL:
   Title: NoSQL Databases Explained
   Publishing Organization: IBM
   Source: https://www.ibm.com/cloud/learn/nosql-databases-explained

6. MongoDB:
   Title: MongoDB Documentation

7. Publishing Organization: MongoDB, Inc.
   Source: https://docs.mongodb.com/

8. Flask:
   Title: Flask Documentation
   Publishing Organization: Flask Documentation Project
   Source: https://flask.palletsprojects.com/en/2.1.x/

9. UAE Ministry of Health and Prevention website (MOHAP):
   Title: MOHAP
   Publishing Organization: Government of the United Arab Emirates
   Source: https://www.mohap.gov.ae

1.5 Overview

The requirements and features of the Dietaide platform are thoroughly described in this Software Requirements Specification (SRS) paper. The SRS details the performance requirements, user interfaces, system interfaces, functional and non-functional requirements, and other specifications required for the software's effective development and implementation.

The SRS is divided into a number of sections that offer in-depth details about the undertaking. Section 2, which provides comprehensive details about the functional requirements and features of the Dietaide platform, will be of particular interest to customers and prospective users of the software. This part describes the functionality and intended use of the software, as well as any constraints or limitations.

The material in Section 3 about the technical requirements and non-functional requirements of the software will be of most interest to developers and other technical stakeholders. The architecture, design, and implementation of the software are all covered in this part, along with any performance or security needs.

The SRS's other sections contain more details, like a glossary of terms, references, and appendices. Overall, the SRS acts as a guidebook for everyone involved in the software development lifecycle, ensuring that the finished product satisfies the needs of the hospital health workers and patients who will use it.

2. The Overall Description

Dietaide is a web application created to give medical staff members and patients an easy method to obtain and administer the right diets based on their diagnoses. By offering patients pertinent and individualized food advice depending on their medical circumstances, the platform is designed with the goal of improving patient outcomes.

The platform is made to help hospital health professionals overcome the difficulties they have when planning and overseeing the right diets for their patients. Conventional food management techniques can be time-consuming and error-prone, which results in less than ideal patient outcomes. Dietaide seeks to improve accuracy and efficiency while lessening the workload on hospital workers by streamlining the dietary management process.

Dietaide includes a variety of features to fulfill the demands of patients and medical professionals in order to accomplish these goals. These features include direct patient-doctor connection via chat channels, patient login and access to pertinent diagnoses, automatic diet advice generation, doctor review and approval of generated diets, and automatic diet recommendation generation.

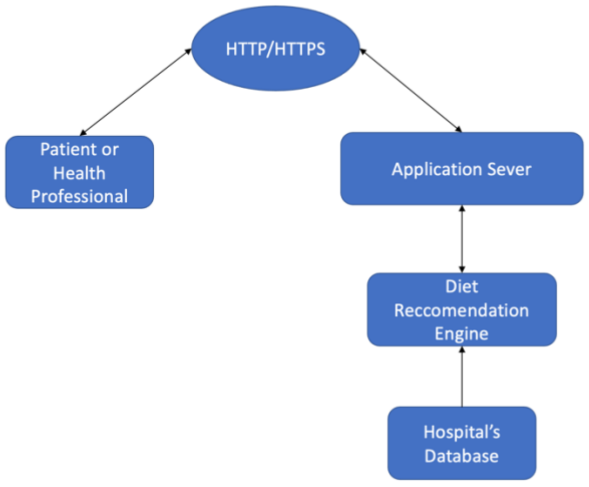
The platform has an intuitive user interface and is designed with usability and accessibility in mind. Overall, Dietaide is an excellent tool created to assist hospital health professionals and patients in better managing nutritional needs, enhancing patient outcomes, and lessening the workload on hospital staff.

2.1 Product Perspective

Dietaide is a standalone web application built specifically to function in a hospital setting. Even though it is not a part of a bigger system, it might communicate with other hospital systems, like the electronic medical record system, to access patient data and medical records.

Although there are a number of diet and nutrition-related web applications on the market, Dietaide is special in that it was created with medical staff and patients in mind. With a focus on streamlining and streamlining the process as much as possible, it offers an intuitive platform for accessing and offering recommended meals based on diagnosis.

Dietaide operates in accordance with all applicable hospital policies and regulations, including those pertaining to data security and privacy. The platform is made to effortlessly integrate with the infrastructure and workflows already in place at the hospital, causing no disruption or alteration to current procedures.



Block Diagram of the major components of the larger system, interconnections, and external interfaces.

2.1.1 System Interfaces

As the Dietaide platform is a self-contained system, there are no external systems that it needs to interact with. However, the platform does have interfaces for different types of users, such as health professionals and patients, to interact with the system. These interfaces include:

1. User interface: This interface allows users to interact with the system through a graphical user interface. Users can log in to the system, select relevant diagnoses, view generated diets, and communicate with health professionals.

2. Database interface: The system database stores diagnosis-dietary correlations and patient profiles. The database interface allows the system to access and manipulate this data as needed to generate appropriate diets for patients.
3. Diet recommendation engine interface: The diet recommendation engine uses the diagnosis-dietary correlations stored in the database to generate appropriate diets for patients based on their selected diagnoses and any additional conditions/severity. The engine interface allows the system to interact with this engine to generate and modify diet recommendations as needed.

Overall, these interfaces allow the different components of the Dietaide platform to interact with each other to provide an intuitive platform for health professionals and patients to access appropriate diets based on diagnosis. Assuming the hospital's database is managed with SQL, the API used to interact with the database will be SQLAPI++ if C++ is used or SQLite3 if Python3 is used. On the other hand, it is assumed that the database is managed with NoSQL if not SQL; in that case, the API is MongoDB.

2.1.2 Interfaces

(1) The logical characteristics of each interface between the Dietade software product and its users include:

- Graphical User Interface (GUI) for patients and health professionals to interact with the system. The GUI should be intuitive and easy to navigate with clear instructions and feedback messages.

- A login interface for patients to access their hospital profile and upload their selected diagnoses relevant to their current dietary needs.

- An interface for health professionals to access patients' profiles and medical records within the hospital, view their selected diagnoses, generate diet recommendations, and make adjustments or other recommendations.

- A chat interface for direct patient-doctor communication on diet recommendations.

(2) Aspects of optimizing the interface with the person who must use the system:

- The GUI should be designed to be user-friendly and visually appealing to the target audience, which includes patients and health professionals.

- The system should provide clear instructions and feedback messages to users to help them navigate the system and understand how to use it effectively.

- The chat interface should be designed to facilitate clear and effective communication between patients and health professionals, with features such as real-time messaging, message notifications, and/or message history.

2.1.3 Hardware Interfaces

For Dietade, the software product will have the following interface with the hardware components of the system:

1. Server hardware: The software product will require a server to run on. The server must meet the minimum hardware requirements to run the software and support the expected number of concurrent users.

2. User hardware: The software product will support the use of standard web browsers on desktop computers.
3. Network interface: The software product will communicate with the hospital's internal network and internet to access patient data and provide diet recommendations. The system will use standard protocols such as HTTP and HTTPS to communicate over the network.

4. Data storage hardware: The software product will use a database management system to store patient data and diet recommendations. The database management system must meet the minimum hardware requirements to support the expected number of concurrent users and data storage capacity. The software product will support both SQL and/or NoSQL database management systems.

The software product will optimize its interface with the hardware components of the system by implementing efficient algorithms, minimizing the use of system resources, and following best practices for web development to ensure compatibility hardware and software configurations. The software product will also provide clear and concise error messages to help diagnose any hardware or software compatibility issues that may arise.

2.1.4 Software Interfaces

Dietade will interact with the following software products and interfaces:

1.
    - Name: MongoDB (if NoSQL is used)
    - Community Server Mnemonic: MongoDB
    - Specification Number: N/A
    - Version Number: 4.4.6
    - Source: https://www.mongodb.com/
    - Purpose: MongoDB will be used as the database management system for Dietade. The system will interact with the MongoDB server through the MongoDB API to perform operations such as data insertion, querying and deletion.
    - Interface: The interface between Dietade and MongoDB will be through the MongoDB API. The API will communicate with the

database server using the BSON (Binary JSON) format. The data to be exchanged will be in the form of JSON objects.

2.

- Name: Microsoft SQL Server (if SQL is used)
- Mnemonic: SQL Server
- Specification number: N/A
- Version number: 2019
- Source: Microsoft
- Purpose: The system must use SQL Server as its database component. Communication with the DB is through ODBC connections.
- Interface Definition:
    - The system must provide SQL data table definitions to be provided to the company DBA for setup.
    - The system must connect to SQL Server using ODBC connections.
    - SQL queries must be executed using the appropriate syntax for SQL Server.
    - Data returned from SQL Server must be in a format compatible with the system's data structures.
    - SQL Server errors must be handled appropriately by the system, and appropriate error messages must be displayed to the user.

3.

- Name: Flask
- Mnemonic: N/A
- Specification Number: N/A
- Version Number: 2.0.1
- Source: https://flask.palletsprojects.com/
- Purpose: Flask will be used as the web framework for Dietade. It will handle incoming HTTP requests and route them to the appropriate application functions. It will also be responsible for generating HTTP responses to send back to the client.
- Interface: The interface between Dietade and Flask will be through the Flask API. The API will receive requests in the form of HTTP requests and send responses in the form of HTTP responses. The API will also use the Jinja2 templating engine to generate HTML templates for dynamic web pages.

4.

- Name: AWS S3
- Mnemonic: S3
- Specification Number: N/A
- Version Number: N/A
- Source: https://aws.amazon.com/s3/
- Purpose: AWS S3 will be used to store and manage user-uploaded files for Dietade. This includes profile pictures, meal plans, and other user-generated content.

- Interface: The interface between Dietade and AWS S3 will be through the AWS SDK for Python (Boto3). The SDK will communicate with the S3 server using the REST API over HTTP/HTTPS. The data to be exchanged will be in the form of binary data or multipart/form-data.

2.1.5 Communications Interfaces

1. TCP/IP Protocol: Dietade requires support for TCP/IP protocol for communication between the client application and the server application.

2. HTTP Protocol: Dietade requires support for HTTP protocol for communication between the client application and the server application.

2.1.6 Memory Constraints

The database management system must meet the following minimum memory requirements to support (by assumption) up to 500 concurrent users:

- Primary memory: The system must have a minimum of 16GB of RAM available to allocate to the database management system.
- Secondary memory: The system must have a minimum of 500GB of hard disk space available for storing data files.

  These memory constraints are based on the expected number of concurrent users and the anticipated amount of data storage required. If additional users or data storage is needed in the future, the memory requirements may need to be adjusted accordingly.

2.1.7 Operations

1. Modes of operation:
   - User login and authentication
   - Data entry and editing
   - Search and retrieval of data
   - Generating reports
   - Data analysis

2. Various modes of operations:
   - Dietade will be used in a continuous mode of operation, with users accessing the system throughout the day.
   - The system must be able to handle peak usage periods, which are expected to occur during the morning and lunchtime hours.

3. Periods of interactive and unattended operations:

- Dietade will primarily operate in interactive mode, with users accessing the system through a web-based interface.
- However, some automated background processes will also be running periodically, such as data imports and exports.

4. Data processing support functions:
   - Dietade must be able to import and export data in various formats, including CSV and Excel.
   - The system must also be able to generate reports on user activity, meal plans, and other metrics.

2.1.8 Site Adaptation Requirements

1. The system requires specific data to be provided by the customer, including:
   - User information: The system requires a list of authorized users and their access levels to be provided by the customer.

2. Site-related features that should be modified to adapt the software to a particular installation:
   - The system requires a reliable internet connection with a minimum bandwidth of 1 Mbps to ensure smooth operation.

   - The system should be installed on a server with a minimum of 8 GB RAM and 500 GB hard disk space.

   - The customer must ensure that the server is located in a secure and climate-controlled environment.

   - The system requires daily backups to be taken and stored off-site to ensure data protection.

   - The customer must ensure that adequate training is provided to the staff who will be using the system.

   - The customer must provide access to their existing database server for the integration of the system.

   - The system requires a dedicated domain name or IP address for hosting the application.

2.2 Product Functions

1. User Management:
   - Allow users to create and manage their accounts with login credentials and personal information
   - Provide the ability for users to update their account information and reset their passwords if needed

- Authenticate users and control access to system features based on user roles and permissions

2. <u>Nutrition Tracking:</u>
   - Allow users to input their dietary goals and restrictions
   - Enable users to track their daily food intake and calculate nutritional values based on the provided product information
   - Provide users with visual representations of their progress towards their nutritional goals

3. <u>Meal Planning:</u>
   - Generate meal plans for users based on their dietary goals and restrictions, using existing meal plans provided by the customer as well as new plans generated by the system
   - Allow users to customize meal plans by swapping out meals or adjusting portion sizes
   - Generate grocery lists for users based on their chosen meal plans

5. <u>Customer Support:</u>
   - Provide users with a help center containing resources and frequently asked questions
   - Allow users to submit support requests and track their progress
   - Enable customer support staff to view and manage support requests and provide assistance to users
6. <u>System Administration:</u>
   - Allow administrators to manage user accounts and access permissions
   - Provide tools for managing product information, including adding new products and updating nutritional values
   - Enable administrators to view system usage data and generate reports on user activity and system performance.

2.3 User Characteristics

The intended users of Dietade are individuals who are seeking to improve their health through better nutrition. As such, the general characteristics of the users are as follows:

1. <u>Educational level</u>: There are no specific educational requirements for using Dietade. However, users are expected to have a basic understanding of nutrition and healthy eating habits.

2. <u>Experience:</u> Users of Dietade are likely to have a range of experience with nutrition and dieting. Some may be experienced in tracking their food intake and understanding nutritional information, while others may be relatively new to the concept.

3. <u>Technical expertise</u>: Dietade is designed to be user-friendly and accessible to individuals with varying levels of technical expertise. However, users are

expected to have basic computer skills such as the ability to navigate a web-based application and enter data using a keyboard.

The characteristics of the users will impact the design of Dietade in several ways. For example, the system will need to be designed with a user interface that is intuitive and easy to use for individuals with varying levels of technical expertise. Finally, the system *may* need to provide guidance and educational resources to users who are less experienced with nutrition and healthy eating habits.

2.4 Constraints

1. Regulatory policies: The system must comply with relevant food safety regulations and guidelines, such as MOHAP requirements.

2. Hardware limitations: The system must be able to operate on standard desktop and laptop computers with a minimum of 2GB RAM and 250GB hard drive space.

3. Parallel operation: The system must be able to handle a maximum of 500 concurrent users without significant performance degradation.

4. Audit functions: The system must log all user actions and transactions for auditing purposes.

5. Control functions: The system must restrict access to certain features and data based on user roles and permissions.

6. Higher-order language requirements: The system must be coded in Python 3.8 or higher, or C++, or Java.

7. Signal handshake protocols: The system must support the ACK-NACK protocol for data transmission between the server and client applications.

8. Reliability requirements: The system must have a 99.9% uptime guarantee and be able to handle spikes in traffic and usage.

9. Criticality of the application: The system is critical to the customer's business and any downtime or errors could result in significant financial losses.

10. Safety and security considerations: The system must adhere to industry-standard security practices, including secure data transmission and access controls based on user roles and permissions.

2.5 Assumptions and Dependencies

1. The customer will provide accurate and up-to-date nutritional information for all products to be used in the system.

2. The customer will provide necessary access to any third-party applications or systems that the software must interface with.

3. The customer will provide adequate hardware and network infrastructure to support the software.

4. The software will be developed and tested using the specified operating system and programming languages.

5. Any changes to the regulatory policies or standards will be communicated to the development team in a timely manner.

6. The customer will provide necessary training to their employees on the use of the software.

7. The customer will ensure that all data entered into the system is accurate and complete.

8. The software will be deployed and used in a safe and secure manner, in compliance with relevant laws and regulations.

9. The development team will have access to all necessary resources and personnel to complete the project within the given timeframe.
10. The system is dependent on the availability and reliability of the internet connection at the customer's site.

11. It is assumed that the customer's existing meal plans are in a format that can be easily integrated into the system.

12. The system is dependent on the performance and capacity of the hardware and infrastructure provided by the customer, such as servers and databases.

13. It is assumed that the customer will provide adequate training to their staff on how to use the system effectively.

14. The system is dependent on the availability and compatibility of any third-party software or services used in conjunction with it, such as payment processing or API integrations.

15. It is assumed that the customer will comply with any relevant regulations or standards related to the storage and handling of sensitive user data.

16. The system is dependent on the security measures implemented by the customer, such as firewalls and user authentication, to protect against unauthorized access or data breaches.

17. It is assumed that the customer's database system will be able to support the amount and type of data that the system will store and manipulate.

18. It is assumed that the customer's network infrastructure will be able to handle the expected traffic and communication between the system's components.

19. It is assumed that the customer's hardware and operating system configurations will be compatible with the software product and any third-party libraries or tools it relies on.

20. It is assumed that the customer's web server or hosting environment will meet the minimum requirements for the software product to run smoothly and securely.

21. It is assumed that the customer's payment gateway or financial service provider will provide reliable and secure payment processing services that integrate with the system's billing functionality.

22. It is assumed that the customer will be responsible for providing any necessary training or support to their own end-users who will be using the system.

2.6 Apportioning of Requirements.

1. <u>Should-Have Requirements:</u>

These requirements are important but can be postponed until future versions of the system. They provide additional value to the system but are not critical for the initial release.
These requirements include:

- Patients should be able to select specific diagnoses from their hospital record that they find relevant to their current dietary needs, and upload them to the system.
- Doctors and other medical professionals should be able to review generated diets, make adjustments and/or other recommendations as they see fit, and then make approvals.
- Venues such as chat channels should be available for direct patient-doctor communication on diet recommendations.
- Additional features such as tracking and monitoring of patient progress, personalized alerts and reminders, and gamification elements to encourage adherence to prescribed diets.

2. <u>Could-Have Requirements:</u>
   These requirements are desirable but are not necessary for the initial release. They can be postponed until later versions of the system.
   These requirements include:
   - Patients should be able to view all generated diets both before and after approval, attached to their own profile.
   - The system should be able to attach the approved diet recommendations to the patient's profile for future reference.

- The system should be able to track the progress of patients' dietary adherence and provide recommendations accordingly.
- Integration/expansion into other hospitals to ensure completeness of medical records.

3. Specific Requirements

This section contains all the software requirements at a level of detail sufficient to enable designers to design a system to satisfy those requirements, and testers to test that the system satisfies those requirements. Throughout this section, every stated requirement should be externally perceivable by users, operators, or other external systems. These requirements should include at a minimum a description of every input (stimulus) into the system, every output (response) from the system and all functions performed by the system in response to an input or in support of an output. The following principles apply:

(1) Specific requirements should be stated with all the characteristics of a good SRS
  - *correct*
  - *unambiguous*
  - *complete*
  - *consistent*
  - *ranked for importance and/or stability*
  - *verifiable*
  - *modifiable*
  - *traceable*

(2) Specific requirements should be cross-referenced to earlier documents that relate
(3) All requirements should be uniquely identifiable (usually via numbering like 3.1.2.3)
(4) Careful attention should be given to organizing the requirements to maximize readability (Several alternative organizations are given at end of document)

Before examining specific ways of organizing the requirements it is helpful to understand the various items that comprise requirements as described in the following subclasses. This section reiterates section 2, but is for developers not the customer. The customer buys in with section 2, the designers use section 3 to design and build the actual application.

Remember this is not design. Do not require specific software packages, etc unless the customer specifically requires them. Avoid over-constraining your design. Use proper terminology:
The system shall… A required, must have feature
The system should… A desired feature, but may be deferred til later
The system may… An optional, nice-to-have feature that may never make it to implementation.

Each requirement should be uniquely identified for traceability. Usually, they are numbered 3.1, 3.1.1, 3.1.2.1 etc. Each requirement should also be testable. Avoid imprecise statements like, "The system shall be easy to use" Well no kidding, what does that mean? Avoid "motherhood and apple pie" type statements, "The system shall be developed using good software engineering practice"

Avoid examples, This is a specification, a designer should be able to read this spec and build the system without bothering the customer again. Don't say things like, "The system shall accept configuration information such as name and address." The designer doesn't know if that is the only two data elements or if there are 200. List every piece of information that is required so the designers can build the right UI and data tables.

3.1 External Interfaces

This contains a detailed description of all inputs into and outputs from the software system. It complements the interface descriptions in section 2 but does not repeat information there. Remember section 2 presents information oriented to the customer/user while section 3 is oriented to the developer.

It contains both content and format as follows:

- *Name of item*
- *Description of purpose*
- *Source of input or destination of output*
- *Valid range, accuracy and/or tolerance*
- *Units of measure*
- *Timing*
- *Relationships to other inputs/outputs*
- *Screen formats/organization*
- *Window formats/organization*
- *Data formats*
- *Command formats*
- *End messages*

3.2 Functions

Functional requirements define the fundamental actions that must take place in the software in accepting and processing the inputs and in processing and generating the outputs. These are generally listed as "shall" statements starting with "The system shall…

These include:

- *Validity checks on the inputs*
- *Exact sequence of operations*
- *Responses to abnormal situation, including*

    - *Overflow*
    - *Communication facilities*
    - *Error handling and recovery*
- *Effect of parameters*
- *Relationship of outputs to inputs, including*
    - *Input/Output sequences*
    - *Formulas for input to output conversion*

It may be appropriate to partition the functional requirements into sub-functions or sub-processes. This does not imply that the software design will also be partitioned that way.

3.3 Performance Requirements

This subsection specifies both the static and the dynamic numerical requirements placed on the software or on human interaction with the software, as a whole. Static numerical requirements may include:

(a) The number of terminals to be supported
(b) The number of simultaneous users to be supported
(c) Amount and type of information to be handled

Static numerical requirements are sometimes identified under a separate section entitled capacity.

Dynamic numerical requirements may include, for example, the numbers of transactions and tasks and the amount of data to be processed within certain time periods for both normal and peak workload conditions.

All of these requirements should be stated in measurable terms.

For example,

95% of the transactions shall be processed in less than 1 second

rather than,

An operator shall not have to wait for the transaction to complete.

(Note: Numerical limits applied to one specific function are normally specified as part of the processing subparagraph description of that function.)

3.4 Logical Database Requirements

This section specifies the logical requirements for any information that is to be placed into a database. This may include:

- *Types of information used by various functions*

- *Frequency of use*
- *Accessing capabilities*
- *Data entities and their relationships*
- *Integrity constraints*
- *Data retention requirements*

If the customer provided you with data models, those can be presented here. ER diagrams (or static class diagrams) can be useful here to show complex data relationships. Remember a diagram is worth a thousand words of confusing text.

3.5 Design Constraints

Specify design constraints that can be imposed by other standards, hardware limitations, etc.

3.5.1 Standards Compliance

Specify the requirements derived from existing standards or regulations. They might include:

(1) Report format
(2) Data naming
(3) Accounting procedures
(4) Audit Tracing

For example, this could specify the requirement for software to trace processing activity. Such traces are needed for some applications to meet minimum regulatory or financial standards. An audit trace requirement may, for example, state that all changes to a payroll database must be recorded in a trace file with before and after values.

3.6 Software System Attributes

There are a number of attributes of software that can serve as requirements. It is important that required attributes by specified so that their achievement can be objectively verified. The following items provide a partial list of examples. These are also known as non-functional requirements or quality attributes.

These are characteristics the system must possess, but that pervade (or cross-cut) the design. These requirements have to be testable just like the functional requirements. Its easy to start philosophizing here, but keep it specific.

3.6.1 Reliability

Specify the factors required to establish the required reliability of the software system at time of delivery. If you have MTBF requirements, express them here. This doesn't refer to just having a program that does not crash. This has a specific engineering meaning.

3.6.2 Availability

Specify the factors required to guarantee a defined availability level for the entire system such as checkpoint, recovery, and restart. This is somewhat related to reliability. Some systems run only infrequently on-demand (like MS Word). Some systems have to run 24/7 (like an e-commerce web site). The required availability will greatly impact the design. What are the requirements for system recovery from a failure? "The system shall allow users to restart the application after failure with the loss of at most 12 characters of input".

3.6.3 Security

Specify the factors that would protect the software from accidental or malicious access, use, modification, destruction, or disclosure. Specific requirements in this area could include the need to:

- *Utilize certain cryptographic techniques*
- *Keep specific log or history data sets*
- *Assign certain functions to different modules*
- *Restrict communications between some areas of the program*
- *Check data integrity for critical variables*

3.6.4 Maintainability

Specify attributes of software that relate to the ease of maintenance of the software itself. There may be some requirement for certain modularity, interfaces, complexity, etc. Requirements should not be placed here just because they are thought to be good design practices. If someone else will maintain the system

3.6.5 Portability

Specify attributes of software that relate to the ease of porting the software to other host machines and/or operating systems. This may include:

- *Percentage of components with host-dependent code*
- *Percentage of code that is host dependent*
- *Use of a proven portable language*
- *Use of a particular compiler or language subset*
- *Use of a particular operating system*

Once the relevant characteristics are selected, a subsection should be written for each, explaining the rationale for including this characteristic and how it will be tested and measured. A chart like this might be used to identify the key characteristics (rating them High or Medium), then identifying which are preferred when trading off design or implementation decisions (with the ID of the preferred one indicated in the chart to the right). The chart below is optional (it can be confusing) and is for demonstrating tradeoff analysis between different non-functional requirements. H/M/L is the relative priority of that non-functional requirement.

ID	Characteristic	H/M/L	1	2	3	4	5	6	7	8	9	10	11	12
1	Correctness		▨											
2	Efficiency		▨	▨										
3	Flexibility		▨	▨	▨									
4	Integrity/Security		▨	▨	▨	▨								
5	Interoperability		▨	▨	▨	▨	▨							
6	Maintainability		▨	▨	▨	▨	▨	▨						
7	Portability		▨	▨	▨	▨	▨	▨	▨					
8	Reliability		▨	▨	▨	▨	▨	▨	▨	▨				
9	Reusability		▨	▨	▨	▨	▨	▨	▨	▨	▨			
10	Testability		▨	▨	▨	▨	▨	▨	▨	▨	▨	▨		
11	Usability		▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	
12	Availability		▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨

Definitions of the quality characteristics not defined in the paragraphs above follow.

- *Correctness - extent to which program satisfies specifications, fulfills user's mission objectives*
- *Efficiency - amount of computing resources and code required to perform function*
- *Flexibility - effort needed to modify operational program*
- *Interoperability - effort needed to couple one system with another*
- *Reliability - extent to which program performs with required precision*
- *Reusability - extent to which it can be reused in another application*
- *Testability - effort needed to test to ensure performs as intended*
- *Usability - effort required to learn, operate, prepare input, and interpret output*

THE FOLLOWING (3.7) is not really a section, it is talking about how to organize requirements you write in section 3.2. At the end of this template there are a bunch of alternative organizations for section 3.2. Choose the ONE best for the system you are writing the requirements for.

3.7 Organizing the Specific Requirements

For anything but trivial systems the detailed requirements tend to be extensive. For this reason, it is recommended that careful consideration be given to organizing these in a manner optimal for understanding. There is no one optimal organization for all systems. Different classes of systems lend themselves to different organizations of requirements in section 3. Some of these organizations are described in the following subclasses.

3.7.1 System Mode

Some systems behave quite differently depending on the mode of operation. When organizing by mode there are two possible outlines. The choice depends on whether interfaces and performance are dependent on mode.

3.7.2 User Class

Some systems provide different sets of functions to different classes of users.

3.7.3 Objects

Objects are real-world entities that have a counterpart within the system. Associated with each object is a set of attributes and functions. These functions are also called services, methods, or processes. Note that sets of objects may share attributes and services. These are grouped together as classes.

3.7.4 Feature

A feature is an externally desired service by the system that may require a sequence of inputs to effect the desired result. Each feature is generally described in as sequence eof stimulus-response pairs.

3.7.5 Stimulus

Some systems can be best organized by describing their functions in terms of stimuli.

3. 7.6 Response

Some systems can be best organized by describing their functions in support of the generation of a response.

3.7.7 Functional Hierarchy

When none of he above organizational schemes prove helpful, the overall functionality can be organized into a hierarchy of functions organized by either common inputs, common outputs, or common internal data access. Data flow diagrams and data dictionaries can be use dot show the relationships between and among the functions and data.

3.8 Additional Comments

Whenever a new SRS is contemplated, more than one of the organizational techniques given in 3.7 may be appropriate. In such cases, organize the specific requirements for multiple hierarchies tailored to the specific needs of the system under specification.

Three are many notations, methods, and automated support tools available to aid in the documentation of requirements. For the most part, their usefulness is a function of organization. For example, when organizing by mode, finite state machines or state charts may prove helpful; when organizing by object, object-oriented analysis *may prove helpful; when organizing by feature, stimulus-response sequences may prove helpful; when organizing by functional hierarchy, data flow diagrams and data dictionaries may prove helpful.*

In any of the outlines below, those sections called "Functional Requirement i" may be described in native language, in pseudocode, in a system definition language, or in four subsections titled: Introduction, Inputs, Processing, Outputs.

4. Change Management Process

Identify the change management process to be used to identify, log, evaluate, and update the SRS to reflect changes in project scope and requirements. How are you going to control changes to the requirements. Can the customer just call up and ask for something new? Does your team have to reach consensus? How do changes to requirements get submitted to the team? Formally in writing, email or phone call?

1. Identification of Changes: Changes can be identified by any member of the project team, the customer or other stakeholders. Changes must be documented in writing and submitted to the project manager.

2. Evaluation of Changes: The project manager, technical leads, and product owner will evaluate the change request and determine its impact on the project scope, timeline, and budget. They will also assess the feasibility and technical viability of the proposed change.
3. Approval of Changes: The product owner will approve or reject the change request based on its impact on the requirements and whether it aligns with the project goals. The project sponsor will also be consulted for any significant changes that may impact the project budget or timeline.

4. Communication of Changes: Once a change has been approved, the project manager will communicate the change to the relevant team members and stakeholders via email. This may include updating the SRS, project plan, and other project documentation.

5. Implementation of Changes: The technical leads will implement the change request in the system and ensure that it meets the customer's requirements and quality standards.

6. Monitoring and Control of Changes: The project manager will monitor the implementation of changes and ensure that they are completed within the agreed-upon timeline and budget. Any deviations from the plan will be reported to the project sponsor and product owner, and appropriate corrective action will be taken.

To ensure that changes are managed effectively, the project team will hold regular meetings to review the status of the project, identify risks and issues, and discuss any proposed changes. The change management process will be documented in the project plan and communicated to all relevant stakeholders. It is important to note that the customer may suggest changes to the requirements, but all changes must be evaluated and approved by the project team before they can be implemented. The team will reach a consensus on whether a change is necessary and if it can be accommodated within the existing project plan

5. Document Approvals

The following are the identified titles of the approvers of this SRS document:

1. Project Manager
2. Business Analyst
3. Technical Lead
4. Quality Assurance Lead
5. Customer or Stakeholder Representative

Name 1: ______________________________

[Insert Name of Approver]

Signature 1: ___________________________

[Insert Signature]

Date: _______________________________

[Insert Date]

Name 2: ______________________________

[Insert Name of Approver]

Signature 2: ___________________________

[Insert Signature]

Date: _______________________________

[Insert Date]

Name 3: ______________________________

[Insert Name of Approver]

Signature 3: ___________________________

[Insert Signature]

Date: _______________________________

[Insert Date]

Name 4: ______________________________

[Insert Name of Approver]

Signature 4: ___________________________

[Insert Signature]

Date: _______________________________

[Insert Date]

Name 5: ______________________________

[Insert Name of Approver]

Signature 5: ___________________________

[Insert Signature]

Date: _______________________________

[Insert Date]

Any changes to the SRS document for the Dietade project must be approved by the same individuals listed above. Changes must be submitted through the Change Management Process outlined in the SRS document.

This Document Approvals section ensures that the SRS document for the Dietade project has been thoroughly reviewed and approved by the appropriate individuals before proceeding with the development phase. It also ensures that any changes to the document are properly reviewed and approved by the same individuals to maintain consistency and accuracy throughout the project.

6. Supporting Information

The supporting information makes the SRS easier to use. It includes:

- *Table of Contents*
- *Index*
- *Appendices*

The Appendices are not always considered part of the actual requirements specification and are not always necessary. They may include:

(a) Sample I/O formats, descriptions of cost analysis studies, results of user surveys
(b) Supporting or background information that can help the readers of the SRS
(c) A description of the problems to be solved by the software
(d) Special packaging instructions for the code and the media to meet security, export, initial loading, or other requirements

When Appendices are included, the SRS should explicitly state whether or not the Appendices are to be considered part of the requirements.

Tables on the following pages provide alternate ways to structure section 3 on the specific requirements. You should pick the best one of these to organize section 3 requirements.