Software Requirements Specification

for

<Project>

Version 1.0

Prepared by <Team Name>

<date created>

Table of Contents

Table of Contents ii

Revision History ii

1. Introduction 1

1.1 Purpose 1

1.2 Product Scope 1

1.3 Definitions, Acronyms and Abbreviation 1

1.4 Organization 1

2. Overall Description 2

2.1 Product Perspective 2

2.2 Product Functions 4

2.3 User Classes and Characteristics 4

2.4 Design and Implementation Constraints/Limitations 4

2.5 Assumptions and Dependencies 4

2.6 Apportioning of Requirements 4

3. User and System Requirements 5

3.1 UR1 5

3.2 UR2 5

4. Other Nonfunctional Requirements 6

4.1 Performance Requirements 6

4.2 Safety Requirements 6

4.3 Security Requirements 6

4.4 Software Quality Attributes 6

5. Use Cases 7

5.1 Overall Use Case Diagram 7

5.2 Overall Activity Diagram 7

5.3 Specific Use Cases 7

6. System Structure 9

6.1 High-Level Class Diagram 9

6.2 Detailed Class Diagram 9

7. Architectural Design 10

7.1 Component Diagram 10

7.2 Architecture Diagram and Description 10

7.3 Deployment Diagram 10

Revision History

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

# Introduction

## Purpose

<Identify the product whose software requirements are specified in this document and delineate the purpose of the software to be specified.>

**Partial example**: The goal of this project is to provide a mobile application for Restaurant Clients and a web-portal for Restaurant Owners and Company’s administrators.

## 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. You should be able to describe the scope of the software under consideration by: (1) Identifying the software product(s) to be produced by name. (2) Explaining what the software product(s) will do. (3) Describing the application of the custom software being specified, including relevant benefits, objectives, and goals. (4) Being consistent with similar statements in higher-level specifications (e.g., the system requirements specification), if they exist.>

**Partial example:** The “Amazing Restaurant Finder” is a GPS-based mobile application, which helps people to find the closest restaurants based on the user’s current position, price, restaurant type and dish. Users view desired restaurants on a map and get navigation to them.

Software Features:

\* Restaurant owners provide their restaurant information using the web-portal.

\* An administrator of the web-portal verifies restaurant owners and manages user information.

## Definitions, Acronyms and Abbreviation

<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.>

**Partial Example:**

|  |  |
| --- | --- |
| **Acronym and Abbreviations** | **Definition** |
| SRS | Software Requirement System |
| PIC | Person in Charge |
| Grade | Level of fruits based on size and color. |

## Organization

<Describe what the rest of this SRS contains and how it is organized.>

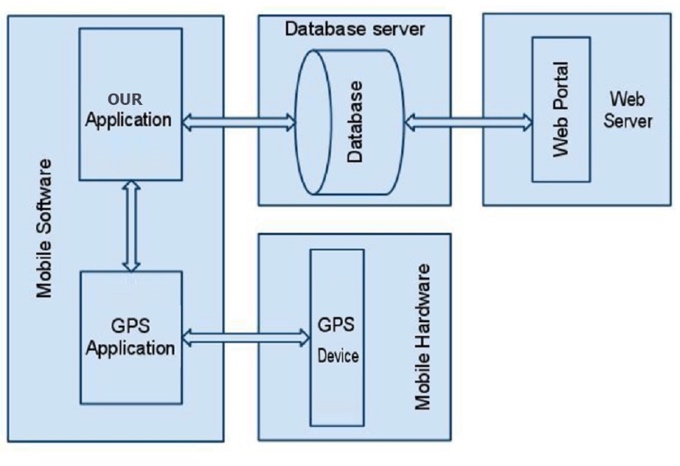
**Partial Example:** The next chapter, the Overall Description section, of this document gives an overview of the functionality of the product. It describes the informal requirements and is used to establish a context for the technical requirements specification in the next chapter. The third chapter, Requirements Specification section, this document is written primarily for both the stakeholders involved and developers. It describes in technical terms the details of the functionality of the product.

# Overall Description

## Product Perspective

< Describe the context and origin of the product being specified in this SRS by defining the system's relationship to other related products. If the product is an element of a larger system, then relate the requirements of that larger system to the functionality of the product covered by the SRS. If the product is an element of a larger system, then identify the interfaces between the product covered by the SRS and the larger system of which the product is an element. A simple diagram that shows the major components of the overall system, subsystem interconnections, and external interfaces can be helpful. One can also reference the high-level class diagram, architecture diagram (if needed).>

**Partial example**: The mobile application requires both Internet and GPS connection to fetch and display results. All system information is maintained in a database, which is located on a webserver. The mobile application interacts with the GPS-Navigator software, which is required to be already installed on the user’s mobile phone.



### System Interfaces

*<List each system interface and identify the functionality of the software to accomplish the system requirement and the interface description to match the system.>*

### 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. You may want to specify the following: (1) the logical characteristics of each interface between the software product and its users. This includes those configuration characteristics (e.g., required screen formats, page or window layouts, content of any reports or menus, or availability of programmable function keys) necessary to accomplish the software requirements. (2) All the aspects of optimizing the interface with the person who uses, maintains, or provides other support to the system. This may simply comprise a list of do's and don'ts on how the system will appear to the user. One example may be a requirement for the option of long or short error messages. (3) A style guide for the user interface can provide consistent rules for organization, coding, and interaction of the user with the system.>

**Partial example:** A first-time user of the mobile application should see the log-in page when he/she opens the application, see Figure 2. If the user has not registered, he/she should be able to do that on the log-in page. If the user is not a first-time user, he/she should be able to see the search page directly when the application is opened, see Figure 3. Here, the user chooses the type of search he/she wants to conduct. Every user should have a profile page where they can edit their e-mail address, phone number and password, see Figure 4. Also, the user can set the mobile application to his/her preferred language.

Graphical user interface

Description automatically generated

### 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. For example, terminal support may specify full-screen support as opposed to line-by-line support >

**Partial Example:** Just like any other system, this system requires basic computer that consists of CPU, monitor, keyboard and mouse for input and output. Printer is needed for print functions too. Besides, for crops shipping, the system is connected with GPS device in every organization transportation vehicle to get the transportation details.

### 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.>

**Partial Example:** The system is a standalone system that does not use external system. But the system will require database management software like MySQL to manage the process of storing the system data, and web server software that organizes the distribution of users in the local network.

### 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.>

**Partial Example:**The system communication interface is completely dependent on server software to ensure correct send and retrieve data from the database. Other than that, an online protocol to connect between GPS and the system.

### Memory Constraints

*<Specify any applicable characteristics and limits on primary and secondary memory.>*

**Partial Example:**The program will need at least 50Mb HDD space and minimum 512Mb ram in order to run at optimum performance.

## 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. For example, an SRS for an accounting program may use this part to address customer account maintenance, customer statement, and invoice preparation without mentioning the vast amount of detail that each of those functions requires. A picture of the major groups of related requirements and how they relate, such as a top-level data flow diagram or class diagram, activity diagram, overall use case 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 and the semantic description 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.>

## Design and Implementation Constraints/Limitations

<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).>

## Assumptions and Dependencies

<List each of the factors that affect the requirements stated in the SRS. These factors are not design constraints on the software but any changes to these factors can affect the requirements 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. For example, an assumption may be that a specific operating system will be available on the hardware designated for the software product. If, in fact, the operating system is not available, the SRS would then have to change accordingly.>

## Apportioning of Requirements

*<Identify requirements that may be delayed until future versions of the system (e.g., blocks and/or increments).>*

# User and System Requirements

*<Provide a traceable list of user requirements and their system requirements mapping.* ***User requirements****, often referred to as user needs, describe what the user does with the system, such as what activities that users must be able to perform.* ***User requirements*** *talk about the problem domain, the world of the user. They describe what effects need to be achieved. These effects are the combined responsibility of the software, the hardware, and the users (together: the socio-technical system).* ***System requirements*** *are the building blocks developers use to build the system. They talk about the solution domain, the world of the software logic. They describe what the software must do (as opposed to the effects in the user's world that this may or may not achieve). They are the responsibility of the technical system alone (without the socio part). User requirements describe what the user should do. System requirements describe how will the user achieve user requirements when interacting with the system plus nonfunctional requirements>*

## UR1

*<Document the user requirements and the corresponding system requirements.>*

**Partial Example:**

**UR1.** The MHC-PMS shall generate monthly management reports showing the cost of drugs prescribed by each clinic during that month.

**\* SR1.1.** On the last working day of each month, a summary of the drugs prescribed, their cost, and the prescribing clinics shall be generated.

**\* SR1.2** The system shall automatically generate the report for printing after 17.30 on the last working day of the month.

## UR2

*<Document the user requirements and the corresponding system requirements.>*

# 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. Specify both the static and the dynamic numerical requirements placed on the software or on human interaction with the software as a whole: (1) Static numerical requirements may include the following: The number of terminals to be supported, the number of simultaneous users to be supported, Amount and type of information to be handled. (2) 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. The performance 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”. Numerical limits applied to one specific function are normally specified as part of the processing subparagraph description of that function.

**Partial example:**

Quality requirement 6

ID: QR6

TITLE: The response time of a search.

DESCRIPTION: The response time of a search is the overall time beginning with the initial user action (click on the search button) on the mobile device, the request going to server, the response received from the server, and finally the response processing by the mobile application.

METER: Measurements obtained from 1000 searches during testing (iOS 9, Android 5.0).

MUST: No more than 2 seconds during 100% of the searches during testing.

WISH: No more than 1 second during 100% of the searches during testing

## 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.>

# Use Cases

*<A* ***requirement*** *is typically a general statement, whereas a* ***use case*** *is typically a specific statement implied or derived from the requirement. A requirement may map to multiple use cases. A* ***scenario*** *might be a set of background assumptions that put a use case in context, or it might be grouping of use cases.>*

## Overall Use Case Diagram

*<This diagram shows the overall use case diagram for the whole system along with semantic description of all actors.>*

## Overall Activity Diagram

*<An activity diagram shows the sequence of activities, decision nodes, loops, and even concurrent activities. In this section, you have to write a short description accompanying the attached figure.>*

## Specific Use Cases

### UC-001: *Validate PIN*

#### Use Case Description

*<In this section, use the template below to document a single use case. You have to pick a use case that has normal, alternative, and error flows.>*

**UC-001 Sample Use Case Description:**

|  |  |
| --- | --- |
| **Use Case Name** | Validate PIN |
| **Summary** | System validates customer PIN |
| **Created by** | *<Student Name who created this use case.>* |
| **Actor(s):** | ATM Customer |
| **Precondition:** | ATM is idle, displaying a Welcome message. |
| **Main/Normal Sequence** | 1. Customer inserts the ATM card into the card reader. 2. If system recognizes the card, it reads the card number. 3. System prompts customer for PIN. 4. Customer enters PIN. 5. System checks the card’s expiration date and whether the card has been reported as lost or stolen. 6. If card is valid, system then checks whether the user-entered PIN matches the card PIN maintained by the system. 7. If PIN numbers match, system checks what accounts are accessible with the ATM card. 8. System displays customer accounts and prompts customer for transaction type: withdrawal, query, or transfer. |
| **Alternative Sequences:** | * **Step 2:** If the system does not recognize the card, the system ejects the card. * **Step 5:** If the system determines that the card date has expired, the system confiscates the card. * **Step 5:** If the system determines that the card has been reported lost or stolen, the system confiscates the card. * **Step 7:** If the customer-entered PIN does not match the PIN number for this card, the system re-prompts for the PIN. * **Step 7:** If the customer enters the incorrect PIN three times, the system confiscates the card. * **Steps 4–8:** If the customer enters Cancel, the system cancels the transaction and ejects the card. |
| **Exception:** | *None* |
| **Postcondition:** | Customer PIN has been validated. |

#### Use Case Diagram

*<Draw the use case diagram for mentioned case above.>*

#### Activity Diagram

*<Document the activity diagram along with description of the different messages annotated in the diagram. Refer to Hassan Gomaa “Software Modeling & Design” book on the case studies chapter to see a sample.>*

#### Sequence Diagram

*<Document the sequence diagram along with description of the different messages annotated in the diagram. Refer to Hassan Gomaa “Software Modeling & Design” book on the case studies chapter to see a sample.>*

# System Structure

*<In this section, we list all available analysis models along with brief description>*

## High-Level Class Diagram

*<The class diagram is used to model the static structure of a system, thus describing the elements of the system and the relationships between them. These elements and the relationships between them do not change over time.>*

## Detailed Class Diagram

*<The class diagram is used to model the static structure of a system, thus describing the elements of the system and the relationships between them. These elements and the relationships between them do not change over time. In this section, along the class diagram, you have to document each class using the below template table.>*

**Partial Example of User Class Description:**

|  |  |  |
| --- | --- | --- |
| **User** | ***This is a base class in which manager and customer extend. This class provides the login ability that is shared between the two as well as some shared variables.*** | |
| **Public:** Yes | |
| **Relationships** | **Associations:** *None* |
| **Aggregations:** *None* |
| **Generalization:** *None* |
| **Variables** | Username: String  Password: String  Email: String  LastActivity: Time  Postal Address: String  Name: String  LoggedIn: Boolean |
| **Functions** | Login  Logout |

# Architectural Design

## Component Diagram

*< The purpose of a component diagram is to show the relationship between different components in a system. In this section, you have to write a short description accompanying the attached figure along with description of different components>*

## Architecture Diagram and Description

<The purpose of an architecture diagram is to show how the different components are related with respect to a selected architectural pattern. In this section, you have to write a short description accompanying the attached figure along with a description of the design goals behind selected a specific design pattern.>

## Deployment Diagram

*<The purpose of this diagram is to show the hardware topology used and the runtime system assigned. The hardware encompasses processing units in the form of nodes as well as communication relationships between the nodes. A runtime system contains artifacts that are deployed to the nodes. In this section, you have to write a short description accompanying the attached figure along with the proper architecture design and design goals, etc.>*