Widerpay

Software Requirement Specification Document

Items that are intended to stay in as part of your document are in bold; explanatory comments are in italic text. Plain text is used where you might insert wording about your project.

The document in this file is an annotated outline for specifying software requirements, adapted from the IEEE Guide to Software Requirements Specifications (Std 830-1993)

(WIDERPAY PROJECT)

(Team Name and Number)

(Team Members)

Software Requirements Specification

Document

Version: (n) Date: (12/01/2020)

Table of Contents

1. Introduction

1.1 Purpose

This SRS document is intended as a functional and non-functional requirements of an improvement to the Widerpay application. This improvement will utilize previous libraries used in the mpos app. However there are still several functions required to be added. Elkanah developers and designers should use this document for development

1.2 Scope

We will be rebuilding the entire widerpay system. Currently this system only does POS transactions by connecting with the mpos device and funds transfer with Flutterwave NIP which was included in December 2020. Current implementation Uses Java libraries to process transactions. The new system must be capable of accomplishing the above tasks and also new functions like the payment of bills, airtime and data recharge, virtual cards(Much later on) etc. Also the scope will encompass an admin application that monitors all transactions that takes place on the agent application, enrollment of new agents, the know your customer initiative, reconciliation, etc.

The whole process from enrolling to reconciling is disconcerted. The sole aim of this software is to build a structure around the processes. Also the new software should targets cross platforms(android, iOS and windows). Therefore framework that target cross platforms should be used (XAMARIN).

1.3 Definitions, Acronyms, and Abbreviations.

API – Application Programmers interface

JAVA – Programming language created by sunsystems.

C# - Programming language

XAMARIN – Framework of C# that targets multiple platforms

KYC – Know your customer

ISO – International Standard Operations server

1.4 References

Document Title	Version No
How to use the Widerpay App	1
Super Admin manual Not yet written	

Others

https://github.com/ostilo/TEMPPROJECT/raw/master/collection-release.apk

www.report.elkanahtech.com

1.5 Overview

The remainder of this document contains an overview of the requirements that have to be in place for the systems on which our implementation will run and the requirements for the implementation itself.

The overall description details the way in which the product will fit into a system as a whole and all of the systems interface and human interfaces will be described. This information is most useful to people who plan on putting this system into implementation as the pre-requisites of the system will be entailed. Potential users of the system should focus more on this section.

The Specific Requirements section will detail the specific requirements of the product we are developing. These requirements are those that will be used by the developers to implement the

system. The requirements will be useful in developing the Design that will eventually be used to produce the final product. Developers should focus on this section.

2. The Overall Description

2.1 Product Perspective

The system that we are building is meant to replace an already existing system called WIDERPAY. It is a fintech application with a wallet system (recently added). Customers approach our agents for withdrawal of funds. Agents uses the app to debit the customers and the funds submits to the wallet. This is related to the Fintech app of Paycentre and baxi.

The processes for setting up a user on the widerpay as it is presently includes the user obtaining and filling a form. The form fields include the following

- Merchant Contact information:

  Business Name, Email Address, Website Address, Office Tel No

- Merchant Principal Information:

Name of Official, Street Address, Telephone Number, Email

- Settlement Details

Mpos Device S/N, Bank name, Nuban account number, Account Name, Account Type, Sort Code.

The form is also a record of what ETL keeps as KYC. This is not encompassing of what is expected of KYC. After the form is filled, the user is then enrolled by tying the POS serial number to the user nuban account. The user can then begin to use the app.

The current system for POS withdrawal is easy to use and the UI is great. The user inputs the amount a customer wants to withdraw. Proceeds to process transaction, an account type is selected, user proceeds, the application connects to an external device which is the mpos. The customer inputs pin on the mpos device and proceeds. The mpos connects back to the application providing the response from the ISO server. The current system also lacks robustness. The user can only carry out withdrawal and not until recently funds transfer. ***The new system will encompass bill payments, recharge cards amongst others.***

After this a link is sent (See link in References). The agent registers again with the following information Phone number, Full name, email address, Transaction pin,

The Mpos serial number is then tied to the client Id(Phone number) of the customer. Customer can begin to use the mpos for accepting and giving out cash.

The existing system is encumbered with manual process and lacks structure. The new system will be process oriented and structured. Through the admin application, from filling of form by the user to support and training of user will be handled by teams through various modules on the app. If a process is not completed, the next process cannot happen.

In the process described above, there is no structural system from enrollment to usage. Any activity can happen arbitrarily. The new system will be structured and robust to contain the missing features in the existing one. Also the new system will target not only the android platform but also the iOS and web. The web version will cover mostly for activation and reports for the users and the full modules for the admin.

In the process described above, there is no structural system from enrollment to usage. Any activity can happen arbitrarily. The new system will be structured and robust to contain the missing features in the existing one. Also the new system will target not only the android platform but also the iOS and web. The web version will cover mostly for activation and reports for the users and the full modules for the admin.

Note

A block diagram showing the major components of the larger system, interconnections, and external interfaces can be helpful. This is not a design or architecture picture. It is more to provide context, especially if your system will interact with external actors. The system you are building should be shown as a black box. Let the design document present the internals.

The following subsections describe how the software operates inside various constraints.

2.1.1 System Interfaces

The interfaces is divided into two main part.

1. USER INTERFACE:

This interface is where the user/agent interact with the software. The interface will include the modules and pages.

Login/registration pages:
The user can logs in through this interface.
New user can register via this interface
Password can be reset using the forgot password button on the login page.

Dashboard page: This will also include various pages and fragments.
Wallet Balance: There is an interface that shows the wallet balance, client id and fund account.

POS: POS transactions will be initiated via this interface.This is connected to an outside system. The mpos.
Transfers: This is where transfer to bank account happens and also to other wallet users
Pay Bills: This is the interface used in the payment of bills.

History Page: This is where transaction history can be viewed.

Accounts: This is where sub accounts is created if the agents has more than one centres or Pos device.
Create Sub-accounts: This is where sub account is created
Commissions: If you are a super-agent. This is where your commission balances appear and you can also transfer it to your main wallet.
Rewards can be added here.

Settings: This is where settings to the account is done by the user.
Account details: Full name, email, client id is displayed here
Transaction Pin: Transaction pin is set up here or could be changed here.
Change Password: Password can be changed here.
MPOS: Settings relating to mpos is found here.
Device Info: App version, client id, about Widerpay and privacy, feedback is provided.
Support: This is where the user goes to interact with Widerpay officers.
Reports: This will mostly be web based. This is where users can pull reports in whatever format they want.

2. ADMIN INTERFACE:

This is where the whole activities happening on the user app is monitored and controlled. It will contain the following interfaces.

KYC (Know Your Customer): When the customers registers, the details submits here for the kyc team to confirm payments.

New Registration: This is where new registration of users submits for kyc. When a particular user registers via the app. The details submits to this module for the KYC team for confirmation. The KYC after confirming that the information provided is accurate. The information is sent to the enrollment team. Else the user is notified.
POS Registration: This works same as new registration. However this involves a more details like business name, business executives, business address, business account number CAC document(Optional) etc. Once information is confirmed. It is sent to the enrollment team.

Enrollment: This is where setting up, transaction privileges are given to the user. When the KYC team confirms user information. The details submits to the enrollment module. The enrollment also gives report privileges.
Enroll New user: New user information shows up here so that the charges field, transaction privileges can be given to the user.
POS User: Pos registration submits here so that serial number of the pos can be tied to the user. TIDs will also be assigned and charges set for the user.
Gives report privileges.

Reconciliation: This is where internal accounts are set up. Chargebacks are also logged here. The features includes
Manage Accounts: Bank accounts that the organization uses are set up here with an amount.
Chargebacks: Chargebacks are logged here with the following fields, RRN, PAN number, Transaction date, Auth/Stan, Date Reverted, Status, etc.
Reports:

Transaction: This shows real time transaction details as they are happening. An overview is shown on its dashboard that includes the total approved, total count, total failed.
Query. A query can be initiated on the go. The fields will include Pan number date and RRN(optional)

Support: This is a module where all complaints submits to and response can be provided.
Chats. This will include chats with merchants.
Whatsapp: This will open whatsapp to chat with a merchant that uses whatsapp to initiate.
Notifications: Notifications like when a new user registers should submit here, also notifications for support should be here.

The whole system will be built around the system interface. Processes will be developed along this interface. For example each system interfaces will serve as a department in the organization.

Modules will be developed for each system interface.

2.1.2 Interfaces

User will interact the app via the user GUI (details and features above)

Admin will interact via the admin app(see details above)

Designers will make sure texts and fonts are easily readable and interactive with icons and images complementing.

2.1.3 Hardware Interfaces

Hardware connected to the software is the POS and Mini Printer. A page from the POS module calls the mpos device after the device must have been connected via Bluetooth. The Mpos device is Me30s

The printer is used to output transaction details (Optional)

2.1.4 Software Interfaces

There are no specific software interfaces that will be used.

2.1.5 Communications Interfaces

There are no specific communication device interface.

2.1.6 Memory Constraints

Android
Minimum android version = Android 5
Minimum Memory(Ram) = 100mb
Minimum Memory(ROM) = 1mb

iOS

2.1.7 Operations

WITHOUT POS

The user download app from playstore. User Installs and registers. Phone number and email is confirmed to be correct and also to be sure it is not being used by another user. A terms and agreement is signed while registering. Customer now have access to the app but it is limited. Registration information is sent to the admin app as user submits. KYC picks it and does the necessary KYC on the user. KYC sends the user for enrollment. The enrollment officer input the charges and the necessary set up required. The enrollment then activates the user.

User can perform the following on the app after enrollment.

Fund wallet
Transfer Money
Pay Bulls,
Recharge cards
Other VAS

WITH POS

If a customer requests POS. POS can be requested for online or physically. A form is then Field agent gives another form to the customer to fill. These information is taken to the KYC to confirm

and corroborate the information. KYC sends to enrollment who then ties the POS to the agents and gives agent access to the POS module. Enrollment sends to Reconciliation who confirms charges.

If there is a complaint from the user. Support team assesses the issue, reach out to the team responsible and gets feedback to give the user.

2.1.8 Site Adaptation Requirements

Requirements needed to install the app is detailed in the “How to use the widerpay” referenced above

To use the POS. The customer will pay for the POS device and printer.

2.2 Product Functions

The product will replace the existing widerpay application
The product will have an admin, agent and super agent module.
The product will contain pos, transfer, bill payments other VAS.
The product will be used to monitor activities and settle superagents

2.3 User Characteristics

The user for this product is anyone who uses a smart phone and wants a seamless fintech solution to handle his/her money problems.. The user is expected to understand how to do basic things on a smart phone like installing and uninstalling an app

2.4 Constraints

Provide a general description of any other items that will limit the developer's options. These can include:

(1) Regulatory policies: We currently have the PTAD and aggregator license. We are working with NIBSS to get other licences and documentation.

(2) Hardware limitations (for example, signal timing requirements) : See Memory constraints above.

(3) Interface to other applications: Connecting with the printer and mpos.

(4) Parallel operation

(5) Audit functions

(6) Control functions

(7) Higher-order language requirements

(8) Signal handshake protocols (for example, XON-XOFF, ACK-NACK)

(9) Reliability requirements

(10) Criticality of the application

(11) Safety and security considerations

(12) Database: There is a current database being used. We are building based on the database.

This section captures non-functional requirements in the customers language. A more formal presentation of these will occur in section 3.

2.5 Assumptions and Dependencies

The design and the operations is dependent on the database schema. There is an existing database. The existing database might not be structured to accommodate some of the operations stated in this document.

2.6 Apportioning of Requirements.

All requirements must be completed for this project.

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.

The User Interface

The app will accept a number of user inputs when a user

Logs on to the app;
Fills out registration form;
Initiates a POS transaction;
Initiates a transfer;
Initiates bills payments;
Makes a complaint;
Update his/her profile;

User inputs will result in the following output

A welcome screen if the user login was successful;
Customer registration screen, terms and conditions screen, and registration confirmation screens;
Transaction response screen
Transfer response screen
Response screen.

The user interface will contain the following screen

The login screen will consist of a user dialogue box with two text boxes and two labels to accept ***username and password***. There will be three buttons: Login, New User? Sign Up, Need Help,

Chat with Support. This screen will be used to authenticate the user to the system, the user will be directed to the dashboard.

The Registration screen will consist of dialogue boxes with the following fields: First name, Last name, Email, Address, State(Drilldown), LGA(drill down), Gender Password, Confirm password. There will be one button with sign up. The user will be directed to the terms and conditions page. Once user has accepted the T&C, the user will be directed to the Login page if sign up is successful.

The main function of the **dashboard screen** will be to carry out the original purpose of why the app was built. So the dashboard will be designed asynchronously to enable users carry out tasks seamlessly without one impacting on the other. It will have four bottom navigation namely Home, History, Account History and Settings. On the ***home screen***, the following screens will be included in the dashboard; *POS, Transfer, Pay bills* etc. On the **History,** you have a list of transaction history which can be filtered. The account history will have **Sub account setup**, commissions and referrals.

The dashboard will have a card box that carries the wallet balance, Fund account.

The **POS screen** on the screen will consist of Withdraw and History page. The withdrawal page will have the following dialogue boxes; amount, transaction status page. The History page will have a list of transaction history and filter.

The **Transfer screen** will consist of two pages; Transfer to bank account, Transfer to widerpay account. The transfer to bank account will have dialog boxes with fields; account Number, Bank name, Account Name(Account name will be validated and populated automatically). Narration, Customer Phone Number(Optional). On clicking the next button at the bottom of the page. User will be asked to confirm the transaction before proceeding to finish the transaction.

The **Transaction history** will consist of two tabs: Transaction history and General overview (This is a charts and line graph)

The **Account History** will have a button to setup sub accounts and also a list of subaccounts.

The support page will consist of text boxes: chat with widerpay staff, Send a mail, Call a staff, contact support. The chat with support will open WhatsApp on the user phone to chat with a staff of widerpay. The send mail will open the default mail operator on the user phone. The call will initiate a call to widerpay support. The contact will chat with an agent on the app.

The **settings page** will include the following pages: The terms and conditions, set up pin, change password, Device Info, Mpos configuration

The ADMIN Interface

The admin interface will have the following modules/pages:

Transactions Module: This page is dedicated for real time transactions. An overview of the transactions is enough. Transactions can also be queried on this module. Query button will have the field Pan number and date as input.

The KYC Module: Registration information submits here, user can long press on a particular information or mark several to either send to enrollment or decline registration.

Enrollment Module will have the following the registration information as approved by the KYC. It shall be editable to include Transfer charges and Mpos charges etc.

Reconciliation Module: will have the chargeback module, Manage account page. The chargeback will have the following field as input: Txn date, Pan Number, RRN, Stan, Amount, Date Reverted, Merchant.

Support Module: This will consist of the following functions. A page to disseminate information to all the agents, a Chat box to communicate with user who has initiated chat from their app. A call logs to see the calls being initiated by a user.

3.1 External Interfaces

- Name of item: ME30s POS and Bluetooth Printer
- Description of purpose: The Me30s is the Mpos used in collecting payments. The bluetooth printer is used in printing receipts.
- Source of input or destination of output: Transaction is initiated from the app and sent to the me30s. Card is imputed in the Me30s and pin is inputted by the customer. The me30s process the transaction by sending it to the ISO server. The server sends response back to the me30s and the app. The app then send the details to the bluetooth printer to print.

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?

5. Document Approvals

Identify the approvers of the SRS document. Approver name, signature, and date should be used.

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.

Outline for SRS Section 3

Organized by mode: Version 1

3. Specific Requirements

3.1 External interface requirements

3.1.1 User interfaces

3.1.2 Hardware interfaces

3.1.3 Software interfaces

3.1.4 Communications interfaces

3.2 Functional requirements

3.2.1 Mode 1

3.2.1.1 Functional requirement 1.1

.....

3.2.1.n Functional requirement 1.n

3.2.2 Mode 2

.....

3.2.m Mode m

3.2.m.1 Functional requirement m.1

.....

3.2.m.n Functional requirement m.n

3.3 Performance Requirements

3.4 Design Constraints

3.5 Software system attributes

3.6 Other requirements

Outline for SRS Section 3

Organized by mode: Version 2

3. Specific Requirements

3.1 Functional Requirements

3.1.1 Mode 1

3.1.1.1 External interfaces

3.1.1.1 User interfaces

3.1.1.2 Hardware interfaces

3.1.1.3 Software interfaces

3.1.1.4 Communications interfaces

3.1.1.2 Functional Requirement

3.1.1.2.1 Functional requirement 1

.....

3.1.1.2.n Functional requirement n

3.1.1.3 Performance

3.1.2 Mode 2

.....

3.1.m Mode m

3.2 Design constraints

3.3 Software system attributes

3.4 Other requirements

Outline for SRS Section 3

Organized by user class (i.e. different types of users ->System Adminstrators, Managers, Clerks, etc.)

3. Specific Requirements

3.1 External interface requirements

3.1.1 User interfaces

3.1.2 Hardware interfaces

3.1.3 Software interfaces

3.1.4 Communications interfaces

3.2 Functional requirements

3.2.1 User class 1

3.2.1.1 Functional requirement 1.1

.....

3.2.1.n Functional requirement 1.n

3.2.2 User class 2

.....

3.2.m User class m

3.2.m.1 Functional requirement m.1

.....

3.2.m.n Functional requirement m.n

3.3 Performance Requirements

3.4 Design Constraints

3.5 Software system attributes

3.6 Other requirements

Outline for SRS Section 3

Organized by object (Good if you did an object-oriented analysis as part of your requirements)

3 Specific Requirements

3.1 External interface requirements

3.1.1 User interfaces

3.1.2 Hardware interfaces

3.1.3 Software interfaces

3.1.4 Communications interfaces

3.2 Classes/Objects

3.2.1 Class/Object 1

3.2.1.1 Attributes (direct or inherited)

3.2.1.1.1 Attribute 1

.....

3.2.1.1.n Attribute n

3.2.1.2 Functions (services, methods, direct or inherited)

3.2.1.2.1 Functional requirement 1.1

.....

3.2.1.2.m Functional requirement 1.m

3.2.1.3 Messages (communications received or sent)

3.2.2 Class/Object 2

.....

3.2.p Class/Object p

3.3 Performance Requirements

3.4 Design Constraints

3.5 Software system attributes

3.6 Other requirements

Outline for SRS Section 3

Organized by feature (Good when there are clearly delimited feature sets.

3 Specific Requirements

3.1 External interface requirements

3.1.1 User interfaces

3.1.2 Hardware interfaces

3.1.3 Software interfaces

3.1.4 Communications interfaces

3.2 System features

3.2.1 System Feature 1

3.2.1.1 Introduction/Purpose of feature

3.2.1.2 Stimulus/Response sequence

3.2.1.3 Associated functional requirements

3.2.1.3.1 Functional requirement 1

.....

3.2.1.3.n Functional requirement n

3.2.2 System Feature 2

.....

3.2.m System Feature m

.....

3.3 Performance Requirements

3.4 Design Constraints

3.5 Software system attributes

3.6 Other requirements

Outline for SRS Section 3

Organized by stimulus (Good for event driven systems where the events form logical groupings)

3 Specific Requirements

3.1 External interface requirements

3.1.1 User interfaces

3.1.2 Hardware interfaces

3.1.3 Software interfaces

3.1.4 Communications interfaces

3.2 Functional requirements

3.2.1 Stimulus 1

3.2.1.1 Functional requirement 1.1

.....

3.2.1.n Functional requirement 1.n

3.2.2 Stimulus 2

.....

3.2.m Stimulus m

3.2.m.1 Functional requirement m.1

.....

3.2.m.n Functional requirement m.n

3.3 Performance Requirements

3.4 Design Constraints

3.5 Software system attributes

3.6 Other requirements

Outline for SRS Section 3

Organized by response (Good for event driven systems where the responses form logical groupings)

3 Specific Requirements

3.1 External interface requirements

3.1.1 User interfaces

3.1.2 Hardware interfaces

3.1.3 Software interfaces

3.1.4 Communications interfaces

3.2 Functional requirements

3.2.1 Response 1

3.2.1.1 Functional requirement 1.1

.....

3.2.1.n Functional requirement 1.n

3.2.2 Response 2

.....

3.2.m Response m

3.2.m.1 Functional requirement m.1

.....

3.2.m.n Functional requirement m.n

3.3 Performance Requirements

3.4 Design Constraints

3.5 Software system attributes

3.6 Other requirements

Outline for SRS Section 3

Organized by functional hierarchy (Good if you have done structured analysis as part of your design.)

3 Specific Requirements

3.1 External interface requirements

3.1.1 User interfaces

3.1.2 Hardware interfaces

3.1.3 Software interfaces

3.1.4 Communications interfaces

3.2 Functional requirements

3.2.1 Information flows

3.2.1.1 Data flow diagram 1

3.2.1.1.1 Data entities

3.2.1.1.2 Pertinent processes

3.2.1.1.3 Topology

3.2.1.2 Data flow diagram 2

3.2.1.2.1 Data entities

3.2.1.2.2 Pertinent processes

3.2.1.2.3 Topology

.....

3.2.1.n Data flow diagram n

3.2.1.n.1 Data entities

3.2.1.n.2 Pertinent processes

3.2.1.n.3 Topology

3.2.2 Process descriptions

3.2.2.1 Process 1

3.2.2.1.1 Input data entities

3.2.2.1.2 Algorithm or formula of process

3.2.2.1.3 Affected data entities

3.2.2.2 Process 2

3.2.2.2.1 Input data entities

3.2.2.2.2 Algorithm or formula of process

3.2.2.2.3 Affected data entities

…..

3.2.2.m Process m

3.2.2.m.1 Input data entities

3.2.2.m.2 Algorithm or formula of process

3.2.2.m.3 Affected data entities

3.2.3 Data construct specifications

3.2.3.1 Construct 1

3.2.3.1.1 Record type

3.2.3.1.2 Constituent fields

3.2.3.2 Construct 2

3.2.3.2.1 Record type

3.2.3.2.2 Constituent fields

…..

3.2.3.p Construct p

3.2.3.p.1 Record type

3.2.3.p.2 Constituent fields

3.2.4 Data dictionary

3.2.4.1 Data element 1

3.2.4.1.1 Name

3.2.4.1.2 Representation

3.2.4.1.3 Units/Format

3.2.4.1.4 Precision/Accuracy

3.2.4.1.5 Range

3.2.4.2 Data element 2

3.2.4.2.1 Name

3.2.4.2.2 Representation

3.2.4.2.3 Units/Format

3.2.4.2.4 Precision/Accuracy

3.2.4.2.5 Range

…..

3.2.4.q Data element q

3.2.4.q.1 Name

3.2.4.q.2 Representation

3.2.4.q.3 Units/Format

3.2.4.q.4 Precision/Accuracy

3.2.4.q.5 Range

3.3 Performance Requirements

3.4 Design Constraints

3.5 Software system attributes

3.6 Other requirements

Outline for SRS Section 3

Showing multiple organizations (Can’t decide? Then glob it all together)

3 Specific Requirements

3.1 External interface requirements

3.1.1 User interfaces

3.1.2 Hardware interfaces

3.1.3 Software interfaces

3.1.4 Communications interfaces

3.2 Functional requirements

3.2.1 User class 1

3.2.1.1 Feature 1.1

3.2.1.1.1 Introduction/Purpose of feature

3.2.1.1.2 Stimulus/Response sequence

3.2.1.1.3 Associated functional requirements

3.2.1.2 Feature 1.2

3.2.1.2.1 Introduction/Purpose of feature

3.2.1.2.2 Stimulus/Response sequence

3.2.1.2.3 Associated functional requirements

…..

3.2.1.m Feature 1.m

3.2.1.m.1 Introduction/Purpose of feature

3.2.1.m.2 Stimulus/Response sequence

3.2.1.m.3 Associated functional requirements

3.2.2 User class 2

.....

3.2.n User class n

.....

3.3 Performance Requirements

3.4 Design Constraints

3.5 Software system attributes

3.6 Other requirements

Outline for SRS Section 3

Organized by Use Case (Good when following UML development)

3. Specific Requirements

3.1 External Actor Descriptions

3.1.1 Human Actors

3.1.2 Hardware Actors

3.1.3 Software System Actors

3.2 Use Case Descriptions

3.2.1 Use Case 1

3.2.2 Use Case 2

3.2.n Use Case n

3.3 Performance Requirements

3.4 Design Constraints

3.5 Software system attributes

3.6 Other requirements

WIDERPAY AGENT AGREEMENT

This agreement made this …… day of …….2021

Between