3e Software House
ul. Podbipięty 51
02-762 Warszawa
tel. (22) 822 48 68
http://3e.pl

PARENT PROJECT NAME – XXX MODULE NUMBER & NAME[1]

Document version:	0.1
Last modification date:	2019-08-06
Status:	Open / Close
Document author:	jsmith
Task#:	https://redmine.yourcompany.com/issues/11111
Tags:	[Keywords]
Validity date:	2019-07-10

Scope:[2]

Brand	*Yourclientcompany.com*
Market	*Poland, Germany, Russia etc.*
API / Mobile	*YES*
Frontend	*YES*
Backend	*YES*

Change history

Version	Author	Date	Description
0.1	BM		First version EN

Contact persons

Side	Scope	Name and surname	E-mail
Client	Accountable	Andrew Smith	andrew.smith@Client.eu
Client	Business	Jack Long	jack.long@Client.eu
Client	Business	Catherine Bond	catherine.bond@Client.eu
Software House	Analysis	Joe Average	joe.average@softwarehouse.eu

[1] In case the requirements document describes a part of a larger project, the title should contain comprehensive reference to parent project and the particular module covered by the document.

[2] A table with a basic client-specific, non-functional description of the project, specifying project's overall scope.

3e Software House
ul. Podbipięty 51
02-762 Warszawa
tel. (22) 822 48 68
http://3e.pl

1. INTRODUCTION

1.1 Business purposes

This section contains a description of the feature commissioned by client. This description is the basis for analysis and evaluation. It answers the question what's the reason for introducing the change. The aim of this section is to outline the scope of the development or changes and also to highlight other constraints, resulting from legal regulations, the company's internal procedures etc. In this section both functional and non-functional requirements are allowed.

1.2 Reason and scope of the development/changes

1.2.1 Description of current process (as stated)[3]

General description of the current process

1.2.2 Description of process after change (to-be stated)

General description of the process after change

1.3 Description of the dependencies

How the introduced or changed functionality is dependent on other applications and/or how it influences other applications. This field is particularly important when the described module is part of a larger project.

1.4 Definitions

Here are the definitions for all the advanced concepts that are used in the document and are necessary to properly understand the user stories and acceptance. Eg.:

client – the person who completed the registration process and entered the payment details

prospect – the person who completed the registration process but haven't entered payment details

guest – unregistered visitors to the website

visit - any user interaction with the website, that do not include purchase

transaction - any visit to the website that ends with a successful purchase

failed transaction - a transaction, that has been interrupted during the purchasing phase

[3] Optional field. Used when both parties decide the description of a current process is necessary for a precise specification of planned changes.

bounce - the visit that consist of viewing a single page and lasts less than 5 seconds

2. DESCRIPTION OF THE FORM FIELDS (IF RELEVANT)

2.1 First form

2.1.1. First Field

2.1.1.1 Name of field

i.e. "name", "surname", "postal code" etc.

2.1.1.2 Type of field

i.e. Number, text, dropdown menu, radiobutton, etc.

2.1.1.3 Size

Number of characters allowed, i.e. 255

2.1.1.4 If compulsory

Yes/No

2.1.1.5 Dependencies

In this field you should indicate, if the content of the field is dependent on other fields (i.e. is active only when another field has been filled) or its content is limited to a dictionary of specified terminology.

2.1.2 Second Field

(description)

2.1.3 Third field

(description)

2.2 Second form

2.2.1 First Field (description)

2.2.2. Second Field (description)

2.2.3. Third Field (description)

2.3 Third form

. etc.

CAUTION! If the fields such as "Name" or "Address" appear more than once within this part of the application covered in this particular document, it should be described as many times as it appears with an emphasis on the differences between the validation rules and other features of each version. I.e. one field may be dependent on another one, whilst in a different place such dependencies may not exist.

Nonetheless, so long as we stick to the general principle of, **covering only single application's module in each requirements document,** such situations should be rather rare.

3. MOCK-UPS, GRAPHICAL DESIGN (IF RELEVANT FOR THE PROJECT)

In this section all mock-ups and/or graphical visualisations of the final product should be attached.

4. USER STORIES AND TESTING SCENARIOS

This section contains the most important data, the description of functionalities that should be developed or changed during the project. Gathered together they should form a four-column table, similar to the example. The meaning of the columns and an example of the content that might fill them is provided in the below table and in the following paragraphs.

EXAMPLE:

Functionality	Title	User story	Acceptance criteria
Adding	Adding a shipment	As a client I need to provide my address and the address of the addressee, print the label to be glued to the parcel and order a courier visit to the convenient time.	- all the address fields are filled, except the optional fields such as second address line and number of the flat - the form accepts only a proper date and time format
	Adding a local shipment	As a client I want to provide an address of an addressee in Poland	- the form accepts only a proper Polish postal code verified by the postal code database - the form accepts all the types of shipment, an envelope, a parcel and a pallet
	Adding a European shipment	As a client I want to provide the address of an addressee on the European continent, but outside Poland.	- the form doesn't accept Polish address data, especially Polish postal code - the form accepts all the types of shipment, an envelope, a parcel and a pallet - the form accepts only European countries

	Adding an international shipment (outside Europe)	As a client I want to provide the address of an addressee in the country outside Europe	- the form doesn't accept Polish address data, especially Polish postal code - the form doesn't accept names of European countries - the form doesn't accept pallets, only envelopes and parcels.

4.1 Functionalities (first column of the table)

In this section at least one type of functionality should be provided. The functionalities should be grouped in a way that each group represents similar types of operations regarding their logic and place in the business process. This form of presentation allows us to identify common parts of the user story and the common acceptance criteria at the beginning and then, in every user story we can focus only the criteria that are specific to that particular story.

The most typical functionality types are:

Adding data (like placing an order, adding a new user, adding a blog post etc.)

Modifying data

Deleting existing data entries

Displaying requested data

Verifying data

Generating new data

etc

4.2 Title (second column of the table)

In this section the detailed name of each particular functionality tile should be added i.e. in Title that falls into the group "Verifying" could be "Verifying of the postal code provided by the user with the database".

4.3 User story (third column of the table)

User story in Gherkin nomenclature describing the expected functionality in the form As a <agent> I want <what> in order to <why>. That means it should include information about the agent, the activity and finally – the benefit achieved through this activity. This part is filled by the client and is later enhanced and completed by the analyst during the initial workshops.

Below example source: https://en.wikipedia.org/wiki/Behavior-driven_development

Story: Returns go to stock

As a store owner

In order to keep track of stock

I want to add items back to stock when they're returned.

Scenario 1: Refunded items should be returned to stock

Given that a customer previously bought a black sweater from me

And I have three black sweaters in stock.

When they return the black sweater for a refund

Then I should have four black sweaters in stock.

Scenario 2: Replaced items should be returned to stock

Given that a customer previously bought a blue garment from me

And I have two blue garments in stock

And three black garments in stock.

When they return the blue garment for a replacement in black

Then I should have three blue garments in stock

And two black garments in stock.

4.4 Acceptance criteria (fourth column of the table)

In this section you should enumerate all the acceptance criteria that would clearly define the way the application should behave and/or highlight its constraints, data formats etc.

Example

- the entered postal code should be 6 characters long

- the entered postal code should math the format xx-xxx

- the entered postal code should contain only numbers

- the field “postal code” is compulsory

4.5 Testing scenarios

Testing scenario takes the form of a concise table, containing an exhaustive set of combinations of input data allowing the team of testers to try every possible combination and make sure, that the application reacts in proper way. The form of a table allows us to substantially reduce the amount of text needed to describe every particular scenario.

EXAMPLE:

Name	Surname	Telephone	Address	Output	Message
John	Smith	+48503848555	Warsaw, Al Ujazdowskie 5/6	OK	Thank you for providing the recipient data!
John	no data	+48503848555	Warsaw, Al Ujazdowskie 5/6	Error	The “surname” field cannot be blank.
no data	no data	+48503848555	Warsaw, Al Ujazdowskie 5/6	Error	The “name” and “surname” fields cannot be blank.

.

.

.

.

.

etc

4.6 Data model

Theoretically, the data model as a relatively sophisticated element of the IT architecture of a system does not fit into the client requirements document. In practice it is beneficial to include the data model, even if it is preliminary one, which can be later modified during the development process. Viewing the expected tables and fields is often a great eye opener for clients and helps them to realize the requirements they hadn’t thought about earlier, or (wrongly) had taken for granted. For example: should the address be stored in a single text string or divided into separate fields like street, house number, flat number etc. The first way is simpler but the second is more fool proof and allows a much more sophisticated search and aggregation operations. Such decisions should be taken at the beginning of the development process, because rearranging the database while the application is already up and running is a relatively complicated and expensive operation, and can generate many bugs. **We highly recommend engaging a database expert at the stage of building the user requirements document, to verify the preliminary data model regarding efficiency issues.**

5. FUNCTIONAL REQUIREMENTS

5.1 Frontend changes

Business requirements for the frontend part of the application should always contain a written description explaining the meaning of the changes. If the goal of the project is to change the already existing part of the interface, the description should be supported by screenshots with comprehensive sketches of the required changes.

If completely new screen layouts are to be developed, the document should contain their mock-ups, with a detailed description, containing

* as precise as possible the placement of every element

* size of the element (like buttons, picture, text fields, dropdown menus etc.)

* fonts (names, size, color codes)

* color codes

This section is usually written by the client and then eventually expanded and clarified by the analyst, but the description and mock-ups can be prepared after a detailed conversation with the client.

5.2 Backend changes

Requirements regarding backend changes should have a written description, containing all the information necessary for a quick and effective implementation, namely

* products, that are affected by the changes

* data sources

* information, if the changes affects the application itself or the API

The source of changes in most cases comes directly from the user stories, and therefore they should be described by the analyst, based on a detailed analysis of the user stories section.

5.3 API changes

Requirements related to API changes should include the following information:

5.3.1 Which methods should be modified and:

* what are new fields

* what data type and value range should be accepted in each field

* what are the dependencies between the fields and methods

5.3.2 What new methods should be available in API, and:

* what are new fields

* what data type and value range should be accepted in each field

* maximum field length

* if empty values, white characters or null strings are allowed

* what are the dependencies between the fields and methods

* dependencies between different methods

5.4 System integration and data transmission (and data flow) *(Optional part, rather to be included in a separate, technical document.)*

Requirements related to data transmission, databases and other related applications should include:

- Which data should be transmitted
- Where it should be transmitted
- What are the dependencies between different streams of data
- What processes and products need data transmission
- What is the maximum size of the transmitted data
- Whether empty field, white character or null values are allowed

There are different approaches towards the presentation of data flow in this part of the document. Some argue, they should be included there, however data flows should usually be considered earlier, before the preparation of this document. Data flow elements must be analysed during the preparation of user stories, acceptance criteria, testing scenarios and data models.

5.5 Security *(Optional part, rather to be included in a separate, technical document.)*

If there is a need for a security audit of the application code before release, that information should be included in the requirements document.

5.6 Dependencies on other projects/changes

Any dependencies on other processes, products or validations should be included in the requirements document, to minimize the risk of logical and functional conflicts between the applications. Specifically, changes related to data transmission, API integrations and backend changes should be carefully analysed and described.

6. USE CASE DIAGRAMS *(Optional part. Used mostly, when the Client Requirements Document is intended to be used by third parties, i.e. to be quoted by other software houses.)*