The COSMIC Functional Size Measurement Method

Version 4.0.2

Case Study

Sizing Natural Language/ User Stories/ UML Use Cases for Web and Mobile Applications using COSMIC FSM

Version 1.2

May 2019

Authors and reviewers of Version 1.1		
Asma Sellami * Institut Supérieur d'Informatique et de Multimédia de Sfax University of Sfax Tunisia	Alain Abran* École de Technologie Supérieure, Université du Québec Canada	Arlan Lesterhuis* MPC Netherlands
Mariem Haoues* Institut Supérieur d'Informatique et de Multimédia de Sfax University of Sfax Tunisia	Hanêne Ben-Abdallah* Higher Colleges of Technology Dubai, UAE	Francisco Valdes Souto National Autonomous University of Mexico, Science Faculty, CDMX Mexico City, Mexico
Bruce Reynolds Tecolote Research United States		
Authors and reviewers of Version 1.0.3		
Asma Sellami* Institut Supérieur d'Informatique et de Multimédia de Sfax University of Sfax Tunisia	Alain Abran* École de Technologie Supérieure, Université du Québec Canada	Arlan Lesterhuis* MPC Netherlands
Mariem Haoues* Institut Supérieur d'Informatique et de Multimédia de Sfax University of Sfax Tunisia	Colin Hammond United Kingdom	Hanêne Ben-Abdallah* Higher Colleges of Technology Dubai, UAE
Francisco Valdes Souto National Autonomous University of Mexico, Science Faculty, CDMX Mexico City, Mexico	Bruce Reynolds Tecolote Research United States	
Authors and reviewers of Version 1.0.2		
Asma Sellami* Institut Supérieur d'Informatique et de Multimédia de Sfax University of Sfax Tunisia	Alain Abran* École de Technologie Supérieure, Université du Québec Canada	Sylvie Trudel * Université du Québec à Montréal Université du Québec Canada
Mariem Haoues* Institut Supérieur d'Informatique et de Multimédia de Sfax University of Sfax Tunisia	Arlan Lesterhuis* The Netherlands	Luigi Lavazza Universita degli Studi dell'Insubria Italy
Hanêne Ben-Abdallah* King Abdulaziz University Jeddah Kingdom of Saudi Arabia	Charles Symons* United Kingdom	

* Authors of this case study

The following is a partial account of the evolution of this case study.

Date	Reviewer (s)	Modifications / Additions
2016-05-20	Sellami, Haoues, Ben-Abdallah, Abran	This case study was entitled “Sizing the functional requirements in COSMIC FP units as documented in natural language/ UML use cases: A case study for Web and Mobile Apps” It was tested with graduate students at ETS
2016-06-01	Sellami, Haoues, Ben-Abdallah, Abran, Symons, Lesterhuis	Updates to: - propose a new title “Case study: Sizing natural language/ UML Requirements for Web and Mobile Applications using COSMIC FSM” - update the purpose of measurement - clarify the used terminolgy and avoid confusion - correct the measurement of the error/confirmation messages
2016-06-25	Sellami, Haoues, Ben-Abdallah, Abran, Lesterhuis	Updates to: - identify Functional Processes - correct the use case diagrams and its related textual descriptions - identify object of interests and data groups
2016-07-03	Sellami, Haoues, Ben-Abdallah, Abran, Lesterhuis, Trudel	Updates to: - identify object of interests and data groups - adjust layout of the document - replace “data validate” action type in UC textual description by “data persist”
2016-09-05	Sellami, Haoues, Ben-Abdallah, Abran, Lesterhuis, Trudel, Lavazza	Significant revision. Updates to: - propose a new title “Case Study Sizing Natural Language/UML Use Cases for Web and Mobile Applications using COSMIC FSM” - adjust UC diagrams - explain the UC textual descriptions - clarify data groups and data attributes, and the description of some functional processes - delete some redundant tables - delete some functional processes
2017-11-10	Sellami, Haoues, Ben-Abdallah, Abran	Significant revision. Updates to: - Clarify the UC description and their associated FP in both mobile and web apps. - Follow the COSMIC method v4.0.2 (COSMIC, 2017) - Take into account some points raised by (Haoues *et al.*, 2017b) - Clarify which use cases are represented each with one functional process and which use cases are represented with two functional processes
2018-01-18	Sellami, Haoues, Ben-Abdallah, Abran, Colin Hammond	Minor revision. Updates to: - Breackdown REQ 9 into REQ 9 and REQ 10 in order to visualize separately each UC (Delete an order) and (View the list of orders), and that correspond respectively to (FP33) and (FP32) - Clarify the use cases description and their associated functional processes in web app s to avoid redundant actions. In particular, UC involving Modify or Delete object. - Adjust the total Functional Size of RestoSys according to the main changes.
2019-03-16	Sellami, Haoues, Ben-Abdallah, Abran, Lesterhuis, Valdes Souto, Reynolds	Significant revision. Updates to: - Breackdown FP2 into FP2 –FP6 and FP3 into FP 7 –FP9 to correct the FP identification - Delete REQ 10, consider cheap meals as one of the item families and delete all its FP - Clarify the data exchanges between mobile and web app components for REQ1 and REQ3 - Update the data persistent storages - Rectify the objects of interests, data groups, and data attributes - Adjust the total functional size of RestoSys according to the main changes
2019-05-27	Sellami, Haoues, Ben-Abdallah, Abran, Lesterhuis	Minor revision. Updates to: - Quantification of REQ using User Stories Formats - Identification of functional processes

Table of Contents

1 RESTAURANT MANAGEMENT SYSTEM REQUIREMENTS

1. 1 Context

This Case Study presents the measurement results of applying the COSMIC FSM method (Version 4.0.2) to the “Restaurant Management System”. This system is composed of hardware and software components. The software component named RestoSys includes two parts: a mobile app and a web app.

The “Restaurant Management System” requirements are documented in a technical report of the final project of study for the Professional Master's Degree at the University of Sfax-Tunisia (Mhadhbi 2013).

This case study is structured as follows:

- Chapter 1 presents the background for the “Restaurant Management System” case study. It provides different representations of Hardware/Software requirements. The focus of this chapter is especially on the description of the Functional Requirements as documented in natural language, user stories, UML use case diagrams, and the textual descriptions of use cases using action-type (see Appendix A).
- Chapter 2 presents the measurement strategy phase, including: measurement purpose and scope, functional users, level of granularity and decomposition.
- Chapter 3 presents the mapping phase including: triggering events, functional processes, data groups, data attributes, and objects of interest.
- Chapter 4 presents the measurement phase. The measurement results are provided in two ways:
  (i) a direct application of COSMIC FSM method on functional requirements described in natural language in section 4.1; and
  (ii) an application of COSMIC FSM method on the action-type of use case actions in section 4.2.

1. 2 Hardware Components

As shown in Figure 1, the hardware configuration of “Restaurant Management System” is composed of:

- A database server allowing a high storage capacity.
- A Web server that hosts the web app part of RestoSys.
- The admin interacts with RestoSys via a PC with Web browser, while the waiter interacts with the same system via a Smartphone.

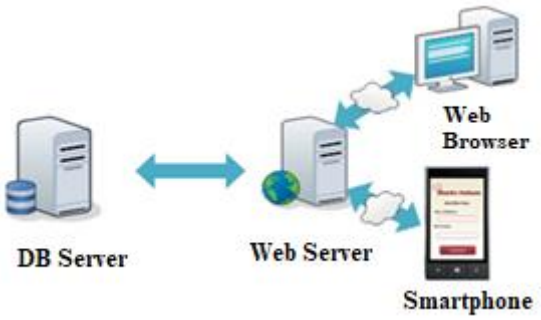

Figure 1 Hardware Configuration

1. 3 Software-Hardware Interactions

The RestoSys is composed of two parts: a mobile app and a web app, collaborating according to the client-server model:

- The RestoSys ensures communication between the mobile app (the waiter) and the web app (the admin).
- The admin physically maintains the database (that is included in the DB server) directly via a Web browser.

The RestoSys application includes the following functionality:

- The waiter logs in with help of the mobile app.
- The Web app retrieves data from the DB and provides the required data to the mobile app.
- The waiter maintains a customer's order (by adding or modifying an order).
- The admin logs in with help of the Web browser.
- The Web app retrieves/provides data from/to the DB.
- The admin maintains the required data via the web app, see the requirements in section 1.4 A.

1. 4 Software Requirements

According to (COSMIC 2017), software requirements are classified into three categories: Functional User Requirements (FUR), Non-Functional Requirements (NFR), and Project Requirements and Constraints (PRC). FUR express “what the software shall do in terms of tasks and services”. NFR include “any requirement for a hardware/software system or for a software product, including how it should be developed and maintained and how it should perform in operation, except any functional user requirement for the software”. PRC describe “how a software system project should be managed and resourced or constraints that affect its performance”. In this case study we will mainly use “Requirements” for convenience.

A. Requirements

In this section, the Requirements (REQ) for the RestoSys are identified. First, the REQ are described in natural language. Then, those same REQ are modeled in different formats including user stories and UML Use Case diagrams. Each use case in a UML Use Case diagram is next detailed in textual descriptions of use cases using action-type as presented in Appendix A.

a. REQ expressed in natural language

The RestoSys includes the following tasks:

- Order management: This task allows the waiter to add, and-or modify an order via a Smartphone. It also allows the admin to delete an order. During working hours, the waiters (mobile app) and the admin (web app) are continuously connected.
- Account management: This task involves employees’ management, and it enables access to the application with a username and a password.
- Restaurant Menu management: This task allows the management of item[1]families and the classification of items into item families.

Note that the users of RestoSys (waiter and-or admin) must be logged in before executing one of the previous tasks (Order management, Account management, and Restaurant Menu management).

The RestoSys must establish the following functionality:

[1]Item is used to describe a dish and beverage

- **REQ1 - Login "Mobile App":** The employees (waiters) must be logged in to RestoSys using their Smartphones. Each waiter has a username and a password.
- **REQ2 - Maintain Order:** The waiter can add and-or modify an order. The waiter takes an order by selecting the customer's table and the chosen items. Each customer is installed at a table and can request a set of items such as: fruit salad, orange juice, etc.
- **REQ3 - Login "Web app":** The employee (administrator) must be logged in to access the web app using a username and a password.
- **REQ4 - Maintain an Employee:** The admin can add an employee (waiter). The admin can view and-or modify a waiter data. The admin can delete an existing waiter and-or view the employees list.
- **REQ5 - Maintain Item:** The admin can add a new item by entering the necessary data for an item. The admin can also view, modify an item. The admin can delete an existing item and view the items list. Each item is classified into a single item family.
- **REQ6 - Maintain Item family:** The admin can add a new item family by entering the required data. The admin may also view and-or modify an item family data. The admin can delete an existing item family, and view the list of existing item families. Examples of item family: juice, salad, soda, etc.
- **REQ7 - Maintain Table:** The admin can add a new table. The admin may also view and-or modify a specific table data. The admin can delete an existing table and view the table list to know the table state (unoccupied, occupied or reserved).
- **REQ8 - View the List of Orders:** The admin can view the list of orders.
- **REQ9 - Delete an Order:** The admin can delete a customer's order.

These requirements will be detailed in the following sections using UML use case diagrams.

b. REQ expressed in UML Use case diagram

Identification of Actors. The employees of the RestoSys are the admin and the waiters.

- Admin: is the manager of the application. The admin can manage the entire RestoSys. The admin can access to the web app via his username and his password (REQ3).
- Waiter: is responsible for customers' orders. The waiter can access to the mobile app via his Smartphone and using his username and his password (REQ1).

Identification of User Stories and Use Cases. Based on the REQ listed in section A) a, the description of each of these REQ is presented below in the form of User Story (see Section B.) and Use case descriptions (see Section C).

B. User Stories Description

To make requirements visible not only to users but also to developers, the notion of splitting-up large sets of user stories into smaller stories is applied. Large US are identified with a high level of detail. Some of them are split into 'smaller' user stories. For example, "US2: Maintain Order" can be split into "US 2.1: Add an Order" and "US 2.2: Modify an Order". The user stories with their corresponding description are presented in Table 1.

Table 1 User stories Identification

Actor	REQ	User Story (US)	User Story Description
Waiter	**REQ1**	**US1 : "Login" Mobile app**	**As a** waiter **I want to** connect to the RestoSys **so that** I can use its services/ functionality
	REQ2	**US2: Maintain Order**	**As a** waiter **I want to** maintain orders **so that** I can add or modify a customer's order
		US2.1 : Add an Order	**As a** waiter **I want to** add an order **so that** I can add a new order as requested by the customer
		US2.2 : Modify an Order	**As a** waiter **I want to** modify an order **so that** I can modify an order as requested by the customer

	REQ3	**US3 : "Login" Web app**	**As an** admin **I want to** connect to the RestoSys **so that** I can use its services/ functionality
	REQ4	**US4: Maintain an Employee**	**As an** admin **I want to** maintain employees **so that** I can update the employees data/list
Admin		US4.1 : Add an employee	**As a**n admin I **want** to add an employee **so that** I can add a new employee to the employee list
		US4.2 : View the Employee List	**As an** admin **I want to** view the list of employees **so that** I can view the users list
		US4.3 : View an Employee data	**As an** admin **I want to** view an employee data **so that** I can view an employee data
		US4.4 : Modify an Employee Data	**As an** admin **I want to** modify an employee data **so that** I can modify an employee data
		US4.5 : Delete an employee	**As an** admin **I want to** delete an employee **so that** I can delete an employee
	REQ5	**US5 : Maintain Item**	**As an** admin **I want to** maintain an item **so that** I can update the items data/list
		US5.1 : Add an Item	**As an** admin **I want to** add an item **so that** I can add a new item
		US5.2 : View the Items List	**As an** admin **I want to** view the list of items **so that** I can view the items list
		US5.3 : View Item data	**As an** admin **I want to** view an item data **so that** I can view an item data
		US5.4 : Modify an Item	**As an** admin **I want to** modify an item data **so that** I can modify an item
		US5.5 : Delete an Item	**As an** admin **I want to** delete an item **so that** I can delete an item
	REQ6	**US6 : Maintain Item Family**	**As an** admin **I want to** maintain an item family **so that** I can update the items family data/list
		US6.1 : Add an Item family	**As an** admin **I want to** add an item family **so that** I can add a new item family
		US6.2 : View the Item families list	**As an** admin **I want to** view the item families list **so that** I can view the items families list
		US6.3 : View Item family data	**As an** admin **I want to** view an item family data **so that** I can view an item family data
		US6.4 : Modify an Item family	**As an** admin **I want to** modify an item family **so that** I can modify an item family
		US6.5 : Delete an Item family	**As an** admin **I want to** delete an item family **so that** I can delete an item family
	REQ7	**US7 : Maintain Table**	**As an** admin **I want to** maintain a table **so that** I can update the table data/list
		US7.1 : Add a table	**As an** admin **I want to** add a table **so that** I can add a new table
		US7.2 : View the tables List	**As an** admin **I want to** view the tables list **so that** I can view the tables list
		US7.3 : View table data	**As an** admin **I want to** view the table data **so that** I can view table data
		US7.4 : Modify table data	**As a** admin **I want to** modify table data **so that** I can modify table data
		US7.5 : Delete a table	**As an** admin **I want to** delete a table **so that** I can delete a table

	REQ8	**US8: View the list of orders**	**As an** admin **I want to** view the list of orders **so that** I can view the orders list
	REQ9	**US9: Delete an order**	**As an** admin **I want to** delete an order **so that** I can delete an order

C. Use Cases Description

The global use case diagram, as presented in **Figure 2**, identifies the main functionality provided by RestoSys. Thus, the employees (admin and waiters) must be logged in before maintaining data and orders. Each use case identified in the global use case diagram can include one or more use cases. As an example, the use case *"Maintain Data"* in the global use case diagram is detailed using four use cases: "Maintain Employee", "Maintain Item", "Maintain Item family", and "Maintain Table". The use case *"Maintain Order"* for the "Admin" is also detailed using "View the List of Orders" and "delete an order" (**Figure 3**).

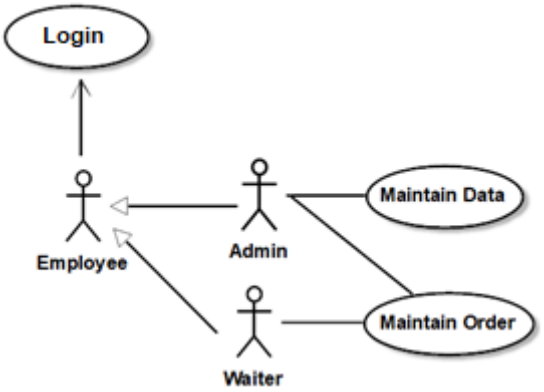

Figure 2 Global Use Case Diagram

Use Cases Textual Descriptions. Each detailed use case identified in **Figure 2** and **Figure 3** is represented using a use case textual description (see Appendix A). Note that a use case textual description represents a discrete unit of interaction between the system RestoSys (mobile app and web app) and its users (Waiter and Admin).

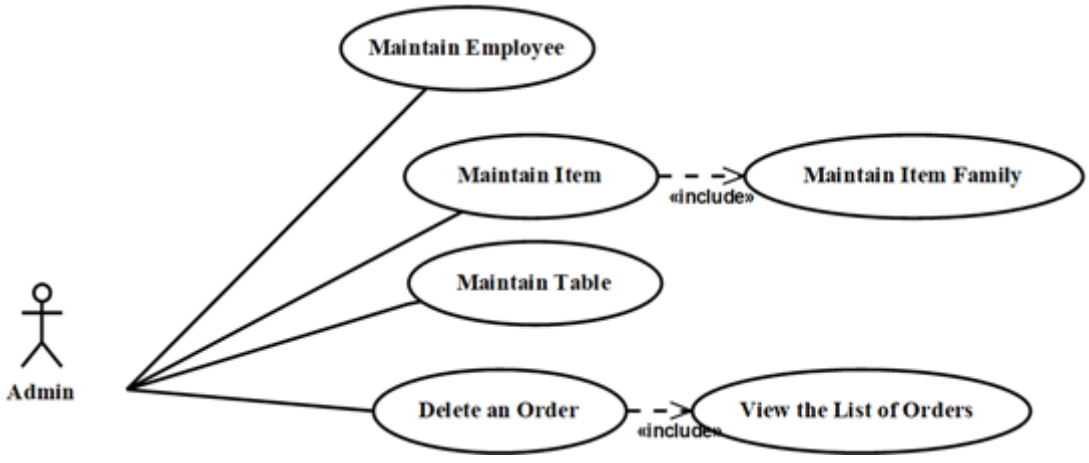

Figure 3 Detailed Use Case Diagram for "Maintain Data" and "Maintain Order" use cases

REQ1 Use case "Login" Mobile app

Name: <Login>
Description: <This use case describes how the employee (the waiter) login to the RestoSys>
Actors: <Primary actor: waiter>

Begin

MS /* Main scenario */

1. <Waiter><Expletive><The waiter requests for a connection to the RestoSys>
2. <System><Expletive><The RestoSys asks the waiter to enter his username and password>
3. <Waiter><Request><The waiter enters his/her username and password> [<Waiter ID>]
4. <System><Response & Request> <The RestoSys checks the validity of username and password> [<Waiter ID>] [<UserState>]
5. <System><Response><The RestoSys displays the user state (Connected / Not Connected) to the waiter> [<UserState>]

/*If the username and password are valid, the waiter is connected (user state = Connected)*/

/*Point 4 of login is detailed by using two actions: Response and Request. These actions lead to an exchange of data. For Response action, out-Parameters are required (username and password). For Request action, one Int-Parameter is required (user state). */

End

ES /* Error scenario */

Begin <Username and password are incorrect, begin at Num '3'>

6. <System><Request & Response> <The RestoSys displays the message "the username and/or the password are invalid">

End
End use case

REQ2: Use case "Maintain Order"

Name:<Maintain Order>
Description: <This use case allows the waiter to add or modify a customer's order>
Actors: <Primary actor: Waiter>

Begin

MS /* Main Scenario */

Add an Order

Begin

1. <System><Expletive><when the waiter logged in, the list of restaurant options is displayed>.
2. <Waiter><Expletive><The waiter presses the option "Add a new order">.
3. <Waiter><Request & Response><The waiter enters the table where the customer wants to be installed. The system changes the order state from active to closed > [<Table ID>&<Table ID>].

/*If a new order for the same table is entered, the previous order (if any) should be closed. In this case, there is always at most one order 'active')*/

4. <System><Request & Response><The system displays the list of item families> [<Item family data>] & [<Item family data>].
5. <Waiter><Request & Response><The waiter chooses the item family based on the items requested by the customer> [<Item family ID>&<Item familyID>].
6. <System><Request & Response><The system displays the list of items according to the selected Item families> [<Item Data>&<Item Data>].
7. <Waiter><Request & Response><The waiter selects the items requested by the customer and specifies for each item the recommended quantity and customer's comment. The system creates a new order> [<Order Items>] & [<Order Items>].

End

AS /*Alternative Scenario*/

Modify an Order

Begin

1. <System><Expletive><When the waiter logged in, the list of restaurant options is displayed>.
2. <Waiter><Expletive><The waiter presses the option “Modify an order”>
3. <Waiter><Request & Response><The waiter enters the table where the customer is installed> [<Table ID>&<Table ID>].
4. <System><Request & Response><The system displays the list of items ordered by the customer with their quantities and comments> [<Order Items> & <Item data> & <Order Items>].
5. <Waiter><Request & Response><The waiter deletes an existing item, modifies items' quantities or modifies items' comments. The system updates the order data > [<Order Items> & <Order Items>].

End

ES /* Error Scenario */

Begin<No table available, begin at Num '2' in the main scenario>
2.1 <System>< Request & Response><The system displays the message “No table available”>
/* The scenario restarts at point 1. */
End
Begin<Orders list is empty, begin at Num '2' in the alternative scenario>
2.1 <System>< Request & Response><The system displays the message “Orders list is empty”>
/* The scenario restarts at point 1. */
Begin<No table available, begin at Num '7' in the main scenario>
7.1 <System>< Request & Response><The system displays the message “New order not created”>
/* The scenario restarts at point 1. */
End
Begin<No table available, begin at Num '5' in the alternative scenario>
5.1 <System>< Request & Response><The system displays the message “The changes cannot be saved”>
/* The scenario restarts at point 1. */
End

End

End use case

REQ3 Use case "Login" Web app

Name: <Login>
Description: <This use case describes how the admin logs into the Web app>
Actors: <Primary actor: Admin>

Begin

MS /* Main scenario */

1. <Admin><Expletive><The Admin requests for a connection to the system>
2. <System><Expletive><The system asks the Admin to enter his username and password>
3. <Admin><Request><The Admin introduces his username and password> [<Admin Data>]
4. <System><DataRecovery><The system checks the validity of username and password> [<Admin Data>]
5. <System><Response><The RestoSys displays the user state (Connected / Not Connected) to the admin> [<UserState>]

/*If the username and password are valid, the Admin is connected*/

End

ES /* Error scenario */

Begin <Username and password are incorrect, begin at Num '3'>

6. <System><Response><The system displays the message "the username and the password are invalid.">

End
End use case

REQ4: Use case "Maintain an Employee"

Name: <Maintain Employee>
Description: <This use case allows employees update>
Actors: <Primary actor: Admin>

Begin

MS /* Main Scenario*/

Add an Employee

1. <Admin><Expletive><The admin requests to add a new employee>.
2. <System><Expletive><The system displays the employee form and requires the admin to enter the necessary data for adding new employee>.
3. <Admin><Request><The admin enters the employee data> [<Employee Data>].
4. <System><Expletive><The system checks that all required data are entered correctly>.
5. <System><DataRecovery><The system checks if admin is trying to add an existing waiter> [<Employee Data>].
6. <System><DataPersist><The system adds the new employee> [<Employee Data>].

AS /* Alternative Scenario */

View the Employees List

1. <Admin><Request><The admin selects to view the list of employees> [<Employee Data>].
2. <System><DataRecovery><The system retrieves the list of employees> [<Employee Data>].
3. <System><Response><The system displays the list of employees> [<Employee Data>].

View an Employee Data

1. <Admin><Request><The admin selects the desired employee> [<Employee ID>].
2. <System><DataRecovery><The system retrieves employee data from the Database> [<Employee Data>].
3. <System><Response><The system displays the data of the selected employee to the admin> [<Employee Data>].

Modify an Employee Data

/*Modify an Employee data should inherit all the default actions from View an Employee Data [1,2,3] */

1. <Admin><Request><The admin modifies employee data that can be changed> [<Employee Data>].
2. <System><DataPersist><The system saves the change> [<Employee Data>].

Delete an Employee

/*Delete an employee should inherit the default actions from View an Employee Data [1,2,3] */

1. <Admin><Request><The admin chooses the user to be deleted> [<username &<password>].
2. <System><DataPersist><The system deletes the selected user> [<username>, <password>].

ES /* Error Scenario */

Begin <Employee already exists, begin at Num '5' in the *main scenario*>
5.1 <System><Response><The system displays the message “Employee already exists”>
/* The scenario restarts at point 2. */
End

Begin <The employees list is empty, begin at Num '3' in the *alternative scenario* 'View an Employees List'>
3.1<System><Response><The system displays the message “The employees list is empty”>
/* The scenario restarts at point 1. */
End

Begin <The employees list is empty, begin at Num '3' in the *alternative scenario* “Modify an Employee Data”>
3.1<System><Response><The system displays the message “The employees list is empty”>
/* The scenario restarts at point 1. */
End
Begin<The employees list is empty, begin at Num '3' in the *alternative scenario* “Delete an Employee”>
3.1<System><Response><The system displays the message “The employees list is empty”>
/* The scenario restarts at point 1. */
End
End use case

REQ5: Use case “Maintain Item”

Name: <Maintain item>
Description: <This use case allows the admin to add, modify, view and delete an item>
Actors: <Primary actor: Admin>

Begin

MS /* Main Scenario */

Add an Item

1. <Admin><Expletive><The admin asks for adding a new item>.
2. <System><Expletive><The system displays the item form and asks the admin to enter the necessary data for adding an item>.
3. <Admin><Request><The admin enters the data of the item and instructs the system to validate the addition> [<Item Data>].
4. <System><Expletive><The system checks that all required data are entered correctly>.
5. <System><DataRecovery><The system checks if admin is trying to add an existing item> [<Item Data>].
6. <System><DataPersist><The system adds the new item> [<Item Data>].

AS /* Alternative Scenario */

View the Items list

1. <Admin><Request><The admin selects to view the items' list> [<Item Data>].
2. <System><DataRecovery><The system retrieves the list of items> [<Item Data>].
3. <System><Response><The system displays the list of items> [<Item Data>].

View an Item Data

1. <Admin><Request><The admin selects the desired item> [<Item ID>].
2. <System><DataRecovery><The system retrieves the item data from the Database> [<Item Data>].
3. <System><Response><The system displays the selected item data> [<Item Data>].

Modify an Item

/*Modify an Item should inherit all the default actions from View Item Data [1,2,3] */

1. <Admin><Request><The admin modifies the desired fields (name, price, quantity, image, etc.)> [<Item Data>].
2. <System><DataPersist><The system saves the change> [<Item Data>].

Delete an Item

/*Delete an Item should inherit the default actions from View Item Data [1,2,3] */

1. <Admin><Request><The admin selects the item to be deleted> [<Item ID>].
2. <System><DataPersist><The system deletes the selected item> [<Item ID>].

ES /* Error Scenario */

Begin<Item already exists, begin at Num '5' in the *main scenario*>

5.1 <System><Response><The system displays the message "Item already exists">

/* The scenario restarts at point 2. */

End

Begin<The Item list is empty, begin at Num '3' in the *alternative scenario* "Modify an Item">

3.1<System><Response><The system displays the message "The item list is empty">

/* The scenario restarts at point 1. */

End

Begin<The Item list is empty, begin at Num '3' in the *alternative scenario* "Delete an Item">

3.1<System><Response><The system displays the message “The item list is empty”>
/* The scenario restarts at point 1. */
End
End use case

REQ 6: Use case “Maintain Item Family”

Name: <Maintain Item Family>
Description: <This use case allows the admin to add, modify, view and delete an item family>
Actors: <Primary actor: Admin>

Begin

MS /* Main Scenario */

Add an Item Family

1. <Admin><Expletive><The admin requests for adding a new item family>.
2. <System><Expletive><The system displays the item family form and asks the admin to enter the necessary data for adding an item family>.
3. <Admin><Request><The admin enters the data of the item family and instructs the system to validate the addition> [<Item Family Data>].
4. <System><Expletive><The system checks that all required data are entered correctly>.
5. <System><DataRecovery><The system checks if the Item Family already exists> [<Item Family Data>].
6. <System><DataPersist><The system adds the new item family> [<Item Family Data>].

AS /* Alternative Scenario */

View Item Families List

1. <Admin><Request><The admin selects to view the item families list> [<Item Families Data>].
2. <System><DataRecovery><The system retrieves the item families list> [<Item Family Data>].
3. <System><Response><The system displays the item families list> [<Item Family Data>].

View Item Family Data

1. <Admin><Request><The admin selects the desired item family> [<Item family ID>].
2. <System><DataRecovery><The system retrieves the item family data from the Database> [<Item family Data>].
3. <System><Response><The system displays the selected Item family data to the admin> [<Item family Data>].

Modify an Item Family

/*Modify an Item Family should inherit all the default actions from View Item Family Data [1,2,3] */
1. <Admin><Request><The admin modifies the desired fields> [<Item family Data>].
2. <System><DataPersist><The system saves the change> [<Item family Data>].

Delete an Item Family

/*Delete an Item Family should inherit the default actions from View Item Family Data [1,2,3] */
1. <Admin><Request><The admin selects the item family to be deleted> [<Item family ID>].

2. <System><DataPersist><The system deletes the selected item family> [<Item family ID>].

ES /* Error Scenario */

Begin<Item Family already exists, begin at Num '5' in the *main scenario*>
5.1 <System><Response><The system displays the message “Item family already exists”>
/* The scenario restarts at point 2. */
End
Begin<The Item Family list is empty, begin at Num '3' in the *alternative scenario* “Modify an Item Family”>
3.1<System><Response><The system displays the message “The item family list is empty”>
/* The scenario restarts at point 1. */
End

Begin<The Item Family list is empty, begin at Num '3' in the *alternative scenario* “Delete an Item Family”>
3.1<System><Response><The system displays the message “The item family list is empty”>
/* The scenario restarts at point 1. */
End
End use case

REQ 7: Use case “Maintain Table”

Name: <Maintain Table>
Description: <This use case allows the admin to add, view, modify and delete a table>
Actors: <Primary actor: Admin>

Begin

MS /* Main Scenario */

Add a Table

1. <Admin><Expletive><The admin requests to add a new table>.
2. <System><Expletive><The system displays the table form and asks the admin to enter the necessary data for adding a table>.
3. <Admin><Request><The admin enters all the table data> [<Table Data>].
4. <System><Expletive><The system checks that all required data are entered correctly>.
5. <System><DataRecovery><The system checks if the table already exists> [<Table Data>].
6. <System><DataPersist><The system adds the new table> [<Table Data>].

AS /* Alternative Scenario */

View Tables List

1. <Admin><Request><The admin selects to view the tables list> [<Table Data>].
2. <System><DataRecovery><The system retrieves the tables list> [<Table Data>].
3. <System><Response><The system displays the tables list > [<Table Data>].

View a Table Data

1. <Admin><Request><The admin selects the desired table> [<Table ID>].

2. <System><DataRecovery><The system retrieves the Table Data from the Database> [<Table ID>].
3. <System><Response><The system displays the data of the selected table to the admin> [<Table Data>].

Modify Table Data

/*Modify Table Data should inherit all the default actions from View Table Data [1,2,3] */
1. <Admin><Request><The admin modifies the table data> [<Table Data>].
2. <System><DataPersist><The system saves the change> [<Table Data>].

Delete a Table

/*Delete a Table should inherit the default actions from View Table Data [1,2,3] */
1. <Admin><Request><The admin chooses the table to be deleted> [<Table ID>].
2. <System><DataPersist><The system deletes the selected table> [<Table ID>].

ES /* Error Scenario */

Begin<Table already exists, begin at Num '5' in the *main scenario*>
5.1 <System><Response><The system displays the message “Table already exists”>
/* The scenario restarts at point 2. */
End
Begin<The Table list is empty, begin at Num '3' in the *alternative scenario* “Modify Table Data”>
3.1<System><Response><The system displays the message “Tables list is empty”>
/* The scenario restarts at point 1. */
End
Begin<Tables list is empty, begin at Num '3' in the *alternative scenario* 'Delete a Table>
3.1<System><Response><The system displays the message “Tables list is empty”>
/* The scenario restarts at point 1. */
End
End use case

REQ 9: Use case “View the List of Orders”

Name: <View the List of Orders>
Description: <This use case allows to view the list of orders>
Actors: <Primary actor: Admin>

Begin

MS /* Main Scenario */

View the List of Orders

1. <Admin><Request><The admin selects to view the orders list> [<Order Data>].
2. <System><DataRecovery><The system retrieves the orders list> [<Order Data>].
3. <System><Response><The system displays the orders list> [<Order Data >].

ES /* Error Scenario */

Begin<Orders list is empty, begin at Num '2' in the *main scenario*>
2.1 <System><Response><The system displays the message “Orders list is empty”>
/* The scenario restarts at point 1. */
End

REQ 10: Use Case “Delete an Order”

Name: <Delete an Order>
Description: <This use case allows to delete a customer’s order>
Actors: <Primary actor: Admin>

Begin

MS /* Main Scenario */

Delete an Order

1. <Admin><Request><The admin selects the order to be deleted from the orders list> [<Order ID>].
2. <System><DataPersist><The system deletes the order> [<Order ID>].

End

ES /* Error Scenario */

Begin<Orders list is empty, begin at Num '2' in the *main scenario*>
2.1 <System><Response><The system displays the message “Orders list is empty”>
/* The scenario restarts at point 1. */
End

D. NFR - Non-Functional Requirements

Non-functional requirements define some quality requirements such as: performance, usability and security. The mobile app interface must be easy to use, simple and clear. Moreover, the web app must be compatible with any operating system, easy to use, understandable; with a “good response time”. It is also evident that RestoSys (including web and mobile applications) requires the presence of an internet connection.

E. PRC - Project Requirements and Constraints

The PRC address types of constraints surrounding the software development project, such as the competing demands of time, cost, and scope limitations on the project resources needed; dependencies on other projects, data storage space, etc.

2 THE MEASUREMENT STRATEGY PHASE

2. 1 Measurement Purpose

The purpose of the measurement is to estimate the development efforts for the mobile app and the web app separately, based on the size of the RestoSys software according to its Requirements, as specified in Chapter 1 of this case study.

2. 2 Measurement Scope

The scope is all functionality as specified in the functional requirements for the software as provided in section 1.4 A. The RestoSys software is in the application layer. The Requirements (REQ) Measurement Purpose requires a decomposition of its software in a mobile app component and a web app component. Hence, because of the Measurement Purpose, a measurement scope for each of both components is defined.

2. 3 Identification of Functional Users

The human functional users of the RestoSys are the employee (admin and waiter).

- The Admin: is the manager of the application. The Admin is entitled to manage the entire RestoSys and specify users and their individual rights.
- The Waiter: is responsible for adding or modifying the customers' orders.

Note that the DB Server is not an actor (functional user) of RestoSys, as the RestoSys requirements do not require data to be stored or retrieved with help of other software (i.e. another functional user). Rather it is merely a storage device, i.e. the physical implementation of the web app's persistent storage.

Figure 4 presents the contextual diagram for the RestoSys showing all its functional users.

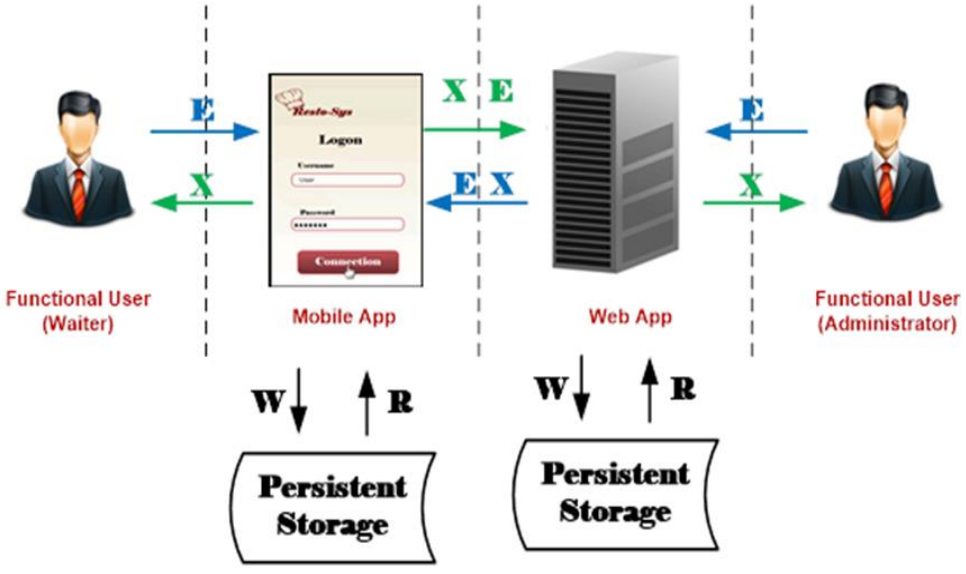

Figure 4 Contextual diagram

Figure 5 presents an example of data exchanges between mobile and web applications for REQ1 (Login for mobile app) and REQ3 (Login for web app). For example, login for mobile app is triggered by an Entry from the waiter which consists of the parameters (user name and password). The measurement details are provided in Section 4.1.

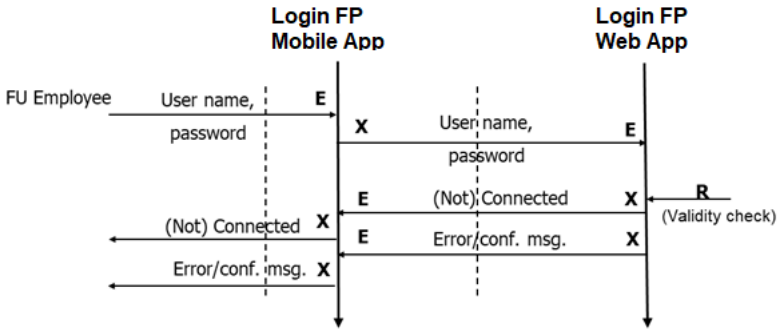

Figure 5 Data exchanges between Client (Mobile app) and Server (Web app) components For REQ1 and REQ3

2. 4 Other Measurement Strategy Parameters

A. Level of Granularity

The REQ of the RestoSys are at two levels of granularity. The first level of granularity is that of the Use Case global diagram (Figure 2). At this level, the data movements cannot be observed. For such cases, a COSMIC approximation method can be applied (COSMIC 2015b). However, since each use case identified in the first level can be detailed in the second level (such as Maintain Data which is detailed in Figure 3), the required level to apply COSMIC is where use cases are detailed using their textual description. More specifically, actions in a use case can be associated with the COSMIC data movement, as their level of granularity is the standard level, the 'functional process level of granularity'.

B. Decomposition

Figure 6 presents the decomposition for the RestoSys. The Mobile App interacts with the Web App via eXit/Entry data movements. The web app can require data to be stored or retrieved to/from persistent storage. Here, the architecture used to implement this system is 2-tier architecture. The two major components are: the Mobile app and the Web app.

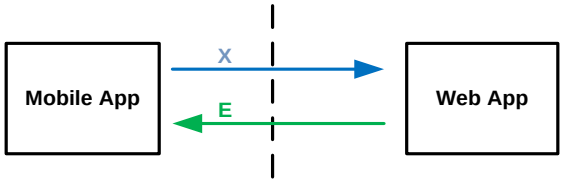

Figure 6 Decomposition

3 THE MAPPING PHASE

3. 1 Identification of the Functional Processes and the Software Triggering Events

From the REQ, the following triggering events are identified. For each functional process, there is only one triggering event (see Table 2). It is to be noted that when a functional process is required to obtain/save some data from/to another piece of software, it is the case of client/server' relationship. The REQ of the client component "Mobile App" can identify the server component "Web App" as one of its functional users, and vice versa. In addition, there is no standard mapping from a Use Case to a COSMIC Functional Process. Some Use Cases are mapped to several Functional Processes (e.g., the UC "Add an Order" is mapped to FP 2 and FP3). Other use cases are mapped to just one FP (e.g., the UC "Add an Employee" is mapped to FP8).

Table 2 Triggering Event Identification

REQ	Functional Processes	Triggering Events
REQ 1: Login "Mobile App"	FP 1: Login	The employee (waiter) needs to access to the login form
REQ 2: Maintain Order	FP 2: Add order's data	The waiter requires to add a new order
	FP 3: Create an order	The web app needs to create a new order
	FP 4: Modify order's data	The waiter requires to modify an existing order
	FP 5: Retrieve selected table data	The mobile app requires to retrieve the selected table data
	FP 6: Save modified data	The mobile app requires to save the modified data
REQ3: Login "Web app"	FP 7: Login	The employee (admin and waiter) needs to access to the login form (FP1 sends the login data (waiter data) to the web application for verification)
REQ 4: Maintain an Employee	FP 8: Add an employee	The admin requires to add an Employee (a waiter)
	FP 9: View the employees list	The admin needs to view the Employees list
	FP 10: View an employee data	The admin requires to view an Employee data
	FP 11: Modify an employee data	The admin requires to modify an Employee data
	FP 12: Delete an employee	The admin requires to delete an Employee
REQ 5: Maintain Item	FP 13: Add an Item	The admin requires to add a new item
	FP 14: View the items list	The admin requires to view the items list
	FP 15: View an Item data	The admin requires to view the item data
	FP 16: Modify an Item	The admin asks to modify an item
	FP 17: Delete an Item	The admin requires to delete an item
REQ 6: Maintain Item Family	FP 18: Add an Item Family	The admin requires to add a new item family
	FP 19: View Item Families List	The admin requires to view the item families list
	FP 20: View Item Family Data	The admin requires to view an item family data
	FP 21: Modify an Item Family	The admin asks to modify an item family
	FP 22: Delete an Item Family	The admin requires to delete an item family
REQ 7: Maintain Table	FP 23: Add a Table	The admin requires to add a table
	FP 24: View Tables List	The admin requires to view the tables list
	FP 25: View a Table Data	The admin requires to view a table Data

	FP 26: Modify Table Data	The admin asks to modify a table data
	FP 27: Delete a Table	The admin requires to delete a table
REQ 8: View the List of Orders	FP 28: View the List of Orders	The admin requires to view the list of orders
REQ 9: Delete an Order	FP 29: Delete an Order	The admin requires to delete an order

3. 2 Identification of Objects of Interest, Data Groups, and Data Attributes

From the REQ, seven objects of interest are identified. These are listed in Table 3 with their data groups and data attributes.

Table 3 List of Objects of Interest, Data Groups, and Data attributes

REQ	Objects of Interest	Data Groups	Data attributes	Example
REQ 1 REQ 3	Employee	Employee ID	username, password	username: alx84 password: 123
		UserState	userstate (Connected/Not Connected)	userstate: connected
		Employee Data	EmpNumber username, password, name, phone_number, address, job	EmpNumber: W2 username: alx84 password: 123 name: Alex phone_number: 22234567 address: Sfax, Tunisia job: waiter
REQ 2 REQ 7	Table	Table ID	table number	table number: 1
		Table Data	table number, state, capacity	table number: 1 state: occupied capacity: 2
	Set of Tables	Unoccupied Tables	table number, state, capacity	table number: 2 state: unoccupied capacity: 4
		All Table Data	table number, state, capacity	table number: 3 state: unoccupied capacity: 5
REQ 2 REQ 6	Item family	Item family Data	item family number, designation	item family number: 1 designation: Juice
				item family number: 2 designation: soda
				item family number: 3 designation: Cheap meals
REQ 2	Item	Item Data	item number,	item number: 1

REQ 5			item family number designation, image, price, quantity cheap price	item family number: 1 designation: orange juice image: orange.jpg price: 1 D quantity: 30 cheap price: 0.5D
				item number: 2 item family number: 2 designation: coca cola image: coca.jpg price: 2 D quantity: 100 cheap price: 1D
		Item ID	item number	item number: 1
REQ 2 REQ 7 REQ 8 REQ 9	Order	Order ID	order number	order number: 20
		Order Data	EmpNumber order number, table number, order state (active/closed), date, time	EmpNumber: W2 order number: 20 table number: 1 order state: active date: 05/07/2016 time: 8:00 pm
		Table ID	table number	table number: 1
REQ 2 REQ 6	Order Item	Order Items	order number, item number, quantity, comment	order number: 20 item number: 1 quantity: 3 comment: none item number: 2 quantity: 1 comment: with ice
		Item family ID	item family number	item family number: 1
From REQ 1 to REQ 9	Messages	E/C message	Message description	Message description: “Orders list is empty”

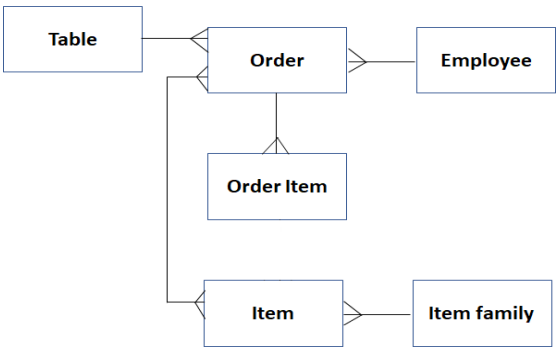

Figure 7 Data Model Diagram

Figure 7 presents the data model diagram for the RestoSys. Symbol **|** represents a one-and-only-one relationship. For instance, an item belongs to a single one item family. The crow’s foot symbol represents a one-or-many relationships. For instance, an employee manages multiple customers’ orders.

4 THE MEASUREMENT PHASE

4. 1 Functional Size Measured from REQ - Natural Language

A. For Mobile App

In this section, we give a detailed measurement of the functional size of the mobile app.

FP1: Login "Mobile App" **Triggering event: The employee (waiter) access to the login form**					
Functional User	**Sub-Process Description**	**Data Group**	**Objects of interest**	**Data Movement Type**	**CFP**
Employee	Enter username and password	Employee ID	Employee	E	1
Web app	The mobile app provides waiter data to the web app	Employee ID	Employee	X	1
Web app	The mobile app receives user state (Connected/Not connected)	UserState	Employee	E	1
Employee	The mobile app displays the userstate (Connected/Not connected)	UserState	Employee	X	1
Web app	The mobile app receives the Error/Confirmation message	E/C Message	Messages	E	1
Employee	The mobile app displays Error/Confirmation messages from the web application	E/C Message	Messages	X	1
					Total size = 6 CFP

FP 2: Add Order's Data **Triggering event: The employee (waiter) adds a new order**					
Functional User	**Sub-Process Description**	**Data Group**	**Objects of interest**	**Data Movement Type**	**CFP**
Employee	The waiter enters the table where the customer wants to be installed	Table ID	Order	E	1
Web app	The mobile app sends the selected table to the web app and then the web app creates a new order	Table ID	Order	X	1
Web app	The mobile app receives the item families list	Item family data	Item family	E	1
Employee	The mobile app displays the item families list to the waiter	Item family data	Item family	X	1

Employee	The waiter enters the item family based on the items requested by the customer	Item family ID	Order Item	E	1
Web app	The mobile app sends the selected item family to the web app	Item family ID	Order Item	X	1
Web app	The mobile app receives the item list of the selected family	Item Data	Item	E	1
Employee	The mobile app displays the list of items to the waiter	Item Data	Item	X	1
Employee	The waiter enters the items required by the customer and specifies for each item the recommended quantity and the customer comments	Order Items	Order item	E	1
Web app	The mobile app sends the selected items, recommended quantities and customer's comments to the web app	Order Items	Order item	X	1
Web app	The mobile app receives Error/Confirmation messages from the web app	E/C Message	Messages	E	1
Employee	The mobile app displays Error/Confirmation messages to the waiter	E/C Message	Messages	X	1
					Total size = 12 CFP

FP 4: Modify Order's Data **Triggering event: The employee (waiter) modifies an existing order**					
Functional User	**Sub-Process Description**	**Data Group**	**Objects of interest**	**Data Movement Type**	**CFP**
Employee	The waiter enters the table where the customer is installed	Table ID	Table	E	1
Web app	The mobile app sends the selected table to the web app	Table ID	Table	X	1
Web app	The mobile app receives the list of items previously selected by the customer with their quantities and comments	Order Items	Order item	E	1

FP 4: Modify Order's Data Triggering event: The employee (waiter) modifies an existing order					
Functional User	**Sub-Process Description**	**Data Group**	**Objects of interest**	**Data Movement Type**	**CFP**
Employee	The mobile app displays the items previously selected by the customer with their quantities and comments to the waiter	Order Items	Order item	X	1
Employee	The waiter deletes the existing item, modifies items' quantities and/or modifies items' comments	Order Items	Order item	E	1
Web app	The mobile app sends the modified items to the web app	Order Items	Order item	X	1
Web app	The mobile app receives Error/Confirmation messages from the web app	E/C Message	Messages	E	1
Employee	The mobile app displays Error/Confirmation messages to the waiter	E/C Message	Messages	X	1
					Total size = 8 CFP

The total functional size of the mobile app is the sum of the sizes of its three functional processes, that is:

6 CFP + 12 CFP + 8 CFP = 26 CFP

B. For Web app

The web app includes 27 functional processes. In this section, we give a detailed measurement of the functional size of the web app.

FP3: Create an order Triggering event: The web app receives the selected items					
Functional User	**Sub-Process Description**	**Data Group**	**Objects of interest**	**Data Movement Type**	**CFP**
Mobile App	The web app receives the selected items, quantities and customer comments	Order Items	Order Items	E	1
	The web app creates the new order	Order Data	Order	W	1
Mobile App	The web app sends Error/Confirmation messages to the mobile app	E/C Message	Message	X	1
					Total size = 3 CFP

FP5: Retrieve selected table data **Triggering event: The mobile app sends the selected table**					
Functional User	**Sub-Process Description**	**Data Group**	**Objects of interest**	**Data Movement Type**	**CFP**
Mobile App	The web app receives the selected table	Table ID	Table	E	1
	The web app retrieves the list of item families selected by the customer	Order Items	Order Item	R	1
Mobile App	The web app sends the list of item families selected by the customer to the mobile app	Order Items	Order Item	X	1
	The web app retrieves the list of items selected by the customer	Item data	Item	R	1
	The web app sends the list of items selected by the customer	Item data	Item	X	1
					Total size = 5 CFP

FP 6: Save modified data **Triggering event: The mobile app sends the modified data**					
Functional User	**Sub-Process Description**	**Data Group**	**Objects of interest**	**Data Movement Type**	**CFP**
Mobile App	The web app receives the modified items	Order Items	Order	E	1
	The web app updates the order data	Order Data	Order	W	1
Mobile App	The web app displays the Error/Confirmation message	E/C Messages	Message	X	1
					Total size = 3 CFP

FP7: Login **Triggering event: The employee (admin) access to the login form**					
Functional User	**Sub-Process Description**	**Data Group**	**Objects of interest**	**Data Movement Type**	**CFP**
Employee	Enter username and password	Employee ID	Employee	E	1
	The web app checks the validity of admin data	Employee ID	Employee	R	1
Employee	The web app displays the user state (Connected/ Not connected)	UserState	Employee	X	1

Employee	The web app displays Error/Confirmation messages	E/C Message	Messages	X	1
					Total size = 4 CFP

FP8: Add an Employee **Triggering event: The admin adds an employee**					
Functional User	**Sub-Process Description**	**Data Group**	**Objects of interest**	**Data Movement Type**	**CFP**
Admin	The admin enters the employee data	Employee Data	Employee	E	1
	The web app retrieves the employee's data to verify if the employee exists	Employee Data	Employee	R	1
	The web app adds the new employee	Employee Data	Employee	W	1
Admin	the web app displays Error/Confirmation messages	E/C Message	Messages	X	1
					Total size = 4 CFP

FP9: View the Employees list **Triggering event: The admin views the users list**					
Functional User	**Sub-Process Description**	**Data Group**	**Objects of interest**	**Data Movement Type**	**CFP**
Admin	The admin selects to view the employees list	Employee Data	Employee	E	1
	The web app retrieves the employees list	Employee Data	Employee	R	1
Admin	The web app displays the employees list	Employee Data	Waiter	X	1
Admin	The web app displays Error/Confirmation messages	E/C Message	Messages	X	1
					Total size = 4 CFP

FP10: View an Employee data **Triggering event: The admin asks for an employee data**					
Functional User	**Sub-Process Description**	**Data Group**	**Objects of interest**	**Data Movement Type**	**CFP**
Admin	The admin selects the desired employee	Employee Data	Employee	E	1
	The web app retrieves user data from the Database	Employee Data	Employee	R	1

Admin	The web app displays the data of the selected waiter	Employee Data	Employee	X	1
					Total size = 3 CFP

FP11: Modify an Employee Data **Triggering event: The admin modifies employee data**					
Functional User	**Sub-Process Description**	**Data Group**	**Objects of interest**	**Data Movement Type**	**CFP**
Admin	The admin modifies employee data that can be changed	Employee Data	Employee	E	1
	The web app saves the change	Employee Data	Employee	W	1
Admin	The web app displays Error/Confirmation messages	E/C Message	Messages	X	1
					Total size = 3 CFP

FP12: Delete an Employee **Triggering event: The admin deletes an employee**					
Functional User	**Sub-Process Description**	**Data Group**	**Objects of interest**	**Data Movement Type**	**CFP**
Admin	The admin enters the employee to be deleted	Employee ID	Employee	E	1
	The web app deletes the selected employee	Employee ID	Employee	W	1
Admin	The web app displays Error/Confirmation messages	E/C Message	Messages	X	1
					Total size = 3 CFP

FP13: Add an Item **Triggering event: The admin adds a new item**					
Functional User	**Sub-Process Description**	**Data Group**	**Objects of interest**	**Data Movement Type**	**CFP**
Admin	The admin enters the data of the item	Item Data	Item	E	1
	The web app retrieves the items data to verify if the admin adds an existing item	Item Data	Item	R	1
	The web app adds the new item	Item Data	Item	W	1
Admin	The web app displays Error/Confirmation messages	E/C Message	Messages	X	1
					Total size = 4 CFP

FP14: View the items list **Triggering event: The employees view the items list**					
Functional User	**Sub-Process Description**	**Data Group**	**Objects of interest**	**Data Movement Type**	**CFP**
Mobile app	The web app receives the waiter request	Item family ID	Order Item	E	1
Admin	The admin selects to view the items list	Item Data	Item	E	1
	The web app retrieves the items list	Item Data	Item	R	1
Admin/ Mobile app	The web app displays/sends the items list	Item Data	Item	X	1
Admin Mobile app	The web app displays/sends Error/Confirmation messages	E/C Message	Messages	X	1
					Total size = 5 CFP

FP15: View an Item data **Triggering event: The admin views an item data**					
Functional User	**Sub-Process Description**	**Data Group**	**Objects of interest**	**Data Movement Type**	**CFP**
Admin	The admin enters the desired item	Item ID	Item	E	1
	The web app retrieves the item data from the Database	Item Data	Item	R	1
Admin	The web app displays the data of the selected item	Item Data	Item	X	1
					Total size = 3 CFP

FP16: Modify an item **Triggering event: The admin modifies an item**					
Functional User	**Sub-Process Description**	**Data Group**	**Objects of interest**	**Data Movement Type**	**CFP**
Admin	The admin modifies the desired fields (name, price, quantity, image, etc.) and asks the web app to update the data of the item	Item Data	Item	E	1
	The web app saves the change	Item Data	Item	W	1

Admin	The web app displays Error/Confirmation messages	E/C Message	Messages	X	1
					Total size = 3 CFP

FP17: Delete an Item **Triggering event: The admin deletes an item**					
Functional User	**Sub-Process Description**	**Data Group**	**Objects of interest**	**Data Movement Type**	**CFP**
Admin	The admin selects the item to be deleted	Item ID	Item	E	1
	The web app deletes the selected item	Item ID	Item	W	1
Admin	The web app displays Error/Confirmation messages	E/C Message	Messages	X	1
					Total size = 3 CFP

FP18: Add an item family **Triggering event: The Admin adds a new item family**					
Functional User	**Sub-Process Description**	**Data Group**	**Objects of interest**	**Data Movement Type**	**CFP**
Admin	The admin enters the data of the item family	Item Family Data	Item Family	E	1
	The web app retrieves the item families' data to verify if the admin adds an existing item family	Item Family Data	Item Family	R	1
	The web app adds the new item family	Item Family Data	Item Family	W	1
Admin	The web app displays Error/Confirmation messages	E/C Message	Messages	X	1
					Total size = 4 CFP

FP19: View Item families list **Triggering event: The employees view the item families list**					
Functional User	**Sub-Process Description**	**Data Group**	**Objects of interest**	**Data Movement Type**	**CFP**
Mobile app	The web app receives the waiter requests	Table ID	Order	E	1
Admin	The admin selects to view the item families list	Item Family Data	Item Family	E	1
	The web app retrieves the item families list	Item Family Data	Item Family	R	1

Admin/Mobile app	The web app displays/sends the item families list	Item Family Data	Item Family	X	1
Admin/Mobile app	The web app displays/sends Error/Confirmation messages	E/C Message	Messages	X	1
					Total size = 5 CFP

FP20: View Item family data **Triggering event: The admin views an item family data**					
Functional User	**Sub-Process Description**	**Data Group**	**Objects of interest**	**Data Movement Type**	**CFP**
Admin	The admin enters the desired Item Family	Item Family ID	Item Family	E	1
	The web app retrieves the Item Family data from the Database	Item Family Data	Item Family	R	1
Admin	The web app displays the data of the selected Item Family to the admin	Item Family Data	Item Family	X	1
					Total size = 3 CFP

FP21: Modify an item family **Triggering event: The admin modifies an item family**					
Functional User	**Sub-Process Description**	**Data Group**	**Objects of interest**	**Data Movement Type**	**CFP**
Admin	The admin enters modified fields (name, image, color)	Item family Data	Item Family	E	1
	The web app saves the change	Item family Data	Item Family	W	1
Admin	The web app displays Error/Confirmation messages	E/C Message	Messages	X	1
					Total size = 3 CFP

FP22: Delete an item family **Triggering event: The admin deletes an item family**					
Functional User	**Sub-Process Description**	**Data Group**	**Objects of interest**	**Data Movement Type**	**CFP**
Admin	The admin enters the family items to be deleted	Item family ID	Item Family	E	1
	The web app deletes the selected family item	Item family ID	Item Family	W	1

Admin	The web app displays Error/Confirmation messages	E/C Message	Messages	X	1
					Total size = 3 CFP

FP23: Add a table **Triggering event: The admin adds a table**					
Functional User	**Sub-Process Description**	**Data Group**	**Objects of interest**	**Data Movement Type**	**CFP**
Admin	The admin enters the table data	Table Data	Table	E	1
	The system checks if the table already exists	Table Data	Table	R	1
	The web app adds the new table	Table Data	Table	W	1
Admin	The web app displays Error/Confirmation messages	E/C Message	Messages	X	1
					Total size = 4 CFP

FP24: View Tables list **Triggering event: The employee (admin or waiter) views the tables list**					
Functional User	**Sub-Process Description**	**Data Group**	**Objects of interest**	**Data Movement Type**	**CFP**
Mobile app	The mobile app selects to view the tables list	Unoccupied Tables	Set of Tables	E	1
Admin	The admin selects to view the tables list	All table Data	Set of Tables	E	1
	The web app retrieves the Tables list	Table Data	Table	R	1
Admin / Mobile app	The system displays / sends the Tables list	Table Data	Table	X	1
Mobile app / Admin	The web app displays Error/Confirmation messages	E/C Message	Messages	X	1
					Total size = 5 CFP

FP25: View a table data **Triggering event: The admin views table data**					
Functional User	**Sub-Process Description**	**Data Group**	**Objects of interest**	**Data Movement Type**	**CFP**
Admin	The admin enters the desired table	Table ID	Table	E	1
	The web app retrieves the Table data from the database	Table Data	Table	R	1

Admin	The web app displays the data of the selected table to the admin	Table Data	Table	X	1
					Total size = 3 CFP

FP26: Modify table data **Triggering event: The admin modifies table data**					
Functional User	**Sub-Process Description**	**Data Group**	**Objects of interest**	**Data Movement Type**	**CFP**
Admin	The admin enters modifies\d table data	Table Data	Table	E	1
	The web app saves the change	Table Data	Table	W	1
Admin	The web app displays Error/Confirmation messages	E/C Message	Messages	X	1
					Total size = 3 CFP

FP27: Delete a table **Triggering event: The admin deletes a table**					
Functional User	**Sub-Process Description**	**Data Group**	**Objects of interest**	**Data Movement Type**	**CFP**
Admin	The admin enters the table to be deleted	Table ID	Table	E	1
	The web app deletes the selected table	Table ID	Table	W	1
Admin	The web app displays Error/Confirmation messages	E/C Message	Messages	X	1
					Total size = 3 CFP

FP28: View the List of Orders **Triggering event: The admin views the list of orders**					
Functional User	**Sub-Process Description**	**Data Group**	**Objects of interest**	**Data Movement Type**	**CFP**
Admin	The admin selects to view the orders list	Order Data	Order	E	1
	The web app retrieves the orders data	Order Data	Order	R	1
Admin	The web app displays the orders list	Order Data	Order	X	1
Admin	The web app displays Error/Confirmation messages	E/C Message	Messages	X	1
					Total size = 4 CFP

FP29: Delete an order Triggering event: The admin deletes an order					
Functional User	**Sub-Process Description**	**Data Group**	**Objects of interest**	**Data Movement Type**	**CFP**
Admin	The admin selects the order to be deleted	Order ID	Order	E	1
	The web app deletes the selected order	Order ID	Order	W	1
Admin	The web app displays Error/Confirmation messages	E/C Message	Messages	X	1
					Total size = 3 CFP

The total functional size of the web app is the sum of the sizes of its 26 functional processes, that is:

3 CFP + 5 CFP + 3 CFP + 4 CFP + 4 CFP + 4 CFP + 3 CFP + 3 CFP + 3 CFP + 4 CFP + 5 CFP + 3 CFP + 3 CFP + 3 CFP + 4 CFP + 5 CFP + 3 CFP + 3 CFP + 3 CFP + 4 CFP + 5 CFP + 3 CFP + 3 CFP + 3 CFP + 4 CFP + 3 CFP = 93 CFP

C. For the Whole System

The total size of RestoSys is then equal to the sum of functional sizes of mobile and web apps (26 CFP + 93 CFP) = 119 CFP.

4. 2 Functional Size Measured from REQ – US and UML UC Textual Description

A. Using User Stories Description

The functional size of RestoSys using User stories description are presented in Table 4. A large user story usually includes a number of Functional Processes. To quantify the size of RestoSys through user stories, Table 4 identifies the set of functional processes for each user story with their size measurement.

Table 4: Sizing RestoSys through US description with their corresponding functional processes

User Story (US)	Functional Processes	CFP	
US1 : "Login" Mobile app	FP 1: Login		**FS(US1) = 6 CFP**
US2: Maintain Order			**FS(US2) = 31 CFP**
US2.1 : Add an Order	FP 2: Add order's data	FS(FP2) = 12 CFP	**FS(US2.1) = 15CFP**
	FP 3: Create an order	FS(FP3) = 3 CFP	
US2.2 : Modify an Order	FP 4: Modify order's data	FS(FP4) = 8 CFP	**FS(US2.2) = 16CFP**
	FP 5: Retrieve selected table data	FS(FP5) = 5 CFP	
	FP 6: Save modified data	FS(FP6) = 3 CFP	
US3 : "Login" Web app	FP 7: Login		**FS(US3) = 4 CFP**

US4: Maintain an Employee		**FS(US4) = 17 CFP**
US4.1 : Add an employee	FP 8: Add an employee	FS(US4.1) = 4 CFP
US4.2 : View the Employee List	FP 9: View the employees list	FS(US4.2) = 4 CFP
US4.3 : View an Employee data	FP 10: View an employee data	FS(US4.3)= 3CFP
US4.4 : Modify an Employee Data	FP 11: Modify an employee data	FS(US4.4)= 3CFP
US4.5 : Delete an employee	FP 12: Delete an employee	FS(US4.5)= 3CFP
US5 : Maintain Item		**FS(US5)= 18 CFP**
US5.1 : Add an Item	FP 13: Add an Item	FS(US5.1)= 4CFP
US5.2 : View the Items List	FP 14: View the items list	FS(US5.2)= 5CFP
US5.3 : View Item data	FP 15: View an Item data	FS(US5.3)= 3CFP
US5.4 : Modify an Item	FP 16: Modify an Item	FS(US5.4)= 3CFP
US5.5 : Delete an Item	FP 17: Delete an Item	FS(US5.5)= 3CFP
US6 : Maintain Item Family		**FS(US6)= 18 CFP**
US6.1 : Add an Item family	FP 18: Add an Item Family	FS(US6.1)= 4CFP
US6.2 : View the Item families list	FP 19: View Item Families List	FS(US6.2)= 5CFP
US6.3 : View Item family data	FP 20: View Item Family Data	FS(US6.3)= 3 CFP
US6.4 : Modify an Item family	FP 21: Modify an Item Family	FS(US6.4)= 3CFP
US6.5 : Delete an Item family	FP 22: Delete an Item Family	FS(US6.5)= 3CFP
US7 : Maintain Table		**FS(US7)= 18 CFP**
US7.1 : Add a table	FP 23: Add a Table	FS(US7.1)= 4CFP
US7.2 : View the tables List	FP 24: View Tables List	FS(US7.2)= 5CFP
US7.3 : View table data	FP 25: View a Table Data	FS(US7.3)= 3CFP
US7.4 : Modify table data	FP 26: Modify Table Data	FS(US7.4)= 3CFP
US7.5 : Delete a table	FP 27: Delete a Table	FS(US7.5)= 3CFP
US8: View the list of orders	FP 28: View the List of Orders	**FS(US8)= 4CFP**
US9: Delete an order	FP 29: Delete an Order	**FS(US9)= 3CFP**
Σ size (US1, US2, US3, US4, US5, US6, US7, US8, and US9) = 119 CFP		

B. Using Action-Type

Table 4 presents the mapping of action-type in use case description with COSMIC data movements. Note that each use case can be associated with more than one functional process. As an example, the “Login” use case is represented by “FP1: Login”. Whereas, the use case “Add an order” is associated with FP2 and FP3.

Table 5 Equivalence between action-type in use case description and COSMIC concepts in terms of Data movements

Actions-types in Use case description	Actions-types description	COSMIC concepts	CFP
Action-type = Expletive	An action that does not lead to an exchange of data	Not applied	0 CFP
Action-type = Request	An action representing the act of asking for something, it is directed by an actor	An Entry data movement from the Functional user to the software to be measured	1 CFP
Action-type = Response	An answer or a reply sent after a Request, it is directed by the system	An eXit data movement from the software to be measured to the Functional user	1 CFP
Action-type = DataRecovery	An action that allows the retrieving of data	A Read data movement from the persistent storage to the software to be measured	1 CFP
Action-type = DataPersist	An action allowing the recording of data	A Write data movement from the software to be measured to the persistent storage	1 CFP

a. For mobile app

Table 6 presents the measurement results of the functional size of the mobile app based on the Action-Type.

Table 6 Measurement Results-Using Action-Type (mobile app)

Functional Requirements	Functional Processes	Sub-Process Description	Action Type	CFP
REQ 1: Login “Mobile App”	**FP 1: Login**	Enter username and password	Request	1
		The mobile app provides waiter data to the web app	Response	1
		The mobile app receives userstate (Connected/Not connected)	Request	1
		The mobile app displays the userstate (Connected/Not connected)	Response	1
		The mobile app receives the Error/Confirmation message	Request	1

		The mobile app displays Error/Confirmation messages from the web application	Response	1
				FS(FP 1: Login) = 6 CFP
	FP 2: Add order's data	The waiter enters the table where the customer wants to be installed	Request	1
		The mobile app sends the selected table to the web app and then the web app creates a new order	Response	1
		The mobile app receives the item families list	Request	1
		The mobile app displays the item families list to the waiter	Response	1
		The waiter enters the item family based on the items requested by the customer	Request	1
		The mobile app sends the selected item family to the web app	Response	1
		The mobile app receives the item list of the selected family	Request	1
		The mobile app displays the list of items to the waiter	Response	1
		The waiter enters the items required by the customer and specifies for each item the recommended quantity and the customer comments	Request	1
		The mobile app sends the selected items, recommended quantities and customer's comments to the web app	Response	1
		The mobile app receives Error/Confirmation messages from the web app	Request	1
		The mobile app displays Error/Confirmation messages to the waiter	Response	1
				FS(FP 2: Add order's data) = 12 CFP
	FP 4: Modify Order's Data	The waiter enters the table where the customer is installed	Request	1
		The mobile app sends the selected table to the web app	Response	1
		The mobile app receives the list of items previously selected by the customer with their quantities and comments to the mobile app	Request	1
		The mobile app displays the items previously selected by the customer with their quantities and comments to the waiter	Response	1

		The waiter deletes the existing item, modifies items' quantities and/or modifies items' comments	Request	1
		The mobile app sends the modified items to the web app	Response	1
		The mobile app receives Error/Confirmation messages from the web app	Request	1
		The mobile app displays Error/Confirmation messages to the waiter	Response	1
	FS(FP 4: Modify order's data) = 8 CFP			

The total functional size of the mobile app is the sum of the sizes of its three functional processes (FP1, FP2, and FP4), that is:

6 CFP + 12 CFP + 8 CFP = 26 CFP

b. For web app

Table 7 presents the measurement results of the functional size of the web app based on the Action-Type.

Table 7 Measurement Results - Using Action-Type (web app)

Functional Requirements	Functional Processes	Sub-Process Description	Action Type	CFP
	FP3: Create an order	The web app receives the selected items, quantities and customer comments	Request	**1**
		The web app creates the new order	DataPersist	1
		The web app sends Error/Confirmation messages to the mobile app	Response	1
	FS(FP3: Create an order) = 3 CFP			
	FP 5: Retrieve selected table data	The web app receives the selected table from the mobile app	Request	1
		The web app retrieves the list of item families selected by the customer	DataRecovery	1
		The web app sends the list of item families selected by the customer to the mobile app	Response	1
		The web app retrieves the list of items selected by the customer	DataRecovery	1
		The web app sends the list of items selected by the customer to the mobile app	Response	1
	FS(FP 5: Retrieve selected table data) = 5 CFP			
		The web app receives the modified items	Request	**1**

	FP 6: Save modified data	The web app updates the order data	DataPersist	1
		the web app displays Error/Confirmation messages	Response	1
			FS(FP 6: Save modified data) = 3 CFP	
REQ 3: Login "Web app"	**FP7: Login**	Enter username and password	Request	1
		The web app checks the validity of admin data	DataReovery	1
		The web app displays the user state (Connected/Not connected)	Response	1
		The web app displays Error/Confirmation messages	Response	1
			FS(FP7: Login) = 4 CFP	
REQ 4: Maintain an Employee	**FP8: Add an Employee**	The Admin enters the employee data	Request	1
		The web app retrieves the employee's data to verify if the employee exists	DataRecovery	1
		The web app adds the new employee	DataPersist	1
		The web app displays Error/Confirmation message	Response	1
			FS(FP8: Add an Employee) = 4 CFP	
	FP 9: View the Employees list	The admin selects to view the employees list	Request	1
		The web app retrieves the employees list	DataRecovery	1
		The web app displays the employees list	Response	1
		The web app displays Error/Confirmation messages	Response	1
			FS(FP 9: View the Employees list) = 4 CFP	
	FP 10: View an Employee Data	The admin selects the desired employee	Request	1
		The web app retrieves user data from the Database	Data-Recovery	1
		The web app displays the data of the selected waiter	Response	1
			FS(FP 10: View an Employee Data) = 3CFP	
	FP 11: Modify an Employee Data	The admin modifies employee data that can be changed	Request	1
		The web app saves the change	DataPersist	1
		The web app displays Error/Confirmation messages	Response	1
			FS(FP 11: Modify an Employee Data) = 3 CFP	
	FP 12: Delete an Employee	The admin enters the employee to be deleted	Request	1
		The web app deletes the selected employee	DataPersist	1
		The web app displays Error/Confirmation messages	Response	1

	FS(FP 12: Delete an Employee) = 3 CFP			
REQ 5: Maintain an Item	**FP 13: Add an Item**	The admin enters the data of the item	Request	1
		The web app retrieves the items data to verify if the admin adds an existing item	Data-Recovery	1
		The web app adds the new item	DataPersist	1
		The web app displays Error/Confirmation messages	Response	1
	FS(FP 13: Add an Item) = 4 CFP			
	FP 14: View the items list	The web app receives the waiter request	Request	1
		The admin selects to view the items list	Request	1
		The web app retrieves the items list	Data-Recovery	1
		The web app displays the items list	Response	1
		The web app displays Error/Confirmation messages	Response	1
	FS(FP14: View the items list) = 5 CFP			
	FP 15: View an Item data	The admin enters the desired item	Request	1
		The web app retrieves the item data from the Database	Data-Recovery	1
		The web app displays the data of the selected item	Response	1
	FS(FP 15: View an Item data) = 3 CFP			
	FP 16: Modify an item	The admin modifies the desired fields (name, price, quantity, image, etc.) and asks the web app to update the data of the item	Request	1
		The web app saves the change	DataPersist	1
		The web app displays Error/Confirmation messages	Response	1
	FS(FP16: Modify an item) = 3 CFP			
	FP 17: Delete an Item	The admin selects the item to be deleted	Request	1
		The web app deletes the selected item	DataPersist	1
		The web app displays Error/Confirmation messages	Response	1
	FS(FP 17: Delete an Item) = 3 CFP			
REQ 6: Maintain Item family	**FP 18: Add an item family**	The admin enters the data of the item family	Request	1
		The web app retrieves the item families' data to verify if the admin adds an existing item family	Data-Recovery	1

		The web app adds the new item family	DataPersist	1
		The web app displays Error/Confirmation messages	Response	1
	FS(FP18: Add an item family) = 4 CFP			
	FP19: View Item families list	The web app receives the waiter requests	Request	1
		The admin selects to view the item families list	Request	1
		The web app retrieves the item families list	DataRecovery	1
		The web app displays the item families list	Response	1
		The web app displays Error/Confirmation messages	Response	1
	FS(FP19: View Item families list) = 5 CFP			
	FP20: View Item family data	The admin enters the desired Item Family	Request	1
		The web app retrieves the Item Family data from the Database	DataRecovery	1
		The web app displays the data of the selected Item Family to the admin	Response	1
	FS(FP20: View Item family data) = 3 CFP			
	FP21: Modify an item family	The admin enters modified fields (name, image, color)	Request	1
		The web app saves the change	DataPersist	1
		The web app displays Error/Confirmation messages	Response	1
	FS(FP21: Modify an item family) = 3 CFP			
	FP22: Delete an item family	The admin enters the family items to be deleted	Request	1
		The web app deletes the selected family item	DataPersist	1
		The web app displays Error/Confirmation messages	Response	1
	FS(FP22: Delete an item family) = 3 CFP			
REQ 7: Maintain Table	**FP23: Add a table**	The admin enters the table data	Request	1
		The system checks if the table already exists	DataRecovery	1
		The web app adds the new table	DataPersist	1
		The web app displays Error/Confirmation messages	Response	1
	FS(FP23: Add a table) = 4 CFP			
	FP24: View Tables list	The mobile app selects to view the tables list	Request	1
		The admin selects to view the tables list	Request	1
		The web app retrieves the Tables list	DataRecovery	1

		The system displays the Tables list	Response	1
		The web app displays Error/Confirmation messages	Response	1
	FS(FP24: View Tables list) = 5 CFP			
	FP25: View a table data	The admin enters the desired table	Request	1
		The web app retrieves the Table data from the database	DataRecovery	1
		The web app displays the data of the selected table to the admin	Response	1
	FS(FP25: View a table data) = 3 CFP			
	FP26: Modify table data	The admin enters modified table data	Request	1
		The web app saves the change	DataPersist	1
		The web app displays Error/Confirmation messages	Response	1
	FS(FP26: Modify table data) = 3 CFP			
	FP27: Delete a table	The admin enters the table to be deleted	Request	1
		The web app deletes the selected table	DataPersist	1
		The web app displays Error/Confirmation messages	Response	1
	FS(FP27: Delete a table) = 3 CFP			
REQ 8: View the List of Orders	**FP28: View the List of Orders**	The admin selects to view the orders list	Request	1
		The web app retrieves the orders data	DataRecovery	1
		The web app displays the orders list	Response	1
		The web app displays Error/Confirmation messages	Response	1
	FS(FP28: View the List of Orders) = 4 CFP			
REQ 9: Delete an order	**FP29: Delete an order**	The admin selects the order to be deleted	Request	1
		The web app deletes the selected order	DataPersist	1
		The web app displays Error/Confirmation messages	Response	1
	FS(FP29: Delete an order) = 3 CFP			

The total functional size of the web app is the sum of the sizes of its 26 functional processes, that is:

3 CFP + 5 CFP + 3 CFP + 4 CFP + 4 CFP + 4 CFP + 3 CFP + 3 CFP + 3 CFP + 4 CFP + 5 CFP + 3 CFP + 3 CFP + 3 CFP + 4 CFP + 5 CFP + 3 CFP + 3 CFP + 3 CFP + 4 CFP + 5 CFP + 3 CFP + 3 CFP + 3 CFP + 4 CFP + 3 CFP = 93 CFP

The total size of RestoSys is then equal to the sum of functional sizes of mobile and web apps (26 CFP + 93 CFP) = 119 CFP, which is the same functional size as measured by referring to the documented REQ in natural language.

REFERENCE

COSMIC (2017) The Common Software Measurement International Consortium. 2017. The COSMIC Functional Size Measurement Method, Version 4.0.2, Measurement Manual

COSMIC (2015a) Guideline on Non-Functional & Project Requirements : How to consider non-functional and project requirements in software project performance measurement, benchmarking and estimating.

COSMIC (2015b) Guideline for Early or Rapid COSMIC Functional Size Measurement by using approximation approaches.

(Haoues *et al.,* 2017a) M. Haoues, A. Sellami, and H. Ben-Abdallah, “Functional change impactanalysis in use cases: An approach based on COSMIC functional sizemeasurement,” Science of Computer Programming, Special Issue onAdvances in Software Measurement., 2017

(Haoues *et al.,* 2017b) M. Haoues, A. Sellami, and H. Ben-Abdallah. 2017. A Rapid Measurement Procedure for Sizing Web and Mobile Applications basedon COSMIC FSM Method. In Proceedings of 27th International Workshopon Software Measurement and 12th International Conference on SoftwareProcess and Product Measurement, Gothenburg, Sweden, October 25–27, 2017(IWSM/Mensura ’17), 9 pages.

Mhadhbi S (2013) Conception et Développement d’un Système de Gestion Restaurant Mobile.

APPENDIX A - STRUCTURED USE CASE DOCUMENTATION FORMAT USING ACTION-TYPE

Many individual proposals of textual descriptions are available to assist in documenting a Use Case (UC). We proposed an extension of the UC textual description with the “Type” of an action which can be: “Request”, “DataPersist”, “DataRecovery”, “Expletive”, or “Response” (Haoues *et al.*, 2017a). A “Request” corresponds to an action representing the act of asking for something, it is directed by an actor. A “Response” corresponds to an answer or a reply sent after a Request, it is directed by the system. A “DataPersist” action corresponds to an action allowing the recording of data. An “Expletive” action is used to get an action started but does not lead to an exchange of data, such as an action allowing data manipulation. “DataRecovery” action allows the retrieving of data.

The following documentation of a use case is provided in (Haoues *et al.*, 2017a).

Number:<unique ID assigned to a use case> **Name:**<unique name assigned to a use case> **Level:**<level of use case description> **Description:**<a summary of use case purpose> **Actors:**<Primary actor: actor that initiates the use case> <Secondary actor: actor that participate within the use case>
Pre-condition:<a list of conditions that must be true to initialize the use case> **Post-condition (success):**<state of the system if goal is achieved> **Post-condition (failure):**<state of the system if goal is abandoned> **Relationship** [**Include:**<use cases in relation with this use case by "include">, **Extend:**<use cases in relation with this use case by "extend">, **Super use case:**<list of subordinate use cases of this use case>, **Sub use case:** <list of all use cases that specialize this use case>] **Begin** **MS /* Main scenario */** <Steps of the scenario from trigger to goal> **Begin** <NumAction> [<Pre-condition>] <Actor|System><Type: Request, DataPersist, Expletive, Response, DataRecovery><Action Description> [<Int-Parameter>] [<Out-Parameter>] **End** **AS /* Alternative scenario */** **Begin**<Event, begin at Num "action number"> <NumAction> [<Pre-condition>] <Actor|System><Type: Request, DataPersist, Expletive, Response, DataRecovery><Action Description> [<Int-Parameter>] [<Out-Parameter>] The main scenario back to NUM **End** **ES /* Error scenario */** **Begin**<Event, begin at Num "action number"> <NumAction> [<Pre-condition>] <Actor|System><Type: Request, DataPersist, Expletive, Response, DataRecovery><Action Description> [<Int-Parameter>] [<Out-Parameter>] **End** **End Use Case**
Special requirements:<list of non-functional requirements>