REQUIREMENT ANALYSIS| GREEN SQA

Martín Gómez

Study case:

Client	Green SQA
User	Company members, service managers, and collaborators
Functional Requirements	**FR1)** Project management **FR2)** Project stages management **FR3)** Culminate a stage of a project **FR4)** Register capsules **FR5)** Capsule approbation **FR6)** Capsule publishing **FR7)** Search knowledge capsules
Problem Context	GreenSQA, is a technology company that works with projects of software quality assurance. As such it searches to, by using this program of knowledge storage based on capsules generated by collaborators, hold on to employees knowledge before it's passed to other employers. *Knowledge capsule: A text in which situations, elements or significant project data is stored*
Non-Functional Requirements	**NFR1)** Make the program taking into consideration that it is a first version, so some data is limited, like the number of projects (with a max of 10) and the number of capsules per stage (50).

Name and identifier	*[FR1: Project management]*

Summary	*The user is requested to input the name of the project, the name of the client, the planned dates for start and end of the project, the value of its budget, the number of managers from each side and their names and phone numbers. Then the project is created, and the user is asked whether he wants to register another project or not, until he says no or 10 projects are created.*		
Inputs	**Input name**	**Data type**	**Valid values condition**
	projectName	String	
	clientName	String	
	datePlanedBegin	String	
	datePlanedEnd	String	
	budget	Double	
	clientManagers	int	
	companyManagers	int	
	clientManagerNames	String[]	
	clientManagerPhones	String[]	
	companyManagerNames	String[]	
	companyManagerPhones	String[]	
Result	Project data sent to the controller class “Green”		
Outputs	**Output Name**	**Data Type**	**Format**
	projects_created	String	“The projects created are” +projectList

Name and identifier	*[FR2/FR3: Project stages management and culmination]*		
Summary	*Project execution is divided into 6 stages: 1. Beginning 2. Analysis 3. Design 4. Execution 5. Closing 6. Follow up and control. Each one of them has a beginning and end planned date, and a real one. Additionally the aprovation date of a stage is saved, and once a project is created the beginning stage is automatically activated. To calculate the planned dates the user must input the number of months for each stage. To culminate a stage the user must input the name of the project, then the current stage is culminated and the next is set as active.*		
Inputs	**Input name**	**Data type**	**Valid values condition**
	projectName	String	
	realDateBegin	String	
	monthsPerStage	Int[]	*Must be a whole number*

Result	The name and realDate end for the stage is sent to the stageCulmination method in “Green”, class that then searches for the project by it’s name, if it finds it, then the culminateStage on the Projects class is called receiving the readSate end (calculated as the current date)		
Outputs	**Output Name**	**Data Type**	**Format**
	stageCulmination Msg	String	“The stage number ” +stage + “was culminated successfully on ” +realDateEnd
	activeStageMsg	String	“The active stage is now the ” stage+1

Name and identifier	*[FR4:Register knowledge capsules]*		
Summary	*Capsules are generated in each of the project stages (50 max) for this version of the code. Each capsule has: A unique identifier, a description of the situation, a capsule type (ranging between technical, management, domain and experiences), the name and post of the collaborator, and the knowledge or lesson learned with the situation. The keywords and concepts given in the description must be between hashtags.*		
Inputs	**Input name**	**Data type**	**Valid values condition**
	uniqueIdentifier	String	
	description	String	
	capsuleType	String	*technical, management, domain, experiences*
	colaboratorName	String	
	colaboratorPost	String	
	lesson	String	*must have #*
	stage	int	Must be between 1 and 6, cannot be a greater number than the current stage
Result	The imputed info is sent to the method in the class “Green” named registerCapsule, where the project is searched by it’s name, when found the info is sent to regCapsule in Projects, where a simple matrix stores the capsule with [stage][capsuleNumber]		
Outputs	**Output Name**	**Data Type**	**Format**
	capsuleRegistered	String	“The capsule has been registered successfully in the project ” +projectName+ “in the

			stage ” +stage+ “ and is ready to be approved”

Name and identifier	*[FR5:Capsule Approbation]*		
Summary	*Registered capsules may be approved by inputting the project name and the capsule unique identifier, then the aprovation date is saved.*		
Inputs	**Input name**	**Data type**	**Valid values condition**
	proyectName	String	
	identifier	String	
Result	The data is sent to the method aproveCapsule on the Green class, that searches for the project by using its name, when it is found, the aproveCapsule method of Projects is called with the identifier and aprovation date, where the capsule is searched by using the identifier.		
Outputs	**Output Name**	**Data Type**	**Format**
	capsuleAproved	String	“The capsule has been approved successfully on ” +aprovationDate

Name and identifier	*[FR6:Capsule publishing]*		
Summary	*Approved capsules may be published, and a html link is generated when they are published.*		
Inputs	**Input name**	**Data type**	**Valid values condition**
	proyectName	String	
	identifier	String	
Result	This requirement functions really similarly to the capsule approbation one with the only differences being the link and that there is no publishing date. The data is sent to the method publishCapsule on the Green class, that searches for the project by using its name, when it is found, the publish Capsule method of Projects is called with the identifier, where the capsule is searched by using the identifier.		
Outputs	**Output Name**	**Data Type**	**Format**
	capsulePublished	String	“The capsule has been published successfully with the url www.capsule.com/”+Capsules[][].getIdentifier()+ “.html”

Name and identifier	*[FR7:Capsule search]*		
Summary	*The capsules may be searched by inputting a search string or hashtag*		
Inputs	**Input name**	**Data type**	**Valid values condition**
	hashtagSearch	String	*must contain at least a #*
Result	By using loops that iterate on the registered capsules, lessons of each one are checked to see if there's any coincidence with the introduced search string, if there is, the capsule project, stage, and lessons are printed.		
Outputs	**Output Name**	**Data Type**	**Format**
	searchCoincidences	String	"With the given search string, the capsules that contain something related are"+relatedCapsules