REQUIREMENT ANALYSIS| ReadX

Martín Gómez

Study case:

Client	ReadX
User	Company members and clients
Functional Requirements	**FR1)** Bibliographic products management **FR2)** User management **FR3)** Buy books and magazine subscriptions **FR4)** Library presentation **FR5)** Reading session simulation **FR6)** Report generation **FR7)** Testing management
Problem Context	ReadX is an egyptian conglomerate that is in need of a software that allows them to manage their publication business globally.
Non-Functional Requirements	**NFR1)** Make the program taking into consideration that they may add other types of biography products **NFR2)** Take into consideration the possible future creation of other user types

Name and identifier	*[FR1: Bibliographic products management]*
Summary	*For now, the two bibliographic products are books and magazines.* *On one hand, each book has a unique 3 hexadecimal digit identifier, a name, number of pages, a brief review, a publication date, a genre (Science Fiction, Fantasy, and Historic Novel), a url that leads to a repository with the book cover, the selling price in dollars, the number of sold units, and the number of accumulated read pages.* *On the other hand, Each magazine has a unique identifier (3 alphanumeric characters), a name, a number of pages, a publication date, a category (Variety, Design, and Sience), a URL leading to a*

	repository with the cover of the magazine, the value of the subscription (in dollars), the frequency of issuance, the number of active subscriptions and accumulated pages read. *As some inputs may be inherited from an upper class, the name remains the same but the valid conditions are different*		
Inputs	**Input name**	**Data type**	**Valid values condition**
	TProduct	int (flag variable)	*must be a valid product number (1 or 2 for now)*
	id	String	*Book : must be hexadecimal && 3 digits && id != all id’s* *Magazine:* must be alphanumeric && 3 digits && id != all id’s
	name	String	
	PageNum	int	
	review	String	
	PublicationDate	Date	
	genre	<<enum>> TGenre	SCIENCEFICTION FANTASY HISTORICNOVEL
	price	Double	
	category	<<enum>> TCategory	VARIETY DESIGN SIENCE
Result	Product added to the data base		
Outputs	**Output Name**	**Data Type**	**Format**
	bookCreated	String	“Book created successfully with the id” + id
	magazineCreated	String	“Magazine created successfully with the id” + id
	URL	String	Book: *a url that leads to a repository with the book cover* *Magazinel: URL leading to a repository with the cover of the magazine*

Name and identifier	*[FR2:* User management *]*
Summary	*For now there are two types of users: normal and premium. The regular user may buy up to 5 books and 2 Magazine subscriptions (with ads). The premium user however, may buy and subscribe to as*

	many books and magazines as he wishes (With no ads); to register to the platform a name, id, and vinculation date (generated) is needed.		
Inputs	**Input name**	**Data type**	**Valid values condition**
	name	String	
	id	String	
	user_type	String	*“normal” || “premium” (not an enum, only to call the constructor of the needed class)*
Result	if the selected user is premium a payment method may be required, then the functionalities are activated		
Outputs	**Output Name**	**Data Type**	**Format**
	userCreationSuccess	String	“User created successfully”
	userCreatiinFailure	String	“Invalid or already registered id”

Name and identifier	*[FR3:* Buy books and magazine subscriptions*]*		
Summary	*When users purchase a book or subscribe to a magazine, the date of the operation and the amount paid must be stored in a receipt, the number of copies sold or active subscriptions also has to be updated. the user also may cancel a magazine subscription at any moment.*		
Inputs	**Input name**	**Data type**	**Valid values condition**
	userID	String	*must be a registered id*
	bookName	String	
	magazineName	String	
	cardNumber	String	
Result			
Outputs	**Output Name**	**Data Type**	**Format**
	successfulPurchase	String	“Product added to your library”
	errorMsg	String	

Name and identifier	*[FR4: Library presentation]*
Summary	*A menu that allows the user to visualize his collection of bibliographic products. The library must be presented through 5x5 arrays that show the id of the products associated with the user. The products must be arranged by publication date, from oldest to newest. When the user's collection exceeds the 5x5 array, he may be able to go to the next and last page of collections.*

	Additionally, the user may select a product, but inputting either the id or coordinate x,y on the array, to initiate a reading session.		
Inputs	**Input name**	**Data type**	**Valid values condition**
	userID	String	*must be a registered user*
	desiredView	String	*must be a valid coordinate or a product owned by the user*
Result	Collection displayed, reading session initialized		
Outputs	**Output Name**	**Data Type**	**Format**
	collection	String	5x5 array (shown format in the document)
	errorView	String	"It must be a valid position, or the code of a product you own"

Name and identifier	*[FR5: reading session simulation]*		
Summary	*The reading session simulation is a simple presentation through the console of the product name, the actual page (starts in 1) the user is reading, and some browsing options like reading the next and last page, and going back to the library. As this is initialized in the user's library, there is no input needed to start a reading session, other than the desired view inputted in the last functional requirement.* *Each read page on the simulation increments the number of read pages of the product in the platform. It must be taken into consideration that during the reading session, normal users will be played ads at the beginning and every 20 pages read of a book or 5 pages read of a magazine.*		
Inputs	**Input name**	**Data type**	**Valid values condition**
	none	**none**	**none**
Result			
Outputs	**Output Name**	**Data Type**	**Format**
	readingSession	String	(Shown in the document)

Name and identifier	*[FR6:Report generation]*
Summary	*To create directed contents, ReadX requires the prototype to generate the following reports in real time:* *1. For each type of bibliographic product (Book and magazine) inform the total number of read pages in all the platform (book or magazine and corresponding read pages)*

	2. Inform the most read book genre and magazine category in all the platform (genre or category and corresponding read pages). *3. Inform the top 5 of books and magazines most read (name of the book or magazine, genre or category, and number of read pages)* *4. From each genre, inform the number of sold books and total revenue.* *5. From each category, inform the number of active subscriptions and the total value paid for them.*		
Inputs	**Input name**	**Data type**	**Valid values condition**
	desiredReport	int	*desiredReport>=0 $$ desiredReport<=5*
Result	The user inputs the number of the report that he wants generated, and there is no need for any other input, as the reports are generated automatically.		
Outputs	**Output Name**	**Data Type**	**Format**
	bookReadPages	String	“The number of read book pages is” + bookReadPages
	mostReadGenre	String	“The most read book genre in the platform is” + mostReadGenre
	mostReadCategory	String	“The most read magazine category in the platform is” + mostReadCategory
	top5ReadBooks	String	“The top 5 most read books are:” + “1.” + book.getName + “,” book.getGenre + “read pages:” + readPages 2….. and so on to 5
	top5ReadMagazines	String	“The top 5 most read magazines are:” + “1.” + magazine.getName + “,” magazine.getGenre + “read pages:” + readPages 2….. and so on to 5
	soldBooksPerGenre	String	“Science fiction:” + soldScienceNum + “With the revenue adding to:” + revenueScience “Fantasy:” + soldFantasyNum + “With the revenue adding to” +revenueFantasy

			“Historical Novel” : soldHistoricalNum + “With the revenue adding to:” + historicalRevenue
	soldMagazinesPerCategory	String	“Variety: ” + soldVarietyNum + “With the revenue adding to” + varietyRevenue “Design:” + soldDesignNum + “With the revenue adding to” + designRevenue “Científic:” soldCientificNum + “With the revenue adding to” +cientificRevenue

Name and identifier	*[FR7:Testing Management]*		
Summary	*Automatically generate, when required by the user at least an object of each user type and bibliographic product.*		
Inputs	**Input name**	**Data type**	**Valid values condition**
	none	**none**	**none**
Result	Object of each type generated for the user to test other functionalities		
Outputs	**Output Name**	**Data Type**	**Format**
	succesfullRandomCreation	String	“Objects created successfully”