Samuel J. Rengifo
A00404150

Client	Mayor's Office of Cali
User	COP Members Cali
Context of the problem	The 16th edition of the United Nations Framework Convention on Climate Change, one of the most important in the world, will be held in the city of Cali. Therefore, the Mayor of the city of Cali, requires a program that allows knowledge and navigation of both sites with biological diversity and communities that care for those places.
Functional requirements	FR0: Register community FR1: Register a place FR2: Register a product FR3: Delete a product FR4: Add species to a place FR5: Update species data in a place FR6: Create test case FR7: Query information about a place FR8: Query the information of the communities of a department FR9: Query communities with major problems FR10: Query the name of the place with the highest number of species FR11: Query the three largest places per square kilometer
Non-functional requirements	NFR0: Registered data must be protected NFR1: Intuitive interface NFR2: Cross-platform accessibility NFR3: The program must have documentation to facilitate future updates

Name or identifier	FR0: Register community		
Summary	This functional requirement will give the user the possibility to register a community filling some required values.		
Input	Input name	Datatype	Selection condition or repetition
	communityName	String	This string does not have any specific condition, but it must

			contain max 50 characters. It is not possible to repeat the name of a community that has already been created.
	communityTypeSelection	CommunityType	This selection should be one of the three following options: [1] Afrocolombian, [2] Indigenous or [3] Raizal.
	communityBossName	String	This string does not have any specific condition, but it must contain max 50 characters.
	communityBossPhoneNumber	String	This value should have exactly 10 digits.
	communityAmountInhabitants	int	This value cannot be negative.
Result or postcondition	The imputed values are going to get stored on the system database if the conditions are correct, if they are not, an error message will output.		
Output	Output name	Datatype	Selection condition or repetition
	message	String	The String message will inform the user if the process has been completed successfully or not.

Name or identifier	FR1: Register a place		
Summary	This functional requirement will give the user the possibility to register a place filling some required values.		
Input	Input name	Datatype	Selection condition or repetition
	placeName	String	The place name must be not already registered.
	placeDepartmentSelection	int	This selection should be one of the four following options: [1] Choco, [2] Valle, [3] Cauca, or [4] Narino.
	amountOfSquareKm	double	This value cannot be negative.

	placeTypeSelection	int	This selection should be one of the three following options: [1] Protected, [2] National Park, or [3] Private.
	placeOpeningDate	String	This selection value should be imputed in the following format: DAY/MONTH/YEAR
	placePhoto	String	This String value should refer to a valid URL direction, which contains the image to save.
	communityName	String	This value should be a community that has been created before, so in that order, this String, should just link them.
	placeFinancialResourcesRequired	double	This value cannot be negative.
Result or postcondition	The imputed values are going to get stored on the system database if the conditions are correct, if they are not, an error message will output.		
Output	Output name	Datatype	Selection condition or repetition
	message	String	The String message will inform the user if the process has been completed successfully or not.

Name or identifier	FR2: Register a product		
Summary	This functional requirement will give the user the possibility to register 20 products max for each community, just filling some required values.		
Input	Input name	Datatype	Selection condition or repetition
	productName	String	This string does not have any specific condition, but it must contain max 50 characters.
	productNaturalPercentage	double	This value cannot be negative.

	productTypeSelection	int	This selection should be one of the two following options: [1] Food, or [2] Craft.
	productHandmade	String	This selection must be YES or NO.
	communityName	String	This value should be a community that has been created before, so in that order, this String, should just link them.
Result or postcondition	The imputed values are going to get stored on the system database if the conditions are correct, if they are not, an error message will output.		
Output	Output name	Datatype	Selection condition or repetition
	message	String	The String message will inform the user if the process has been completed successfully or not.

Name or identifier	FR3: Delete a product		
Summary	This functional requirement will enable the user to delete a product that has been already registered.		
Input	Input name	Datatype	Selection condition or repetition
	productNameToDelete	String	Product name must be already registered, so in that order delete it.
Result or postcondition	The imputed value will work as an identifier on the system database, if they are correct, the selected product will be deleted, if they are not, an error message will be produced.		
Output	Output name	Datatype	Selection condition or repetition
	message	String	The String message will inform the user if the process has been completed successfully or not.

Name or identifier	FR4: Add species to a place

Summary	This functional requirement will enable the user to create a specie and add it to a place.		
Input	Input name	Datatype	Selection condition or repetition
	specieName	String	This string does not have any specific condition, but it must contain max 50 characters.
	specieBiodiversityType Selection	int	This selection should be one of the two following options: [1] Flora, or [2] Fauna.
	speciePhoto	String	This String value should refer to a valid URL direction, which contains the image to save.
	specieLivingAmount	int	This value cannot be negative.
	placeName	String	Place name must be already registered to link it to a specie.
Result or postcondition	The imputed values are going to get stored on the system database if the conditions are correct, if they are not, an error message will output.		
Output	Output name	Datatype	Selection condition or repetition
	message	String	The String message will inform the user if the process has been completed successfully or not.

Name or identifier	FR5: Update species data in a place		
Summary	This functional requirement will enable the user to update a specie data in a place.		
Input	Input name	Datatype	Selection condition or repetition
	specieName	String	Specie name must be already registered.
	attributeToUpdate	int	This selection should be one of the four following options:

			[1] Specie name, [2] Biodiversity Type, [3] Specie photo, or [4] Living Species Amount.
	newSpecieName	String	This string does not have any specific condition, but it must contain max 50 characters.
	newSpecieBiodiversityTypeSelection	String	This selection should be one of the two following options: [1] Flora, or [2] Fauna.
	newSpeciePhoto	String	This String value should refer to a valid URL direction, which contains the image to save.
	newSpecieLiving	int	This value cannot be negative.
Result or postcondition	The imputed values are going to replace the past values stored on the system database if the conditions are correct, if they are not, an error message will be produced.		
Output	Output name	Datatype	Selection condition or repetition
	message	String	The String message will inform the user if the process has been completed successfully or not.

Name or identifier	FR6: Create test case		
Summary	This functional requirement will create all the possible objects to test the program functional requirement queries.		
Input	Input name	Datatype	Selection condition or repetition
	N/A		
Result or postcondition	The system will create 5 communities, 5 places, 5 products and 5 species with default values assigned before.		
Output	Output name	Datatype	Selection condition or repetition
	message	String	The String message will inform the user if

			the process has been completed successfully.

Name or identifier	FR7: Query information about a place		
Summary	This functional requirement will show to the user all the information about a specific place.		
Input	Input name	Datatype	Selection condition or repetition
	placeName	String	Place name must be already registered.
Result or postcondition	The imputed value will work as an identifier on the system database, if it is correct, all the information about the selected place will be displayed, if the identifier is not valid, an error message will be produced.		
Output	Output name	Datatype	Selection condition or repetition
	message	String	The String message will show to the user all the attributes of the selected place if the place name was registered, otherwise it will display that the place wasn't found.

Name or identifier	FR8: Query the information of the communities of a department		
Summary	This functional requirement will show to the user all the information about all the communities in a specific department.		
Input	Input name	Datatype	Selection condition or repetition
	placeDepartmentSelection	int	This selection should be one of the four following options: [1] Choco, [2] Valle, [3] Cauca, or [4] Narino.
Result or postcondition	The imputed value will work as an identifier on the system database, if it is correct and there are communities linked to the place, all the information about the communities in the selected department will be displayed, if the identifier is not valid, an error message will be produced.		
Output	Output name	Datatype	Selection condition or repetition
	message	String	The String message will show to the user all the communities of

			the selected department if the same was registered, otherwise it will display that there aren't registered communities in that department or that there aren't registered places.

Name or identifier	FR9: Query communities with major problems		
Summary	This functional requirement will show to the user all the communities that match with the selected major problem.		
Input	Input name	Datatype	Selection condition or repetition
	N/A		
Result or postcondition	If there are any communities that match with the conditions, all the information about the communities will be displayed, otherwise a generic message will be displayed.		
Output	Output name	Datatype	Selection condition or repetition
	message	String	If there are any communities that match with the conditions, the String message will be returned with all the information about the communities, otherwise the message will be that no communities were created or none of the communities had the indicated problems.

Name or identifier	FR10: Query the name of the place with the highest number of species		
Summary	This functional requirement will show to the user, the place with the highest number of species.		
Input	Input name	Datatype	Selection condition or repetition

	N/A		
Result or postcondition	By executing a method, the system will search between the registered places, which is the place with the highest number of species, and finally it will show on screen the name of the place that applies to the condition.		
Output	Output name	Datatype	Selection condition or repetition
	message	String	If there are any places that match with the conditions, the String message will be returned with all the information about the place(s), otherwise the message will be that there are no places created or no species linked.

Name or identifier	FR11: Query the three largest places per square kilometer		
Summary	This functional requirement will show to the user, the three largest places per square kilometer.		
Input	Input name	Datatype	Selection condition or repetition
	N/A		
Result or postcondition	By executing a method, the system will search between the registered places, which are the three largest places per square kilometer, and finally it will show on screen the name of the three places that apply to the condition.		
Output	Output name	Datatype	Selection condition or repetition
	message	String	If there are any places that match with the conditions, the String message will be returned with all the information about the three largest place(s), per Km2 otherwise the message will be that there are no places created.

Functional Requirement	**Class Name**	**Method name**
FR0 Register community	Class CopSystem	+registerCommunity(Controller objAdmin) : void
	Class Controller	+registerCommunity(String communityName, int communityTypeSelection, String communityBossName, String communityBossPhoneNumber, int[] tempCommunityMajorProblems, int communityAmountInhabitants, int amountOfProblems) : String +communityAvailable() : boolean +searchCommunity(communityName) : Community
	Class Community	+Community(String communityName, CommunityType communityType, String communityBossName, String communityBossPhoneNumber, CommunityMajorProblems[] communityMajorProblems, int communityAmountInhabitants)
FR1 Register a place	Class CopSystem	+registerPlace(Controller objAdmin) : void
	Class Controller	+registerPlace(String placeName, int placeDepartmentSelection, double amountOfSquareKm, int placeTypeSelection, String placeOpeningDate, String placePhoto, String communityName, double placeFinancialResourcesRequired) : String +placeAvailable() : boolean +searchPlace(placeName) : Place +searchCommunity(communityName) : Community
	Class Place	+Place(String placeName, PlaceDepartment

		placeDepartment, double amountOfSquareKm, PlaceType placeType, String placeOpeningDate, String placePhoto, Community placeCommunity, double placeFinancialResourcesRequired)
FR2 Register a product	Class CopSystem	+registerProduct(Controller objAdmin) : void
	Class Controller	+registerProduct(String productName, double productNaturalPercentage, int productTypeSelection, String productHandmade, String communityName) : String +searchCommunity(communityName) : Community
	Class Community	+addProduct(Product product) : boolean
	Class Product	+Product(productName : String, productNaturalPercentage : double, productType : String, productHandmade : String)
FR3 Delete a product	Class CopSystem	+deleteProduct(Controller objAdmin) : void
	Class Controller	+deleteProduct(String productName) : String
	Class Community	+deleteOneProduct(String productName, Product[] productList) : String
FR4 Add species to a place	Class CopSystem	+addSpecieToPlace(Controller objAdmin) : void
	Class Controller	+addSpecieToPlace(String specieName, int specieBiodiversityTypeSelection, String speciePhoto, int specieLivingAmount, String placeName) : String +searchPlace(placeName) : Place
	Class Place	+addSpecie(Specie specie) : boolean

	Class Specie	+Specie(specieName : String, specieBiodiversityType : String, speciePhoto : String, specieLivingAmount : int)
FR5 Update species data in a place	Class CopSystem	+updateSpecieToPlace(Controller objAdmin) : void
	Class Controller	+updateSpecie(String specieName, int newSpecieBiodiversityTypeSelection, String newSpeciePhoto, int newSpecieLivingAmount, String newSpecieName) : String +searchSpecie(specieName) : Specie
	Class Specie	+setSpecieBiodiversityType(SpecieBiodiversityType specieBiodiversityType) : void +setSpeciePhoto(String speciePhoto) : void +setSpecieLivingAmount(int specieLivingAmount) : void +setSpecieName(String specieName) void
FR6 Create test case	Class CopSystem	+testCase(Controller objAdmin) : void
	Class Controller	+createTestCaseValues() : String
FR7 Query information about a place	Class CopSystem	+showPlace(Controller objAdmin) : void
	Class Controller	+showPlace(String placeName) : String +searchPlace(placeName) : Place
	Class Place	+toString() : String
FR8 Query the information of the communities of a department	Class CopSystem	+showDepartmentCommunity(Controller objAdmin) : void
	Class Controller	+showCommunityDepartment(int placeDepartmentSelection) : String
	Class Place	+getValue() : String +getPlaceDepartment() :

		PlaceDepartment +getCommunity() : Community
	Class Community	+getName() : String
FR9 Query communities with major problems	Class CopSystem	+showCommunityWithMajorProblems(Controller objAdmin) : void
	Class Controller	+queryCommunityMajorProblem() : String
	Class Community	+getCommunityMajorProblems() : CommunityMajorProblems[]
		+toString() : String
FR10 Query the name of the place with the highest number of species	Class CopSystem	+showPlaceNameWithMostSpecies(Controller objAdmin) : void
	Class Controller	+queryPlaceNameWithMostSpecies() : String
	Class Place	+getSpecieList() : Specie[] +getName() : String
FR11 Query the three largest places per square kilometer	Class CopSystem	+showThreeBiggestPlacesPerSquareKm(Controller objAdmin) : void
	Class Controller	+queryThreeBiggestPlacesPerSquareKm() : String
	Class Place	+getSquareKm() : double