

H.I.I.T

A scheduling/booking system by:

TEAM 7

STRENGTHENING SOLUTIONS

Luke Partridge

U19010240 u19010240@tuks.co.za +27 64 682 0521

Shannon Noel

U20502746 <u>u20502746@tuks.co.za</u> +27 73 748 6187

George Liatos

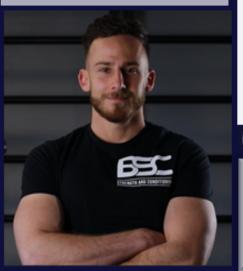
U20453061 u20453061@tuks.co.za +27 84 095 8527

Lyné Keet

U20690844 u20690844@tuks.co.za +27 83 407 7093

Joshua Bester

U20625741 u20625741@tuks.co.za +27 65 804 5796





ITERATION 6 - Design, Model, Interface

Development and Coding

Logical Narratives:

The following document contains the logical narratives for the Administrator Subsystem, Employee Subsystem, Login Subsystem, Sale Subsystem, Client/Member Subsystem, Payments Subsystem, Booking Subsystem, Lesson Plan Subsystem and Inventory Subsystem. Logical Narratives are displayed that hold a detailed explanation of how these subsystems will be operating. The explanation is based on the narratives that include the actions that will take place alongside all entities and attributes for the functionality of the proposed system.

01/08/2022

Client Information:

Organization: Bester Strength and Conditioning Gym

Organisational Contact: Neil Bester Contact No: +27 66 057 1889

Email: info@besterstrengthandconditioning.co.za

Position: Owner



Iteration 6 2022/08/01

Table of Contents

1. DOCUMENT INTRODUCTION	3
2. LOGICAL NARRATIVES	4
2.1 Logical Narratives	4
2.1.2 Employee Subsystem	4
Table 2.1.2.1 Create Employee	4
Table 2.1.2.2 Search Employee	7
Table 2.1.2.3 Update Employee	10
Table 2.1.2.4 Delete Employee	13
Table 2.1.2.5 View Employee	16
2.1.3 Login Subsystem	18
Table 2.1.3.1 Forgot Password	18
Table 2.1.3.2 Change Password	21
2.1.4 Sale Subsystem	23
Table 2.1.4.1 Search Sale Item	24
Table 2.1.4.2 Search Sales Category	26
Table 2.1.4.3 Add to Cart	28
Table 2.1.4.4 View Cart	30
Table 2.1.4.5 Edit Cart	32
Table 2.1.4.7 View Shop	34
Table 2.1.4.10 Create Refund Reason	35
Table 2.1.4.11 Search Refund Reason	37
Table 2.1.4.12 Update Refund Reason	39
Table 2.1.4.13 Delete Refund Reason	41
Table 2.1.4.14 View Refund Reason	43
2.1.5 Client/Member Subsystem	44
Table 2.1.5.1 Register Client	44
2.1.7 Booking Subsystem	46
Table 2.1.7.1 Create Schedule	46
Table 2.1.7.2 Search Schedule	49
Table 2.1.7.3 Update Schedule	51
Table 2.1.7.4 Delete Schedule	54
Table 2.1.7.5 View Schedule	56
Table 2.1.7.6 Create Booking Type	57
Table 2.1.7.7 Search Booking Type	59
Table 2.1.7.8 Update Booking Type	61
Table 2.1.7.9 Delete Booking Type	63
Table 2.1.7.10 View Booking Type	65
2.1.8 Lesson Plan Subsystem	66
Table 2.1.8.6 Create Exercise	66
Table 2.1.8.7 Search Exercise	68
Table 2.1.8.8 Update Exercise	70





Iteration	(
2022/08/0	٠

3. DOCUMENT CONCLUSION	95
Table 2.1.9.14 Receive Stock	94
Table 2.1.9.8 ViewWrite Off Reason	93
Table 2.1.9.7 Delete Write Off Reason	91
Table 2.1.9.6 Update Write Off Reason	89
Table 2.1.9.5 Search Write Off Reason	87
Table 2.1.9.4 Create Write Off Reason	85
2.1.9 Inventory Subsystem	85
Table 2.1.8.15 View Exercise Category	84
Table 2.1.8.14 Delete Exercise Category	82
Table 2.1.8.13 Update Exercise Category	79
Table 2.1.8.12 Search Exercise Category	77
Table 2.1.8.11 Create Exercise Category	75
Table 2.1.8.10 View Exercise	74
Table 2.1.8.9 Delete Exercise	72



1. DOCUMENT INTRODUCTION

The following document contains the logical narratives for the Administrator Subsystem, Employee Subsystem, Login Subsystem, Sale Subsystem, Booking Subsystem, Client/Member Subsystem, Inventory Subsystem, Lesson Plan Subsystem. Logical Narratives are displayed that hold a detailed explanation of how these subsystems will be operating. The explanation is based on the narratives that include the actions that will take place alongside all attributes and entities for the functionality and processes of the proposed system.



3

2. LOGICAL NARRATIVES

2.1 LOGICAL NARRATIVES

2.1.2 Employee Subsystem *Table 2.1.2.1 Create Employee*

Table 2.1.2.1 Create Emp	noyee	
Author(s): Shannon Noel		Creation Date:20/05/2022
Version: 1.0		Last Review Date: 18/07/2022
Use Case Name:	Create Employee	Use case type
Use Case ID:	2.1	Business Requirements:
Priority:	HIGH	System Analysis:
Source:	Bester Strength and Conditioning	System Design: □
Primary Business Actor:	Administrator	
Primary System Actor:	None	
Other Participating Actors:	None	
Other Interested Stakeholders:	None	
Description:	This use case describes the event where an employee record is created on the system by the administrator. The use case begins when the administrator requests to create a recently employed employee on the system. The administrator enters all the employee's information required by the system. The system verifies the information and then stores it. The use case concludes when the employee record has been successfully created on the system.	
Pre-condition:	 The administrator must be left. 	ogged into the system.
Trigger:	The administrator requests to create	e a new employee record on the system
Typical Course	Actor Action:	system Response:
		1. The system prompts the administrator to enter the following details concerning the employee on a form: - Employee_Photo - Employee_SA_ID - Employee_Contract_File
		from the Employee table
		 Firstname Lastname Email PhoneNumber From the AspNetUser table (The Primary Key: User_ID is the Foreign Key in the Employee table).
		- Employee_Type_Name from the Employee_Type table (the Primary Key: Employee_Type_ID is the Foreign Key in Employee table)



	- Qualification_Description from the Qualification table (the Primary Key: Qualification_ID is the Foreign Key in Employee table) - Name (representing role name) From the AspNetRoles table (the Primary Key: Roleld is the Foreign Key in the AspNetUserRoles table, the Primary Key in the AspNetUsers table: User_ID is the Foreign Key in the AspNetUserRoles table). - Title_Description From the Title table (The Primary Key: Title_ID is the Foreign Key in the User table).
The administrator enters the required information.	3. The system validates the fields ensuring that the required data fields are not empty and that data types are correct - Name should be text - Surname should be text - Photo is optional - SA_ID should be validated according to a valid South African ID (13 digits) Email should have an @ - Phone Number should be 10 digits. [ALT] 4. The system will display the
	entered information and request the administrator to confirm the entered information.
5. The administrator confirms the information entered. [ALT]	6. The system verifies that the employee record does not already exist in the Employee and AspNetUser tables by checking that the following attributes: - Employee_SA_ID - Email - PhoneNumber do not match an existing employee record. [ALT]



2022/08/01

		T T T T T T T T T T T T T T T T T T T
		7. The system generates a new User_ID and a new Employee_ID which is set to the next consecutive number and adds the new employee record to the Employee table with the following attributes: - Employee_ID (the system will auto generate an Employee_ID by taking the last ID and adding one.) - Employee_SA_ID - Employee_SA_ID - Employee_Contract_File - Qualification_ID - Employee_Type_ID - Title_ID - Role_ID and AspNetUser table: - User_ID (the system will auto generate a User_ID by taking the last ID and adding one.) - Firstname - Lastname - Email - PhoneNumber
		8. The system displays a success message to the administrator that the employee record has been created and an email is sent to the employee with a password to access the system.
Alternate	[Al T] Stan 4: A required field is not	filled in or it is not in the correct format.
Courses:		age to the administrator. Return to step 3.
		nies the information entered is correct.
	Return to step 3.	
	[ALT] Step 7: The employee alread	
	alsplays an error notification informi	ing the administrator. Return to step 3.
Conclusion:	The use case concludes when the	
Doot condition:	database and the success message	, ,
Post-condition:	A new employee is created on the s This employee can now access the	system.
Business Rules:	 Only authorised users of the employee (an administrator) 	system will be able to create an
Implementation Constraints and Specifications	None	•
Assumptions	None	
Open Issues	None	





Table 2.1.2.2 Search Employee

Table 2.1.2.2 Search Employee		
Author(s): Shannon Noel		Creation Date:20/05/2022
Version: 1.0		Last Review Date: 18/07/2022
Use Case Name:	Search Employee	Use case type
Use Case ID:	2.2	Business Requirements:
Priority:	HIGH	System Analysis:
Source:	Bester Strength and Conditioning	System Design: □
Primary Business Actor:	Administrator	
Primary System Actor:	None	
Other Participating Actors:	None	
Other Interested Stakeholders:	None	
Description:	search for an employee on the administrator requests to searc when the information is display	-
Pre-condition:	The administrator must be logg	
Triggor:	The employee must exist on the	search an employee on the system.
Trigger: Typical Course	Actor Action:	system Response:
	 The administrator requests to search for an employee on the system. 	The system prompts the administrator to enter any details concerning the employee.
		4. The system retrieves and displays a list of all the employees which matches the search criteria entered with the following attributes: - Employee_Photo - Employee_SA_ID - Employee_Contract_File from the Employee table - Employee Type Name
	The administrator enters the search criteria.	from the Employee_Type table (the Primary Key: Employee_Type_ID is the Foreign Key in Employee table)
		- Qualification_Description from the Qualification table (the Primary Key: Qualification_ID is the Foreign Key in Employee table)
		FirstnameLastnameEmailPhoneNumber



du a n b	
۳IOJI۳	
89	
()	

		I =
		From the AspNetUser table (The Primary
		Key: User_ID is the Foreign Key in the Employee table).
		Linployee table).
		- Title_Description
		From the Title table (The Primary Key:
		Title_ID is the Foreign Key in the User
		table). [ALT]
		6. The system retrieves and displays the following information regarding the specific employee record: - Employee_Photo - Employee_SA_ID - Eployee_Contract_File
		from the Employee table
		- Employee_Type_Name from the Employee_Type table (the Primary Key: Employee_Type_ID is the Foreign Key in Employee table)
	5. The administrator selects the specific employee record they wish to view from the	- Qualification_Description from the Qualification table (the Primary Key: Qualification_ID is the Foreign Key in Employee table)
	displayed results.	- Firstname
	-	- Lastname
		- Email
		- PhoneNumber From the AspNetUser table (The Primary
		Key: User_ID is the Foreign Key in the
		Employee table).
		, i
		- Title_Description
		From the Title table (The Primary Key: Title_ID is the Foreign Key in the User table).
Alternate		
Courses:	[ALT] Step 4: No results were found. The use case terminates.	
Conclusion:	The use case concludes with the searched employee record being	
Doot condition:	displayed successfully.	
Post-condition: Business Rules:	The administrator can view the searched employee record successfully. - Only authorised users of the system will be able to search an	
Dusiliess Kules.	employee (an administrator). The administrator must be registered on the system and logged into the system to be able to search for an employee record.	
Implementation Constraints and Specifications	None	





Logical Narratives

Iteration 6 2022/08/01

Assumptions	None
Open Issues	None





Table 2.1.2.3 Update Employee

Table 2.1.2.3 Update Employee		
Author(s): Shannon Noel Version: 1.0		Creation Date:20/05/2022 Last Review Date: 18/07/2022
Use Case Name:	Update Employee	Use case type
Use Case ID:	2.3	Business Requirements:
Priority:	HIGH	System Analysis:
Source:	Bester Strength and Conditioning	System Design: □
Primary Business Actor:	User	•
Primary System Actor:	None	
Other Participating Actors:	Employee Administrator	
Other Interested Stakeholders:	None	
Description:	This use case describes the event where the user would like to update the information of an employee. The use case starts when the user requests to update an employee's information. The use case concludes when the user is notified that the employee's information has successfully been updated.	
Pre-condition:	The user must be loggeThe employee must exist	st on the system.
Trigger:	The user requests to update an employee's personal information on the system.	
Typical Course	Actor Action:	system Response:
	1. The administrator requests to update a specific employee record on the system.	2. The system retrieves and displays the employee information that has been retrieved with certain modifiable fields which includes the following attributes: - Employee_Photo - Employee_SA_ID - Employee_Contract_File from the Employee table - Qualification_Description from the Qualification table (the Primary Key: Qualification_ID is the Foreign Key in Employee table) - Firstname - Lastname - Lastname - Email - PhoneNumber From the AspNetUser table (The Primary Key: User_ID is the Foreign Key in the Employee table).



-	-
	- Title_Description From the Title table (The Primary Key: Title_ID is the Foreign Key in the AspNetUsers table).
	3. The system prompts the user to edit the pre-populated fields. The fields are populated with the information that was retrieved from the Employee, Qualification, Title and AspNetUsers tables.
4. The user will update the relevant information of the employee record and select the option to save the changes made.	5. The system validates the fields ensuring that the required data fields are not empty and that data types are correct Name should be text Surname should be text Photo is optional SA_ID should be validated according to a valid South African ID (13 digits). User_Email should have an @ User_Cell should be 10 digits. [ALT]
	6. The system displays the newly entered information and requests the user to read and confirm the changes that are to be made to the employee information.
7. The user confirms the updates made to the employee information. [ALT]	8. The system verifies that the employee record does not already exist in the Employee and AspNetUsers tables by checking that the following attributes: - Employee_SA_ID - Firstname - Lastname - Email - PhoneNumber

11



		do not match an existing employee record. [ALT]
		9. The system will update the employee record to the Employee table with the following attributes: - Employee_ID (remain the same.) - Employee_Photo - Employee_SA_ID - Employee_Contract_File - Qualification_ID - Employee_Type_ID and AspNetUser table: - User_ID (remain the same.) - Firstname - Lastname - Email - PhoneNumber 10. The system displays a success notification for the successfully updated
Alternate Courses:	fields are not in the correct form under the input fields that are expected [ALT] Step 7: The user denies to step 4. [ALT] Step 8: The employee al	employee record. put fields are not filled in or the required nat. The system displays error messages mpty or invalid. Return to step 4. the information entered is correct. Return ready exists on the system, the system orming the user. Return to step 6.
Conclusion:	The use case concludes when the system notifies the user that the changes to the employee record have been successfully made.	
Post-condition:	The relevant employee information has been successfully updated in the respective tables: Employee table and AspNetUser table.	
Business Rules:	 Only authorised users of the system will be able to update an employee (an administrator and employee). 	
Implementation Constraints and Specifications	None	
Assumptions	None	
Open Issues	None	



Table 2.1.2.4 Delete Employee

Table 2.1.2.4 Delete Emplo	oyee		
Author(s): Shannon Noel		Creation Date:20/05/2022	
Version: 1.0	Ta = .	Last Review Date:	
Use Case Name:	Delete Employee	Use case type	
Use Case ID:	2.4	Business Requirements:	
Priority:	HIGH	System Analysis:	
Source:	Bester Strength and Conditioning	System Design: □	
Primary Business Actor:	Administrator	•	
Primary System Actor:	None		
Other Participating Actors:	None		
Other Interested Stakeholders:	None		
Description:	delete a specific employee rec when the administrator request administrator will search for the	This use case describes the event where the administrator would like to delete a specific employee record on the system. The use case begins when the administrator requests to delete an employee. The administrator will search for the employee and proceed to delete them. The use case concludes once the employee has been deleted.	
Pre-condition:		be logged into the system.	
	The employee must exist exists.	st on the system.	
Trigger:	The administrator requests to c	deactivate an employee on the system.	
Typical Course	Actor Action:	system Response:	
	1. The administrator requests to delete a specific employee record in the system.	2. The system will confirm no lesson or schedule records are associated with the selected Emploee_ID as a FK in the Lesson and Schedule tables. [ALT]	
		3. The system retrieves and displays the employee information that has been retrieved with the following attributes: - Employee_Photo - Employee_SA_ID - Employee_Contract_File	
		- Qualification_Description from the Qualification table (the Primary Key: Qualification_ID is the Foreign Key in Employee table) - Firstname - Lastname - Email - PhoneNumber	



_	Strengthening	Solutions
1	Team 7	

		From the AspNetUser table (The Primary Key: User_ID is the Foreign Key in the Employee table). - Name (Role Name) from the AspNetRoles table where the User_ID FK is the PK in the AspNetUsers table. - Title_Description From the Title table (The Primary Key: Title_ID is the Foreign Key in the AspNetUser table).	
		4. The system responds by prompting the administrator to confirm the deletion of the selected employee.	
	5. The administrator confirms the deletion of the employee record. [ALT]	6. The system deletes the employee record with the following attributes: - Employee_Photo - Employee_SA_ID - Employee_Contract_File from the Employee table - Firstname - Lastname - Email - PhoneNumber From the AspNetUsers table (The Primary Key: User_ID is the Foreign Key in the Employee table).	
		7. The system notifies the administrator that the employee has been successfully deleted.	
Alternate Courses:	[ALT] Step 2: The system determines there are lesson or schedule record/s associated with the selected employee record. The system will notify the administrator, and ask them to ensure no lesson or schedule records are associated before deletion. The use case terminates		
	[ALT] Step 4: The administrato use case terminates.	r denies the deletion of the employee.The	
Conclusion:	The use case concludes when the system notifies the administrator that the employee has been successfully deleted.		
Post-condition:	The Employee table does not d	The Employee table does not contain the information of the deleted employee.The deleted employee cannot gain access into the system.	







Logical Narratives

Iteration 6 2022/08/01

Business Rules:	 Only authorised users of the system will be able to delete an employee (an administrator).
Implementation Constraints and Specifications	None
Assumptions	None
Open Issues	None



Table 2.1.2.5 View Employee

Table 2.1.2.5 View Employe	8	
Author(s): Shannon Noel Version: 1.0		Creation Date:20/05/2022 Last Review Date:
Use Case Name:	View Employee	Use case type
Use Case ID:	2.5	Business Requirements:
Priority:	Medium	System Analysis:
Source:	Bester Strength and Conditioning	System Design:
Primary Business Actor:	Administrator	
Primary System Actor:	None	
Other Participating Actors:	None	
Other Interested Stakeholders:	None	
Description:	access the employee pag the administrator to acces employees.	the event where the administrator wants to ge. The system will display the page; allowing ss the create, search, update and delete for
Pre-condition:	The administrator must be	
Trigger:		requests to view the employee page
Typical Course	Actor Action:	system Response:
	1. The administrator requests to view the employee page	2. The system will display a populated list with : - Employee_Photo from the Employee table - Firstname - Lastname - Email - PhoneNumber From the AspNetUsers table (the primary key User_ID is the Foreign Key in the Employee table) -Name (Role name) From the AspNetRoles table (the Primary Key User_ID in the AspNetUsers table is the Foreign Key in the AspNetRoles table)
	3. The administrator requests to search an employee record. [ALT]	4. The system invokes Use Case <u>2.2</u> <u>"Search Employee".</u>
Alternate Courses:	[ALT] Step 3a: The administrator requests to create an employee record. The system invokes Use Case 2.1 "Create Employee".	



	[ALT] Step 3b: The administrator requests to update employee records.	
	The system invokes Use Case 2.3 "Update Employee".	
	[ALT] Step 3c: The administrator requests to delete employee records. The system invokes Use Case 2.4 "Delete Employee".	
Conclusion:	The system displays a list with all the employee records and their information	
Post-condition:	The administrator will be able to search on the employee Page	
Business Rules:	 Only administrators can view the employee page. 	
Implementation Constraints and Specifications	None	
Assumptions	None	
Open Issues	None	

17



2.1.3 Login Subsystem

Table 2.1.3.1 Forgot Password

Author(s): Lyne Keet, Shannon Noel Version: 1.0	ii d	Creation Date:12/04/2022 Last Review Date:26/07/2022
Use Case Name:	Forgot Password	Use case type
Use Case ID:	3.3	Business Requirements:
Priority:	HIGH	System Analysis:
Source:	Bester Strength and Conditioning	System Design: □
Primary Business Actor:	User	
Primary System Actor:	None	
Other Participating Actors:	None	
Other Interested Stakeholders:	None	
	selecting the forgot passw will request the user to en searches for a user profile address entered by the us auto-generates a One-Tin that is retrieved from the s SMS and enters the sequ the entered pin matches t the user to enter a new pa use case concludes once password has been succe	
Pre-condition:	The user must be registered on the system	
Trigger:	The user will selects "Forgot password" on the Login screen	
Typical Course	Actor Action:	system Response:
	The user will select the forgot password function on the login screen.	The system responds by prompting the user to enter his/her email address.
	3. The user enters his/her email address in the corresponding field and submits to proceed	4. The system validates the fields ensuring it is in the correct format and that the field is not left empty. (there must be an '@' sign within the email) [ALT]
		5. The system further validates the entered information against the credentials of all user profiles in the AspNetUser table, verifying if the entered email address matches the email address of a user on the system.



	The following information is read from
	the AspNetUser table: - Email
	- NormalizedEmail
	[ALT] 6. The system retrieves the matching
	user profile and the following information is read from the
	AspNetUser table
	phoneNumber7. The system generates a random
	One-Time-Pin to send to the user for them to gain access to the system.
	The system will SMS the OTP to the retrieved phoneNumber record. [ALT]
	The system will load a screen prompting the user to enter the OTP
40 The week	sent via SMS
10. The user accesses the	
auto-generated SMS and enters	11. The system validates the entered One-Time-Pin sequence against the
the received	system-generated pin that was sent,
One-TimePin in a designated	confirming a match [ALT]
input area	12. The system will load a new screen
	where it will prompt the user to enter
	a new password and to confirm the password.
13. The user enters his/her new	14. The system will validate the entered
password and	password to ensure the password
confirms the new password in the	matches the confirmed password entered. [ALT]
specified fields.	15. The system also validates the
	password requirements, validating that:
	 The new password meets the
	minimum length requirement of 8 characters
	 The new password contains at least one special character
	 The new password has one upper case letter.
	[ALT]
	16. The system hashes the user's new password and creates a new record
	in the Password_History table, using the following attributes:
	using the following attributes.



11 - 11 0
۳ILOJI۳
ור זו

	 Password_ID(the value of 	
	which is the previous	
	Password_ID incremented by	
	1)	
	- Password_Date	
	- Hashed_Password	
	 User_ID (which is read from 	
	the AspNetUser table)	
	17. The system displays a message	
	which notifies the user that their	
	password has been successfully	
	reset.	
	18. The system will load the login page.	
Alternate	[ALT] Step 4: the email field has been left empty or the email data	
Courses:	entered have incorrect data formats. The system displays a notification	
Courses.	stating the error. Return to step 3.	
	· · · · · · · · · · · · · · · · · · ·	
	[ALT] Step 5: The system's validation was unsuccessful as no	
	matching profile was retrieved with the entered email. The system will	
	display an error message. Return to step 3.	
	[ALT] Step 8: The system fails to SMS the otp and thus emails the otp	
	to the user's email address. Go to step 8.	
	[ALT] Step 11:The system's One-Time-Pin validation was	
	unsuccessful. The system will display an error message. Return to step	
	10	
	[ALT] Step 14: The new password and confirm password fields do not	
	match. The system will display an error message. Return to step 13	
	[ALT] Step 15: The new password does not meet the requirements.	
	The system will display an error message. Return to step 13	
Conclusion:	The case concludes when the user's password has been reset	
	successfully and the system has returned to the login screen.	
Post-condition:	The user's password has been inserted in the Password_History table	
	and the user is able to login to the system with their new password.	
Business Rules:	- Passwords should contain a minimum of 8 characters.	
	- Passwords should contain at least one special character and	
	one upper case letter.	
	 New password cannot be the same as the old password. 	
Implementation Constraints and	The password must be hashed before it is stored in the	
Specifications	database.	
	- The user requires internet	
Assumptions	The user's email address on the system is their most recent email	
	address and they have access to the account.	
Open Issues	N/A	
- P - 1 100000		



Table 2.1.3.2 Change Password

Table 2.1.3.2 Change Password		
Author(s): Lyne Keet, Shannon Noel		Creation Date:12/04/2022 Last Review Date:26/07/2022
Version: 1.0		
Use Case Name:	Change Password	Use case type
Use Case ID:	3.4	Business Requirements:
Priority:	HIGH	System Analysis:
Source:	Bester Strength and Conditioning	System Design:
Primary Business Actor:	User	
Primary System Actor:	None	
Other Participating Actors:	None	
Other Interested Stakeholders:	None	
Description:	This use case describes the event where a user wants to change their saved password on the system. The use case begins when the user requests to update their password on the system. The system will first request the user to enter their current password, and then the user will be able to enter a new password. The use case concludes when the system has successfully updated the user's password and displays a success notification to the user.	
Pre-condition:	The user is registered and log	
Trigger:	The user requests to change	their password.
Typical Course	Actor Action:	system Response:
	The user requests to change their password.	The system will prompt the user to enter their current password, their new password, and to confirm their new password.
	3. The user will enter their current password, their new password and will then confirm their new password in the corresponding fields.	4. The system retrieves the matching user profile and the following information is read from the Password_History table - Hashed_Password The system will successfully validate if the current password entered matches the current password associated with the user's profile [ALT]
		The system will validate the new password and the confirmed password matches [ALT]
		6. The system also validates the password requirements, validating that: • The new password meets the minimum length requirement of 8 characters • The new password contains at least one special characters





The new password has at least one upper case letter. ALT			
7. The system will validate that the new password entered in the field does not match the user's current password stored. The system will match the user's current password entered with the user's current password stored in the Password_History table as the Hashed_Password attribute, by comparing the User_ID attribute in AspNetUser table with the User_ID attribute in the Password_History table. [ALT] 8. The system hashes the user's new password and creates a new record in the Password_History table. using the following attributes: - Password_History table, using the following attributes: - Password_ID incremented by 1) - Password_ID incremented by 1) - Password_Date - Hashed_Password - User_ID (which is read from the AspNetUser table) 9. The system will display an appropriate success message stating that the user's password stored in the database. An appropriate error message will be displayed to the user. Return to step 3 [ALT] Step 5: The new password does not match the user's password stored in the database. An appropriate error message. Return to step 3 [ALT] Step 6: The new password does not meet the requirements. The system will display an error message. Return to step 3 [ALT] Step 6: The new password does not meet the requirements. The system will display an error message. Return to step 3 [ALT] Step 7: The new password does not meet the requirements. The system will display an error message. Return to step 3 [ALT] Step 6: The new password does not meet the requirements. The system will display an error message. Return to step 3 [ALT] Step 6: The new password does not meet the requirements. The system will display an error message. Return to step 3 [ALT] Step 6: The new password date the requirements. The system will display an error message. Return to step 3 [ALT] Step 6: The new password date the user's password has been updated. The user's password has been inserted in the Password, History table and the user is password. The the was had the user is password. The the was had the user is pa			least one upper case
7. The system will validate that the new password entered in the field does not match the user's current password stored. The system will match the user's current password stored with the user's current password entered with the user's current password attribute, by comparing the User_ID attribute in AspNetUser table with the User_ID attribute in AspNetUser table with the User_ID attribute in the Password_History table. [ALT] 8. The system hashes the user's new password and creates a new record in the Password_History table. [ALT] 8. The system hashes the user's new password and creates a new record in the Password_History table, using the following attributes: - Password_ID(the value of which is the previous Password_ID incremented by 1) - Password_ID incremented by 1) - Password_Date - Hashed_Password - User_ID (which is read from the AspNetUser table) 9. The system will display an appropriate success message stating that the user's password has been updated. Alternate [ALT] Step 4: The information entered in the current password field is incorrect. The current password does not match the user's password stored in the database. An appropriate error message will be displayed to the user. Return to Step 3. [ALT] Step 5: The new password does not meet the requirements. The system will display an error message. Return to step 3 [ALT] Step 6: The new password does not meet the requirements. The system will display an error message. Return to step 3 [ALT] Step 6: The new password does not meet the requirements. The system will display an error message. Return to step 3 [ALT] Step 7: The new password does not meet the requirements. The system will display an error message. Return to step 3 [ALT] Step 7: The new password deen not meet the user's password saved in the database. The system will display an error message. Return to step 3 [ALT] Step 7: The new password entered matches the old password saved in the database. The system will display an error message. Return to step 3 [ALT] Step 7: The n			
current password entered with the user's current password stored in the Password_History table as the Hashed_Password attribute, by comparing the User_ID attribute in AspNetUser table with the User_ID attribute in the Password_History table. [ALT] 8. The system hashes the user's new password and creates a new record in the Password_History table, using the following attributes: - Password_Dit(the value of which is the previous Password_ID incremented by 1) - Password_Date - Hashed_Password - User_ID (which is read from the AspNetUser table) 9. The system will display an appropriate success message stating that the user's password has been updated. Alternate [ALT] Step 4: The information entered in the current password field is incorrect. The current password does not match the user's password stored in the database. An appropriate error message will be displayed to the user. Return to Step 3. [ALT] Step 5: The new password and confirm password fields do not match. The system will display an error message. Return to step 3 [ALT] Step 6: The new password does not meet the requirements. The system will display an error message. Return to step 3 [ALT] Step 7: The new password does not meet the requirements. The system will display an error message. Return to step 3 [ALT] Step 7: The new password does not meet the requirements. The system will display an error message. Return to step 3 [ALT] Step 7: The new password entered matches the old password saved in the database. The system will display an error message. Post-condition: The user's password has been inserted in the Password_History table and the user's password has been inserted in the password_History table and the user is able to login to the system will their new password.			 The system will validate that the new password entered in the field does not match the user's current
new password and creates a new record in the Password_History table, using the following attributes: - Password_ID(the value of which is the previous Password_ID incremented by 1) - Password_Date - Hashed_Password - User_ID (which is read from the AspNetUser table) 9. The system will display an appropriate success message stating that the user's password has been updated. Alternate [ALT] Step 4: The information entered in the current password field is incorrect. The current password does not match the user's password stored in the database. An appropriate error message will be displayed to the user. Return to Step 3. [ALT] Step 5: The new password and confirm password fields do not match. The system will display an error message. Return to step 3 [ALT] Step 6: The new password does not meet the requirements. The system will display an error message. Return to step 3 [ALT] Step 7: The new password does not meet the requirements. The system will display an error message. Return to step 3 [ALT] Step 7: The new password dees not meet the requirements. The system will display an error message. Return to step 3 [ALT] Step 5: The new password has been updated bear of the password has been updated. Conclusion: The use case concludes when the user's password history table and the user is able to login to the system with their new password.			current password entered with the user's current password stored in the Password_History table as the Hashed_Password attribute, by comparing the User_ID attribute in AspNetUser table with the User_ID attribute in the Password_History table.
Password_History table, using the following attributes: - Password_ID(the value of which is the previous Password_ID incremented by 1) - Password_Date - Hashed_Password - User_ID (which is read from the AspNetUser table) 9. The system will display an appropriate success message stating that the user's password has been updated. Alternate [ALT] Step 4: The information entered in the current password field is incorrect. The current password does not match the user's password stored in the database. An appropriate error message will be displayed to the user. Return to Step 3. [ALT] Step 5: The new password does not meet the requirements. The system will display an error message. Return to step 3 [ALT] Step 6: The new password does not meet the requirements. The system will display an error message avenue avenue avenue will display an error message. Return to step 3 [ALT] Step 6: The new password does not meet the requirements. The system will display an error message. Return to step 3 [ALT] Step 7: The new password entered matches the old password saved in the database. The system will display an error message. Return to step 3 [ALT] Step 7: The new password entered matches the old password saved in the database. The system will display an error message. Return to step 3 [ALT] Step 3: The new password has been updated been updated been updated been updated been updated been updated and the user is able to login to the system with their new password.			The system hashes the user's new password and creates a
which is the previous Password_ID incremented by 1) - Password_Date - Hashed_Password - User_ID (which is read from the AspNetUser table) 9. The system will display an appropriate success message stating that the user's password has been updated. Alternate [ALT] Step 4: The information entered in the current password field is incorrect. The current password does not match the user's password stored in the database. An appropriate error message will be displayed to the user. Return to Step 3. [ALT] Step 5: The new password and confirm password fields do not match. The system will display an error message. Return to step 3 [ALT] Step 6: The new password does not meet the requirements. The system will display an error message. Return to step 3 [ALT] Step 7: The new password entered matches the old password saved in the database. The system will display an error message. Return to step 3 [ALT] Step 7: The new password entered matches the old password saved in the database. The system will display an error message. Return to step 3 [ALT] Step 7: The new password has been updated Post-condition: The use case concludes when the user's password has been updated and the user is able to login to the system with their new password.			Password_History table, using the following attributes:
incremented by 1) Password_Date Hashed_Password User_ID (which is read from the AspNetUser table) 9. The system will display an appropriate success message stating that the user's password has been updated. Alternate Courses: [ALT] Step 4: The information entered in the current password field is incorrect. The current password does not match the user's password stored in the database. An appropriate error message will be displayed to the user. Return to Step 3. [ALT] Step 5: The new password and confirm password fields do not match. The system will display an error message. Return to step 3 [ALT] Step 6: The new password does not meet the requirements. The system will display an error message. Return to step 3 [ALT] Step 7: The new password entered matches the old password saved in the database. The system will display an error message. Return to step 3 [ALT] Step 7: The new password entered matches the old password saved in the database. The system will display an error message. Return to step 3 The use case concludes when the user's password has been updated Post-condition: The user's password has been inserted in the Password_History table and the user is able to login to the system with their new password.			which is the previous
- Hashed_Password - User_ID (which is read from the AspNetUser table) 9. The system will display an appropriate success message stating that the user's password has been updated. Alternate Courses: [ALT] Step 4: The information entered in the current password field is incorrect. The current password does not match the user's password stored in the database. An appropriate error message will be displayed to the user. Return to Step 3. [ALT] Step 5: The new password and confirm password fields do not match. The system will display an error message. Return to step 3 [ALT] Step 6: The new password does not meet the requirements. The system will display an error message. Return to step 3 [ALT] Step 7: The new password entered matches the old password saved in the database. The system will display an error message. Return to step 3 [ALT] Step 5: The new password entered matches the old password saved in the database. The system will display an error message. Conclusion: The use case concludes when the user's password has been updated Post-condition: The user's password has been inserted in the Password_History table and the user is able to login to the system with their new password.			incremented by 1)
Alternate Courses: Alternate Image: Course Image: Cour			 Hashed_Password
Alternate Courses: [ALT] Step 4: The information entered in the current password field is incorrect. The current password does not match the user's password stored in the database. An appropriate error message will be displayed to the user. Return to Step 3. [ALT] Step 5: The new password and confirm password fields do not match. The system will display an error message. Return to step 3 [ALT] Step 6: The new password does not meet the requirements. The system will display an error message. Return to step 3 [ALT] Step 7: The new password entered matches the old password saved in the database. The system will display an error message. Return to step 3 [ALT] Step 7: The new password entered matches the old password saved in the database. The system will display an error message. Return to step 3 Conclusion: The use case concludes when the user's password has been updated The user's password has been inserted in the Password_History table and the user is able to login to the system with their new password.			from the AspNetUser
Alternate [ALT] Step 4: The information entered in the current password field is incorrect. The current password does not match the user's password stored in the database. An appropriate error message will be displayed to the user. Return to Step 3. [ALT] Step 5: The new password and confirm password fields do not match. The system will display an error message. Return to step 3 [ALT] Step 6: The new password does not meet the requirements. The system will display an error message. Return to step 3 [ALT] Step 7: The new password entered matches the old password saved in the database. The system will display an error message. Return to step 3 [ALT] Step 7: The new password entered matches the old password saved in the database. The system will display an error message. Return to step 3 Conclusion: The use case concludes when the user's password has been updated The user's password has been inserted in the Password_History table and the user is able to login to the system with their new password.			9. The system will display an
Alternate Courses: [ALT] Step 4: The information entered in the current password field is incorrect. The current password does not match the user's password stored in the database. An appropriate error message will be displayed to the user. Return to Step 3. [ALT] Step 5: The new password and confirm password fields do not match. The system will display an error message. Return to step 3 [ALT] Step 6: The new password does not meet the requirements. The system will display an error message. Return to step 3 [ALT] Step 7: The new password entered matches the old password saved in the database. The system will display an error message. Return to step 3 Conclusion: The use case concludes when the user's password has been updated The user's password has been inserted in the Password_History table and the user is able to login to the system with their new password.			• • • • • • • • • • • • • • • • • • • •
incorrect. The current password does not match the user's password stored in the database. An appropriate error message will be displayed to the user. Return to Step 3. [ALT] Step 5: The new password and confirm password fields do not match. The system will display an error message. Return to step 3 [ALT] Step 6: The new password does not meet the requirements. The system will display an error message. Return to step 3 [ALT] Step 7: The new password entered matches the old password saved in the database. The system will display an error message. Return to step 3 Conclusion: The use case concludes when the user's password has been updated The user's password has been inserted in the Password_History table and the user is able to login to the system with their new password.			has been updated.
[ALT] Step 5: The new password and confirm password fields do not match. The system will display an error message. Return to step 3 [ALT] Step 6: The new password does not meet the requirements. The system will display an error message. Return to step 3 [ALT] Step 7: The new password entered matches the old password saved in the database. The system will display an error message. Return to step 3 Conclusion: The use case concludes when the user's password has been updated The user's password has been inserted in the Password_History table and the user is able to login to the system with their new password.		incorrect. The current password does not match the user's password stored in the database. An appropriate error message will be displayed to the user. Return to Step 3. [ALT] Step 5: The new password and confirm password fields do not	
[ALT] Step 6: The new password does not meet the requirements. The system will display an error message. Return to step 3 [ALT] Step 7: The new password entered matches the old password saved in the database. The system will display an error message. Return to step 3 Conclusion: The use case concludes when the user's password has been updated Post-condition: The user's password has been inserted in the Password_History table and the user is able to login to the system with their new password.			
[ALT] Step 7: The new password entered matches the old password saved in the database. The system will display an error message. Return to step 3 Conclusion: The use case concludes when the user's password has been updated Post-condition: The user's password has been inserted in the Password_History table and the user is able to login to the system with their new password.		[ALT] Step 6: The new passw	ord does not meet the requirements. The
saved in the database. The system will display an error message. Return to step 3 Conclusion: The use case concludes when the user's password has been updated Post-condition: The user's password has been inserted in the Password_History table and the user is able to login to the system with their new password.			•
Conclusion: The use case concludes when the user's password has been updated Post-condition: The user's password has been inserted in the Password_History table and the user is able to login to the system with their new password.		saved in the database. The sy	•
Post-condition: The user's password has been inserted in the Password_History table and the user is able to login to the system with their new password.	Conclusion:		the user's password has been updated
		The user's password has been	n inserted in the Password_History table
	Business Rules:		







Logical Narratives

Iteration 6 2022/08/01

Implementation Constraints and Specifications	 The password must be hashed before it is stored in the database. The user requires internet
Assumptions	None
Open Issues	N/A





2.1.4 Sale Subsystem

Table 2.1.4.1 Search Sale Item

Author(s): Luke Partridge Version: 1.0	Creation Date:12/04/2022 Last Review Date:	
Use Case Name:	Search Sales Item	Use case type
Use Case ID:	4.1	Business Requirements:
Priority:	HIGH	System Analysis:
Source:	Bester Strength and Conditioning	System Design: □
Primary Business Actor:	User	•
Primary System Actor:	None	
Other Participating Actors:	None	
Other Interested Stakeholders:	None	
Description:	item records on the syster requests to search for sale the information is displaye	he event where the user searches for sale m. The use case begins when the user e item records. The use case concludes when ed for the sales item records.
Pre-condition:	None	
Trigger:	The user requests to search Actor Action:	ch for sales item records on the system.
Typical Course	1. The user requests to search for sales item records on the system.	2. The system prompts the user to enter any a search criteria related to any of the sale item attributes
	The user enters the information as search criteria.	The system retrieves and displays a list of all the sale item records that matches with the search criteria entered. [ALT]
	5. The user selects the specific sale item record they wish to view from the displayed results.	6. The system retrieves and displays the following information regarding the selected sale item record from the Sale_Item table: - Sale_Item_Name - Sale_Item_Photo - Sale_Item_Description - Sale_Item_Quotable - Sale_Item_Price? - Sale_Item_Quantity? The Sale_Item_Price and Sale_Item_Quantity fields will not be populated with values, or shown to the user if the Sale_Item_Quotable field has a value of "true"



"ILOJI"
\mathcal{A}
(\bigcap)

Alternate Courses:	[ALT] Step 4: There are no matching search results. The use case terminates.
Conclusion:	The use case concludes with the searched sales item record and associated sale category name and description displayed successfully.
Post-condition:	The administrator can view the searched sales item record successfully.
Business Rules:	None
Implementation Constraints and Specifications	None
Assumptions	None
Open Issues	None



Table 2.1.4.2 Search Sales Category

Author(a): Luke Pertridge		Creation Data: 12/04/2022
Author(s): Luke Partridge		Creation Date:12/04/2022
Version: 1.0		Last Review Date:
Use Case Name:	Search Sale Category	Use case type
Use Case ID:	4.2	Business Requirements: □
Priority:	HIGH	System Analysis:
Source:	Bester Strength and Conditioning	System Design:
Primary Business Actor:	User	
Primary System Actor:	None	
Other Participating Actors:	None	
Other Interested Stakeholders:	None	
Description:	category records on the sy requests to search for sale when the information is dis	he event where the user searches for sale ystem. The use case begins when the user e category records. The use case concludes splayed for the sales category records.
Pre-condition:	None	
Trigger:		ch for sales category records on the system.
Typical Course	Actor Action:	system Response:
	The user requests to search for sales category records on the system. The administrator enters the required information as	The system prompts the user to enter any one of the following details concerning the sales category records from the Sale_Category table: Sale_Category_Name Sale_Category_Description 4. The system retrieves and displays a list of all the sale category records that matches with the search criteria entered. [ALT]
	5. The administrator selects the specific sale category record they wish to view from the displayed results.	6. The system retrieves and displays the following information regarding the sale category record from the Sale_Category table: - Sale_Category_Name - Sale_Category_Description



Alternate	[ALT] Step 4: There are no matching search results. The use case
Courses:	terminates.
Conclusion:	The use case concludes with the searched sale category record and associated sale item records with their Sale_Category_Name are displayed successfully.
Post-condition:	The administrator can view the searched sales category record successfully.
Business Rules:	None
Implementation Constraints and Specifications	None
Assumptions	None
Open Issues	None

27



Table 2.1.4.3 Add to Cart

Table 2.1.4.3 Add to Cart		
Author(s): Joshua Bester		Creation Date:20/06/2022
Version: 1.0		Last Review Date:
Use Case Name:	Add to Cart	Use case type
Use Case ID:	4.3	Business Requirements:
Priority:	HIGH	System Analysis:
Source:	Bester Strength and Conditioning	System Design:
Primary Business Actor:	User	
Primary System Actor:	None	
Other Participating Actors:	None	
Other Interested Stakeholders:	None	
Description:		use case starts where the user I like to add to their cart. The system information from the relevant entities case concludes when the item is
Pre-condition:	,	
Trigger:	The user requests to add a product	t to their cart
Typical Course	Actor Action:	system Response:
	The user requests to add a new product to their cart	2. The system will read through the Sale_Item table to find the record with the matching Sale_Item_ID (primary key) - Sale_Item_Name - Sale_Item_Price - Sale_Item_Stock - Sale_Item_Descripti on 3. The system verifies if there is more than 0 stock of the selected sale item by reading through the Sale_Item table to find the record with the matching Sale_Item_ID (primary key) and retrieving the Sale_Item_Stock attribute. [ALT] 4. The system decreases the Sale_Item_Stock attribute by 1 5. The system verifies that the Sale_Item_ID does not exist in the Sale_Line table [ALT] 6. The system will add the new sale line record to the

	Sale_Line table with the following attributes. - Sale_Line_ID(the value of which is the previous Employee_Type_ID incremented by 1) - Sale_Line_Quantity - Sale_Item_ID	
	7. The system displays a success notification to the user that the sale item has been added to the cart.	
Alternate	[ALT] Step 3: No items in the stock associated with the Sale_Item_ID.	
Courses:	The system will notify the user that there is no stock available	
	The use case terminates.	
	[ALT] Step 5: The Sale_Item_ID exists in the Sale_Line table. The system will update the sale line record by incrementing the Sale_Line_Quantity with 1. Return to step 7.	
Conclusion:	The use case concludes when the sale line details are added to the database and the system indicates with a message that the item were successfully added to the cart	
Post-condition:	The new sale line record has been added to the Sale_Line table	
Business Rules:	- All users of the system are able to add items to their carts	
Implementation Constraints and Specifications	None	
Assumptions	None	
Open Issues	None	



Table 2.1.4.4 View Cart

Table 2.1.4.4 View Ca		O
Author(s): Joshua Bester		Creation Date:20/06/2022
Version: 1.0	View Cort	Last Review Date:
Use Case Name:	View Cart	Use case type
Use Case ID:	4.4	Business Requirements:
Priority:	HIGH	System Analysis:
Source:	Bester Strength and Conditioning	System Design:
Primary Business Actor:	User	•
Primary System Actor:	None	
Other Participating Actors:	None	
Other Interested Stakeholders:	None	
Description: Pre-condition:	of their cart. The user requests reading the cart information from information as well as the total a	ent where the user would like to view the content to view their cart, the system then responds by m the relevant entities and displaying the amount due by the user. The use case e cart is successfully viewed by the user
Trigger:	The user requests to view their	cart
Typical Course	Actor Action:	System Response:
	The user requests to view the items in their cart	2. The system retrieves and displays a list of all the items in the cart which matches the user logged in The system will then search for the matching Sale_Line_ID in the Sale_Line table and retrieve the following information: - Sale_Line_Quantity - Sale_ID - Sale_Item_ID The system will then search for the matching Sale_Item_ID (primary key) in the Sale_Item table and retrieve the following information: - Sale_Item_Name - Sale_Item_Price (Using the foreign key Sale_Item_ID from the Sale_Line table) The system will then use the VAT_Percentage attribute to calculate the VAT of the total amount in the cart. The system will get the latest dated Vat percentage based on the VAT_Date attribute [ALT]
		The system will perform the following calculations, and display the





	4. The uper changes the	calculations at the bottom of the listed items in the cart
	4. The user changes the quantities of the sale_items in their cart [ALT]	5. The system invokes Use Case 4.5 <u>Edit</u> <u>Cart</u>
Alternate	[ALT] Step 2: No items in the cart associated with the User_ID. The system will	
Courses:	notify the user that the cart is empty.	
	The use case terminates.	
	[Al T] Stan 4: The user requests	s to check out the cart
	[ALT] Step 4: The user requests to check out the cart. The system invokes Use Case 4.6 "Checkout Cart".	
	The use case terminates.	
Conclusion:	The system displays a list with a cart	all the sale item records and their quantity in the
Post-condition:	All cart information is successful	lly displayed to the user
Business Rules:	 All users of the system c 	an view their cart
Implementation Constraints and Specifications	None	
Assumptions	None	
Open Issues	None	



Table 2.1.4.5 Edit Cart

Table 2.1.4.5 Edit Cart		
Author(s): Joshua Bester		Creation Date:20/06/2022
Version: 1.0		Last Review Date:
Use Case Name:	Edit Cart	Use case type
Use Case ID:	4.5	Business Requirements:
Priority:	HIGH	System Analysis:
Source:	Bester Strength and Conditioning	System Design: □
Primary Business Actor:	User	
Primary System Actor:	None	
Other Participating Actors:	None	
Other Interested Stakeholders:	None	
Description:		
Pre-condition:	 There must be sale items i 	n the cart
Trigger:		
Typical Course	Actor Action:	system Response:
	The user requests to update the quantities of the sale items listed in the cart	2. The system verifies if there is more than 0 stock of the sale item by reading through the Sale_Item table to find the record with the matching Sale_Item_ID (primary key) and retrieving the Sale_Item_Stock attribute. [ALT] 3. The system increases or
		decreases the Sale_Item_Stock attribute depending on the what the user requests 4. The system updates the
		quantity of the sale item by decreasing or increasing the quantity based on what the user requests
		 The system verifies that the edited quantity of the sale item is above 0 [ALT]
		6. The system reads the Sale_Item_Price from the Sale_Item table regarding the all the sale items in the cart
		 7. The system recalculates the following amounts Cart Subtotal VAT exclusive (Cart Total minus the vat amount) VAT amount (The Cart total vat inclusive multiplies with



	Vat_percentage last date retrieved from the	
	VAT entity	
	Cart Total Vat inclusive	
	(The total of each	
	Sale Item price multiply	
	by the quantity selected	
	by the user)	
Alternate Courses:	[ALT] Step 2: The sale_item_stock attribute is 0, the sale item is not available. The system notifies the user that no stock is available. The	
	use case terminates	
	[ALT] Step 5: The quantity is 0, the sale item is removed from the cart	
	information. Return to step 6	
Conclusion:	The use case concludes when the quantity next to the sale item has	
	been updated, and is displayed to the user	
Post-condition:	The subtotal, vat and cart total amounts have been recalculated and	
	displayed to the user	
Business Rules:	- All users can edit their own cart	
Implementation Constraints and	None	
Specifications		
Assumptions	None	
Open Issues	None	



Table 2.1.4.7 View Shop

Table 2.1.4.7 View Shop		
Author(s): Joshua Bester Version: 1.0		Creation Date:20/06/2022 Last Review Date:
Use Case Name:	View Shop	Use case type
Use Case ID:		Business Requirements:
	4.7	
Priority:	HIGH	System Analysis:
Source:	Bester Strength and Conditioning	System Design:
Primary Business Actor:	User	
Primary System Actor:	None	
Other Participating Actors:	None	
Other Interested Stakeholders:	None	
Description:	This use case describes the event where the user wants to access the shop page. The system will display the page; allowing the user to view sale items, add to cart and request for quote	
Pre-condition:		
Trigger:	The user requests to view the shop	
Typical Course	Actor Action:	system Response:
	The administrator requests to view the shop page 3 The user requests to filter.	2. The system will read information regarding each item on the shop screen with their - Sale_Item_Name - Sale_Ite_Photo - Sale_Item_Price - Sale_Item_Quaotable Retrieved from the Sale_Item table The system will display the retrieved information in cards for the user to view. If the Sale_Item_Quotable attribute is "True" the Sale_Item_Price attribute will not be enabled for that specific sale item
	3. The user requests to filter the items in the shop [ALT]	4. Invoke Use Case 4.8 Filter Shop
Alternate	[ALT] Step 3a: The user requests	to add an item to their cart.
Courses:	The system invokes Use Case 4.3 "Add to Cart".	
	[ALT] Step 3b: The requests for a The system invokes Use Case 4.8	•
Conclusion:	The system displays a list with all the sale item records and their information.	
Post-condition:	The administrator will be able to filter on the shop page.	
Business Rules:	- All users are able to view the shop	
Implementation Constraints and Specifications	None	
Assumptions	None	
Open Issues	None	



Table 2.1.4.10 Create Refund Reason

Table 2.1.4.10 Create Refun	u Reason		
Author(s): Lyne' Keet		Creation Date:20/06/2022	
Version: 1.0		Last Review Date:	
Use Case Name:	Create Refund Reason	Use case type	
Use Case ID:	4.10	Business Requirements:	
Priority:	HIGH	System Analysis:	
Source:	Bester Strength and Conditioning	System Design:	
Primary Business Actor:	Administrator		
Primary System Actor:	None		
Other Participating Actors:	None		
Other Interested Stakeholders:	None		
Description:	This use case describes the event where the administrator creates a new refund reason on the system. The use case begins when the administrator wants to create a new refund reason record, and enters the needed information required by the system. The system verifies the information and then stores it. The use case concludes when the refund reason record has successfully been created on the system.		
Pre-condition:		st be logged into the system	
Trigger:		create a new refund reason on the system	
Typical Course	Actor Action:	system Response:	
	The administrator requests to create a new refund reason record on the system	The system prompts the administrator to enter the following details concerning the refund reason from the Refund_Reason table Refund_Reason_Description	
	The administrator enters the required information in the field provided	4. The system validates that the Description input field is not left empty. [ALT]	
		5. The system displayed the entered information and requests the administrator to read and confirm the entered information	
	6. The administrator confirms the information entered [ALT]	7. The system verifies that the refund reason record is not duplicated in the Refund_Reason table by checking if the attribute - Refund_Reason_Description Does not match any existing refund reason record [ALT]	
		8. The system will add the new refund reason record to the Refund_Reason table with the following attributes - Refund_Reason_ID(the value of which is the previous Refund_Reason_ID incremented by 1)	



dol b	Strengthening	Solutions
	Team 7	

		<u> </u>
		 Refund Reason Description
		The system displayed a success
		notification to the administrator that
		the refund reason has been created
Alternate	[ALT] Step 4: All input fields a	re not filled in or the required fields are not in
Courses:		n displays error messages under the input
	fields that are empty or invalid	. Return to step 3.
	[ALT] Step 6: The administrat	or denies the information entered is correct.
	Return to step 3	
	[ALT] Step 7: The refund reas	on record already exists on the system, the
	system displays an error mess	sage informing the administrator. Return to
	step 5.	
Conclusion:	The use case concludes wher	the refund reason details are added to the
	database and the system indic	cates with a message that the details were
	added successfully	
Post-condition:	The new refund reason record	has been added to the system in the
	Refund_Reason table	
Business Rules:	-	
Implementation Constraints and	None	
Specifications	None	
Assumptions	None	
Open Issues	None	



Table 2.1.4.11 Search Refund Reason

Use Case Name: Search Refund Reason Use case type		rch Refund Reason	
Use Case Name: Use Case ID: 4.11 Priority: HIGH Source: Bester Strength and Conditioning Primary Business Actor: Primary System Actor: Other Participating Actors: Other Participating Actors: Description: This use case describes the event where the administrator searches for a refund reason on the system. The use case begins when the administrator selects the option to search for a refund reason. The use case concludes when the information is displayed for that specific refund reason. The administrator must be logged into the system. Trigger: The administrator requests to search a refund reason on the system search for a refund reason on the system. Actor Action: The administrator requests to search a refund reason on the system search for a refund reason on the system. Trigger: The administrator requests to search a refund reason on the system search for a refund reason on the system search for a refund reason on the system. Trigger: The administrator requests to search a refund reason on the system search for a refund reason on the system search for a refund reason on the system prompts the administrator to enter the following details concerning the refund reason from the Refund Reason_Description from the Refund Reason table: - Refund Reason_Description from the Refund Reason table: - Refund Reason_Description from the Refund Reason Description from the Refund Reason table: - Refund Reason Description from the Refund Reason Description from the Refund Reason table: - Refund Reason Description from the Refund Reason table: - Refund Reason table:	Author(s): Lyne' Keet		Creation Date:20/06/2022
Use Case ID: Priority:	Version: 1.0		Last Review Date:
Priority: HIGH Source: Bester Strength and Conditioning System Design: Description: None Other Participating Actors: None Other Participating Actors: None Other Participating Actors: None Description: This use case describes the event where the administrator searches for a refund reason on the system. The use case begins when the administrator selects the option to search for a refund reason. The use case concludes when the information is displayed for that specific refund reason. Pre-condition: The administrator must be logged into the system. Trigger: The administrator requests to search a refund reason on the system Actor Action: System Response: 1. The administrator requests to search a refund reason on the system (Dlowing details concerning the refund reason from the system. 1. The administrator requests to search a refund reason on the system (Partind Reason table: Refund, Reason	Use Case Name:	Search Refund Reason	
Source: Bester Strength and Conditioning System Design: Primary Business Actor: Administrator Primary System Actor: None Other Participating Actors: Other Interested Stakeholders: None Description: This use case describes the event where the administrator searches for a refund reason on the system. The use case begins when the administrator selects the option to search for a refund reason. The use case concludes when the information is displayed for that specific refund reason. Pre-condition: • The administrator must be logged into the system. Trigger: The administrator requests to search a refund reason on the system Paston: System Response: 2. The system prompts the administrator on the system Paston and displays a list of all the refund reason from the system. 3. The administrator enters the required information as search criteria. 3. The administrator enters the required information as search criteria. 5. The administrator selects the specific refund reason they wish to view from the displayed results. 6. The system retrieves and displays the following information regarding the refund reason from the Refund Reason table: - Refund_Reason_Description from the Refund_Reason table: - Refund_	Use Case ID:	4.11	Business Requirements:
Primary Business Actor: Administrator Primary System Actor: None Other Participating Actors: Other Interested Stakeholders: Description: This use case describes the event where the administrator searches for a refund reason on the system. The use case begins when the administrator selects the option to search for a refund reason. The use case concludes when the information is displayed for that specific refund reason. Pre-condition: Trigger: The administrator requests to search a refund reason on the system. Actor Action: System Response: 2. The system prompts the administrator to enter the following deals concerning the refund reason on the system. 1. The administrator requests to search a refund reason from the Refund_Reason table: Refund_Reason table: Refund_Reason_Descript ion. 3. The administrator enters the required information as search criteria. 5. The administrator selects the specific refund reason they wish to view from the displayed results. Alternate Courses: [ALT] Step 4: No results were found. The use case terminates Conclusion: The administrator can view the searched refund reason being displayed successfully. Post-condition: The administrator can view the searched refund reason successfully. Post-condition: The administrator can view the searched refund reason arefund refund reason arefund reason and the system will be able to search a refund.	Priority:	HIGH	System Analysis:
Primary System Actor: Other Participating Actors: Other Interested Stakeholders: Obescription: This use case describes the event where the administrator searches for a refund reason on the system. The use case begins when the administrator selects the option to search for a refund reason. The use case concludes when the information is displayed for that specific refund reason. Pre-condition:	Source:	Bester Strength and Conditioning	System Design: □
None	Primary Business Actor:	Administrator	
None	Primary System Actor:	None	
None Stakeholders: Description: This use case describes the event where the administrator searches for a refund reason on the system. The use case begins when the administrator selects the option to search for a refund reason. The use case concludes when the information is displayed for that specific refund reason.		Nama	
Stakeholders: None		None	
Stakeholders:	Other Interested	Nana	
reason on the system. The use case begins when the administrator selects the option to search for a refund reason. The use case concludes when the information is displayed for that specific refund reason. Pre-condition: • The administrator must be logged into the system. • The title must exist on the system. Trigger: The administrator requests to search a refund reason on the system Response: 2. The system prompts the administrator to enter the following details concerning the refund reason from the Refund_Reason_Descript ion. 1. The administrator requests to search for a refund reason on the system refund reason from the Refund_Reason_Descript ion. 4. The system retrieves and displays a list of all the refund reasons with the required information as search criteria. 3. The administrator enters the required information as search criteria. 6. The system retrieves and displays the following information regarding the refund reason from the Refund_Reason_Descript ion. 5. The administrator selects the specific refund reason they wish to view from the displayed results. 6. The system retrieves and displays the following information regarding the refund reason from the Refund_Reason table: - Refund_Reason_Descript ion Alternate Courses: Conclusion: [ALT] Step 4: No results were found. The use case terminates Conclusion: The use case concludes with the searched refund reason being displayed successfully. Post-condition: The administrator can view the searched refund reason successfully. Only authorised users of the system will be able to search a refund	Stakeholders:	None	
option to search for a refund reason. The use case concludes when the information is displayed for that specific refund reason. The administrator must be logged into the system. The administrator requests to search a refund reason on the system. The administrator requests to search a refund reason on the system. Actor Action: System Response: 2. The system prompts the administrator to enter the following details concerning the refund reason from the Refund_Reason_Descript ion. 4. The system retrieves and displays a list of all the refund reason swith the required information as search criteria. The administrator enters the required information as search criteria. The administrator selects the specific refund reason they wish to view from the displayed results. Alternate Courses: Conclusion: The administrator can view the searched refund reason successfully. Post-condition: The administrator can view the searched refund reason successfully. Only authorised users of the system will be able to search a refund	Description:		
Information is displayed for that specific refund reason.		reason on the system. The use case be	egins when the administrator selects the
Pre-condition: Trigger: The administrator must be logged into the system. The title must exist on the system. The administrator requests to search a refund reason on the system Actor Action: System Response: 2. The system prompts the administrator to enter the following details concerning the refund reason from the Refund_Reason_Descript ion. 3. The administrator enters the required information as search criteria. 3. The administrator enters the required information as search criteria. 5. The administrator selects the specific refund reason they wish to view from the displayed results. Alternate Courses: Conclusion: The use case concludes with the searched refund reason being displayed successfully. Post-condition: The administrator can view the searched refund reason successfully. Only authorised users of the system will be able to search a refund		option to search for a refund reason. T	he use case concludes when the
Trigger: The administrator requests to search a refund reason on the system Actor Action: System Response: 2. The system prompts the administrator to enter the following details concerning the refund reason from the system. 1. The administrator requests to search for a refund reason on the system. Prefund Reason table: Refund Reason table: Refund Reason Description. 3. The administrator enters the required information as search criteria. Search criteria. 5. The administrator selects the specific refund reason they wish to view from the displayed results. Alternate Courses: Conclusion: The use case concludes with the searched refund reason being displayed successfully. Post-condition: The administrator can view the searched refund reason successfully. The administrator can view the searched refund reason successfully. Only authorised users of the system will be able to search a refund			
Trigger: Typical Course Actor Action: System Response: 2. The system prompts the administrator requests to search for a refund reason on the system. 1. The administrator requests to search for a refund reason on the system. 1. The administrator requests to search for a refund reason on the system. 1. The administrator requests to search for a refund reason on the system. 2. The system prompts the administrator to enter the following details concerning the refund reason from the Refund_Reason_Descript ion. 3. The administrator enters the required information as search criteria. 3. The administrator enters the required information as search criteria. 4. The system retrieves and displays a list of all the refund reasons with the Refund_Reason_Description from the Refund_Reason_Description from the Refund_Reason table that matches the search criteria entered. [ALT] 5. The administrator selects the specific refund reason they wish to view from the displayed results. 5. The administrator selects the specific refund reason they wish to view from the displayed results. 6. The system retrieves and displays the following information regarding the refund reason from the Refund_Reason_Descript ion Alternate Courses: Conclusion: [ALT] Step 4: No results were found. The use case terminates The use case concludes with the searched refund reason being displayed successfully. Post-condition: The administrator can view the searched refund reason successfully. - Only authorised users of the system will be able to search a refund	Pre-condition:	_	· ·
Typical Course Actor Action: System Response:			
2. The system prompts the administrator to enter the following details concerning the refund reason from the system. 1. The administrator requests to search for a refund reason on the system. 2. The system prompts the administrator to enter the following details concerning the refund reason from the Refund_Reason_Descript ion. 3. The administrator enters the required information as search criteria. 4. The system retrieves and displays a list of all the refund reasons with the Refund_Reason_Description from the Refund_Reason_Description from the Refund_Reason table: 5. The administrator selects the specific refund reason they wish to view from the specific refund reason they wish to view from the displayed results. 6. The system retrieves and displays the following information regarding the refund reason from the Refund_Reason_Descript ion Alternate Courses: Conclusion: [ALT] Step 4: No results were found. The use case terminates [ALT] Step 4: No results were found. The use case terminates The use case concludes with the searched refund reason being displayed successfully. Post-condition: The administrator can view the searched refund reason successfully. - Only authorised users of the system will be able to search a refund	Trigger:	The administrator requests to search a	refund reason on the system
1. The administrator requests to search for a refund reason on the system. 1. The administrator requests to search for a refund reason on the system. 1. The administrator requests to search for a refund reason on the system. 1. The administrator requests to search criteria. 2. Refund_Reason_Descript ion. 3. The administrator enters the required information as search criteria. 3. The administrator enters the required information as search criteria. 4. The system retrieves and displays a list of all the refund reasons with the Refund_Reason_Description from the Refund_Reason_Description from the Refund_Reason criteria entered. [ALT] 5. The administrator selects the specific refund reason they wish to view from the displayed results. 6. The system retrieves and displays the following information regarding the refund reason from the Refund_Reason_Descript ion 4. The system retrieves and displays the following information regarding the refund reason from the Refund_Reason table: 6. The system retrieves and displays the following information regarding the refund reason from the Refund_Reason_Descript ion Alternate Courses: [ALT] Step 4: No results were found. The use case terminates Conclusion: The use case concludes with the searched refund reason being displayed successfully. Post-condition: The administrator can view the searched refund reason successfully. The administrator can view the system will be able to search a refund	Typical Course	Actor Action:	system Response:
displays a list of all the refund reasons with the Refund_Reason_Description from the Refund_Refund table that matches the search criteria entered. [ALT] 5. The administrator selects the specific refund reason they wish to view from the specific refund reason they wish to view from the displayed results. Alternate Courses: Conclusion: Alternate Courses: Conclusion: The use case concludes with the searched refund reason being displayed successfully.		search for a refund reason on	administrator to enter the following details concerning the refund reason from the Refund_Reason table: - Refund_Reason_Descript
5. The administrator selects the specific refund reason they wish to view from the displayed results. Alternate Courses: Conclusion: The use case concludes with the searched refund reason being displayed resulty. Post-condition: The administrator selects the displays the following information regarding the refund reason from the Refund_Reason table: Refund_Reason_Descript ion The use case terminates The use case concludes with the searched refund reason being displayed successfully. Post-condition: The administrator can view the searched refund reason successfully. Only authorised users of the system will be able to search a refund		required information as	displays a list of all the refund reasons with the Refund_Reason_Description from the Refund_Refund table that matches the search criteria
Courses: Conclusion: The use case concludes with the searched refund reason being displayed successfully. Post-condition: The administrator can view the searched refund reason successfully. Business Rules: - Only authorised users of the system will be able to search a refund		specific refund reason they wish to view from the	displays the following information regarding the refund reason from the Refund_Reason table: - Refund_Reason_Descript
Conclusion: The use case concludes with the searched refund reason being displayed successfully. Post-condition: The administrator can view the searched refund reason successfully. Business Rules: Only authorised users of the system will be able to search a refund		[ALT] Step 4: No results were found. 7	The use case terminates
Post-condition: The administrator can view the searched refund reason successfully. Business Rules: - Only authorised users of the system will be able to search a refund	Conclusion:		
Business Rules: - Only authorised users of the system will be able to search a refund	Post-condition:		
reason (an auministrator).			





Logical Narratives

Iteration 6 2022/08/01

Implementation Constraints and Specifications	None
Assumptions	None
Open Issues	None



Table 2.1.4.12 Update Refund Reason

· ·	e Refund Reason	
Author(s): Lyne' Keet		Creation Date:20/06/2022
Version: 1.0		Last Review Date:
Use Case Name:	Update Refund Reason	Use case type
Use Case ID:	4.12	Business Requirements:
Priority:	HIGH	System Analysis:
Source:	Bester Strength and Conditioning	System Design:
Primary Business Actor:	Administrator	
Primary System Actor:	None	
Other Participating Actors:	None	
Other Interested Stakeholders:	None	
Description:	update a specific refund reason. The	where the administrator would like to e use case begins with the administrator son. The use case concludes when the d the new details are saved on the
Pre-condition:	The administrator must be loThe refund reason must exis	
Trigger:	The administrator requests to update	e a refund reason on the system.
Typical Course	Actor Action:	system Response:
	The administrator requests to update a refund reason on the system.	The system retrieves and displays the following information regarding the refund reason record from the Refund_Reason table: Refund_Reason_Descrip tion.
		3. The system will prompt the administrator to edit pre-populated fields with the information that was retrieved from the Refund_Reason table.
	4. The administrator will update the relevant information of the refund reason and select the option to save the changes made.	 The system verifies that the required data fields are not empty. [ALT]
		6. The system displays the new entered information and requests the administrator to read and confirm the changes that are to be made.
	7. The administrator confirms the updates made to the refund reason record. [ALT]	8. The system verifies that the refund reason record is not being duplicated in the Refund_Reason table by confirming that the entered attributes:



Strengthening	Solutions
Team 7	

		
		- Refind_Reason_Descripti on
		Do not match any existing refund
		reason record.
		[ALT]
		The system saves the relevant
		changes to the
		Refund_Reason_Description in
		the Refund_Reason table.
		10. The system notifies the
		administrator that the changes
		have been successfully made.
Alternate	[ALT] Step 5: The required fields are not entered. The system displays an	
Courses:	error notification informing the admir	
		nies the updates to be made. The use
	case terminates	waatab ayiatiga wafi wad waasay waxayda
		s match existing refund reason records.
Conclusion:		forming the administrator. Return to step
Conclusion.	refund reason changes have been s	ystem notifies the administrator that the
Post-condition:		ssfully updated in the Refund_Reason
r ost-condition.	table in the system's database.	ssidily apadied in the Kerding_Keason
Business Rules:		system will be able to update a refund
Dusiness ixules.	reason (an administrator).	system will be able to update a return
Implementation Constraints	reason (an administrator).	
and Specifications	None	
Assumptions	None	
Open Issues	None	
O P 011 100000	1 10110	



Table 2.1.4.13 Delete Refund Reason

	te Refund Reason	O 1: D 1 00/00/0000
Author(s): Lyne' Keet		Creation Date:20/06/2022
Version: 1.0	I	Last Review Date:
Use Case Name:	Delete Refund Reason	Use case type
Use Case ID:	4.13	Business Requirements:
Priority:	HIGH	System Analysis:
Source:	Bester Strength and Conditioning	System Design:
Primary Business Actor:	Administrator	
Primary System Actor:	None	
Other Participating Actors:	None	
Other Interested Stakeholders:	None	
Description:	specific refund reason record. The us	n record. The use case concludes when the
Pre-condition:	 The administrator must be log The refund reason record must 	
Trigger:		a refund reason record on the system.
Typical Course	Actor Action:	System Response:
	The administrator requests to delete a refund reason record on the system.	The system will confirm no refund records are associated with the selected Refund_Reason_ID as a FK in the Refund table. [ALT] The system will retrieve and display the:
		The system prompts the administrator to confirm the deletion of the refund reason record The system deletes the refund
	The administrator confirms the deletion of the refund reason record. [ALT]	reason record from the Refund_Reason table with the following attribute: Refund_Reason_Description.
		7. The system notifies the administrator that the refund reason record has been successfully deleted.
Alternate Courses:	[ALT] Step 2: The system determines there are return record/s associated with the selected refund reason record. The system will notify the administrator, and ask them to ensure no return records are associated before deletion. The use case terminates. [ALT] Step 5: The administrator denies the deletion of the refund reason record.	
Conclusion:	The use case terminates. The use case concludes when the system notifies the administrator that the refund reason record has been successfully deleted from the system.	
	1 1000011 100010 1100 Decit 30000331011y	aciotoa iroin tilo oyotoin.





Post-condition:	The refund reason record is deleted from the Refund_Reason table in the system's database
Business Rules:	Only authorised users of the system will be able to delete a refund reason record (an administrator).
Implementation Constraints and Specifications	None
Assumptions	None
Open Issues	None

42



Table 2.1.4.14 View Refund Reason

Author(s): Lyne' Keet Version: 1.0		Creation Date:20/06/2022 Last Review Date:
Use Case Name:	View Refund Reason	Use case type
Use Case ID:	4.14	Business Requirements:
Priority:	HIGH	System Analysis:
Source:	Bester Strength and Conditioning	System Design: □
Primary Business Actor:	Administrator	•
Primary System Actor:	None	
Other Participating Actors:	None	
Other Interested Stakeholders:	None	
Description:	for refund reasons.	e system will display the page; s the create, search, edit and delete
Pre-condition:	The administrator must be logged i	•
Trigger:	The administrator requests to view	
Typical Course	Actor Action:	system Response:
	The administrator requests to view the refund reason page. The administrator	The system will display the populated table on the screen with the: Refund_Reason_De scription from the retrieved Refund_Reason records The system invokes Use
	requests to search a	Case 4.11 "Search Refund
Alternate Courses:	refund reason record. [ALT] Step 3a: The administrator record. The system invokes Use Case 4.10 [ALT] Step 3b: The administrator records. The system invokes Use Case 4.12 [ALT] Step 3c: The administrator records. The system invokes Use Case 4.13	equests to update refund reason "Update Refund Reason". equests to delete refund reason "Delete Refund Reason".
Conclusion:	The system displays a table with al their information.	I the refund reasons records and
Post-condition:	The administrator will be able to se	arch on the refund reason page
Business Rules:	 Only authorised users of the page (an administrator) 	e system can view the refund reason
Implementation Constraints and Specifications	None	
Assumptions	None	
Open Issues	None	





2.1.5 Client/Member Subsystem

Table 2.1.5.1 Register Client

Table 2.1.5.1 Register Clien		0 1: D 1 00/00/0000
Author(s): Lyne' Keet		Creation Date:20/06/2022
Version: 1.0		Last Review Date:
Use Case Name:	Register Client	Use case type
Use Case ID:	5.1	Business Requirements:
Priority:	HIGH	System Analysis:
Source:	Bester Strength and Conditioning	System Design: □
Primary Business Actor:	Potential Client (PBA)	
Primary System Actor:	None	
Other Participating Actors:	None	
Other Interested Stakeholders:	None	
Description:	register themselves on The BSC sy	where a potential client requests to ystem. The use case begins when the ter on the system. The system will prompt
	the potential client to enter their inf their details, where it will be valida	ormation. The potential client will enter
Pre-condition:		
Trigger:	The potential customer wishes to re	egister themselves onto the BSC system
Typical Course	Actor Action:	system Response:
	The potential client requests to add register themselves onto the BSC system	2. The system prompts the client to enter the following details - FirstName - LastName - PhoneNumber - Title - Email - Password concerning the client from the AspNetUsers table And the - Title_Name From the Title table
	3. The user enters the required information in the respective fields, and submits their details	4. The system validates the information entered - all the input fields are not left empty - Phone number consists of 10 digits - Email consists a "@' sign - The password length is more than 8 characters - The password consists of 1 special character - The password consists of one capital letter



	- Confirms that the password and confirm password fields are matched
	[ALT]
	5. The system hashed the
	password entered by the client 6. The system will add the new
	client record to the AspNetUsers table with the following attributes - Id(the value of which is the previous Id incremented by 1) - FirstName - LastName - PhoneNumber - Email - Title - Password (Hashed
	password in step 5)
	7. The system assigns the user a "client" role using the AspNetRoles table matching the
	Name attribute, and assigning the "Client" role id in the AspNetUserRoles table with the associated User ID table
	8. The system displays a success
	notification to the client that the has been successfully registered.
Alternate	[ALT] Step 4: All input fields are not filled in or the required fields are not in
Courses:	the correct format. The system displays error messages under the input fields that are empty or invalid. Return to step 3.
Conclusion:	The client information is stored in the AspNetUsers table, and the "client"
	user role is assigned
Post-condition:	The user is redirected to the login screen, and can successfully log in
Business Rules:	Only clients are created through the register page
Implementation Constraints and Specifications	None
Assumptions	None
Open Issues	None



2.1.7 Booking Subsystem

Table 2.1.7.1 Create Schedule

A		O
Author(s): Shannon Noel		Creation Date:20/06/2022
Version: 1.0		Last Review Date:
Use Case Name:	Create Schedule	Use case type
Use Case ID:	7.1	Business Requirements:
Priority:	HIGH	System Analysis:
Source:	Bester Strength and Conditioning	System Design: □
Primary Business Actor:	Administrator	
Primary System Actor:	None	
Other Participating Actors:	None	
Other Interested Stakeholders:	None	
Description:	This use case describes the event new schedule on the system. The cadministrator wants to create a new needed information required by the information and then stores it. The schedule record has successfully be	use case begins when the v schedule record, and enters the system. The system verifies the use case concludes when the
Pre-condition:	 The administrator must be I 	ogged into the system
Trigger:	The administrator requests to creat	te a new schedule on the system
Typical Course	Actor Action:	system Response:
	The administrator requests to create a new schedule record on the system	2. The system prompts the administrator to enter the following details concerning the schedule from the Schedule table - Venue_Name - Booking_Type_Nam e - Lesson_Name - Date_Session - Date_session _ StartTime - Date_Sessio n_EndTime - Employee_Name
	The administrator enters the required information in the fields provided	4. The system validates that the input fields are not left empty. The system validates the following fields to ensure that valid inputs are entered The date selected does not precede the current date at



	the time of creation [ALT]
	The system displayed the entered information and requests the administrator to read and confirm the entered information The system will add the new
6. The administrator confirms the information entered [ALT]	schedule table with the following attributes - Schedule_ID(the value of which is the previous Schedule_ID incremented by 1) - Venue_ID (which is retrieved from the Venue_Name; selected from the dropdown list) - Booking_Type_ID (which is retrieved from the Booking_Type_ID (which is retrieved from the Booking_Type_Nam e; selected from the dropdown list) - Lesson_ID (which is retrieved from the Lesson_Name; selected from the dropdown list) - Lesson_ID (which is retrieved from the Lesson_Name; selected from the dropdown list) - Date_Session_ID(which is retrieved from the dropdown list) - Date_Session_StartTime and Date_Session_EndTime; selected from the dropdown list) - Employee_ID (which is retrieved from the dropdown list) - Employee_ID (which is retrieved from the dropdown list) - Employee_Name; selected from the Employee_Name; selected from the dropdown list)



J [⊕]	Strengthening	Solutions
)	Team 7	

	8. The system displayed a success notification to the administrator that the schedule event record has been created
Alternate	[ALT] Step 4: All input fields are not filled in or the required fields are
Courses:	not in the correct format. The system displays error messages under
	the input fields that are empty or invalid. Return to step 3.
	[ALT] Step 6: The administrator denies the information entered is correct. Return to step 3
Conclusion:	The use case concludes when the schedule details are added to the database and the system indicates with a message that the details were added successfully
Post-condition:	The new schedule record has been added to the system in the
	Schedule table
Business Rules:	-
Implementation Constraints and Specifications	None
Assumptions	None
Open Issues	None



Table 2.1.7.2 Search Schedule

Table 2.1.7.2 Search Sche	dule	0 (' D) 00/00/0000
Author(s): Shannon Noel		Creation Date:20/06/2022
Version: 1.0		Last Review Date:
Use Case Name:	Search Schedule	Use case type
Use Case ID:	7.2	Business Requirements:
Priority:	HIGH	System Analysis:
Source:	Bester Strength and Conditioning	System Design:
Primary Business Actor:	Administrator	
Primary System Actor:	None	
Other Participating Actors:	None	
Other Interested Stakeholders:	None	
Description:	a schedule on the system. The use selects the option to search for a swhen the information is displayed f	or that specific schedule.
Pre-condition:	The administrator must be IThe schedule must exist on	•
Trigger:	The administrator requests to sear	
Typical Course	Actor Action:	system Response:
	The administrator requests to search for a schedule on the system.	The system prompts the administrator to enter the following details concerning the schedule from the Schedule table:
	The administrator enters the required information as search criteria.	4. The system retrieves and displays a list of all the schedules with the Date_Session_Start and Date_Session_End from the Date_Session table that matches the search criteria entered. [ALT]
	5. The administrator selects the specific schedule they wish to view from the displayed results.	6. The system retrieves and displays the following information regarding the schedule from the Schedule table: - Venue_Name - Booking_Type_Nam e - Lesson_Name - Date_Session - Employee_Name
Alternate Courses:	[ALT] Step 4: No results were four	
Conclusion:	The use case concludes with the retrieved schedule being displayed successfully.	
Post-condition:	The administrator can view the retrieved schedule successfully.	
	The definition of the title for	







Logical Narratives

Iteration 6 2022/08/01

Business Rules:	 Only authorised users of the system will be able to search a schedule (an administrator).
Implementation Constraints and Specifications	None
Assumptions	None
Open Issues	None





Table 2.1.7.3 Update Schedule

Table 2.1.7.3 Update Schedule		
Author(s): Shannon Noel	Creation Date:20/06/2022	
Version: 1.0		Last Review Date:
Use Case Name:	Update Schedule	Use case type
Use Case ID:	7.3	Business Requirements:
Priority:	HIGH	System Analysis:
Source:	Bester Strength and Conditioning	System Design:
Primary Business Actor:	Administrator	
Primary System Actor:	None	
Other Participating Actors:	None	
Other Interested Stakeholders:	None	
Description: Pre-condition:	update a specific schedule. The us requesting to update the schedule schedule has been updated and th system.	
Pre-condition.	The administrator must beThe schedule must exist or	
Trigger:	The administrator requests to upda	
Typical Course	Actor Action:	system Response:
	The administrator requests to update a schedule on the system.	2. The system retrieves and displays the following information regarding the schedule record from the Schedule table: - Venue_Name - Booking_Type_Nam e - Lesson_Name - Date_Session - Date_Sessio n_Start_Time - Date_Sessio n_End_Time - Employee_Name
	4. The administrator will update the relevant information of the schedule and select the option to save the changes made.	3. The system will prompt the administrator to edit pre-populated fields with the information that was retrieved from the Schedule table. 5. The system verifies that the required data fields are not empty. [ALT]
		The system displays the new entered information and requests the



11 - 11 0	
Фп≂пФ	
77	
\mathcal{L}	
11 11	

		administrator to read and
		confirm the changes that
		are to be made.
	7. The administrator confirms the updates made to the schedule record. [ALT]	8. The system saves the relevant changes to the Schedule table. - Venue_ID (which is retrieved from the Venue_Name; selected from the dropdown list) - Booking_Type_ID (which is retrieved from the Booking_Type_Nam e; selected from the dropdown list) - Lesson_ID (which is retrieved from the dropdown list) - Lesson_ID (which is retrieved from the Lesson_Name; selected from the dropdown list) - Date_Session_ID(which is retrieved from the dropdown list) - Date_Session_ID(which is retrieved from the Date_Session table using the Date_Session_Start_Time and Date_Session_End_Time; selected from the dropdown list) - Employee_ID (which is retrieved from the Employee_TD (which is retrieved from the Employee_TD (which is retrieved from the Employee_Name; selected from the dropdown list)
		The system notifies the
		administrator that the
		changes have been successfully made.
Alternate	[ALT] Step 5: The required fields a	re not entered or the required fields
Courses:	are not in the correct format. The system displays an error notification informing the administrator. Return to step 4. [ALT] Step 7: The administrator denies the updates to be made. The	
Conducion	use case terminates	overtom notifice the administrator that
Conclusion:		system notifies the administrator that
	the schedule changes have been s	aveu successiully to the system.





Iteration 6 2022/08/01

Post-condition:	The schedule details are successfully updated in the Schedule table in the system's database.
Business Rules:	 Only authorised users of the system will be able to update a schedule (an administrator).
Implementation Constraints and Specifications	None
Assumptions	None
Open Issues	None



Table 2.1.7.4 Delete Schedule

Table 2.1.7.4 Delete Sched	uie	
Author(s): Shannon Noel		Creation Date:20/06/2022
Version: 1.0		Last Review Date:
Use Case Name:	Delete Schedule	Use case type
Use Case ID:	7.4	Business Requirements:
Priority:	HIGH	System Analysis:
Source:	Bester Strength and Conditioning	System Design:
Primary Business Actor:	Administrator	
Primary System Actor:	None	
Other Participating Actors:	None	
Other Interested Stakeholders:	None	
Description:	delete a specific schedule record. administrator requesting to delete to concludes when the schedule reco	the schedule record. The use case and has been deleted from the system.
Pre-condition:	The administrator must be The ask adviser accord must be	
Triggory	The schedule record must express to delege	7
Trigger:		te a schedule record on the system.
Typical Course	1. The administrator requests to delete a schedule record on the system.	2. The system will confirm no booking attendances are associated with the selected Schedule_ID as a FK in the Booking_Attendance table. [ALT] 3. The system will retrieve and display the: - Venue_Name - Booking_Type_Nam e - Lesson_Name - Date_Session - Employee_Name from the Schedule table 4. The system prompts the administrator to confirm the deletion of the schedule record
	5. The administrator confirms the deletion of the schedule record. [ALT]	6. The system deletes the schedule record from the Schedule table with the following attribute: - Venue_ID - Booking_Type_ID - Lesson_ID - Date_Session_ID - Employee_ID 7. The system notifies the administrator that the



		_
		schedule record has been successfully deleted.
Alternate	[ALT] Step 2: The system determines there are booking attendance	
Courses:	record/s associated with the selected schedule record. The system will	
	notify the administrator, and ask the records are associated before dele	em to ensure no booking attendance
	records are associated before dele	tion. The use case terminates.
	[ALT] Step 5: The administrator denies the deletion of the schedule	
	record. The use case terminates.	
Conclusion:	The use case concludes when the the schedule record has been succ	system notifies the administrator that essfully deleted from the system.
Post-condition:	The schedule record is deleted fror database	m the Schedule table in the system's
Business Rules:	 Only authorised users of the schedule record (an administration) 	e system will be able to delete a strator).
Implementation Constraints and	· ·	,
Specifications	None	
Assumptions	None	
Open Issues	None	



Table 2.1.7.5 View Schedule

Author(s): Shannon Noel		Creation Date:20/06/2022
Version: 1.0		Last Review Date:
Use Case Name:	View Schedule	
Use Case ID:		Use case type Business Requirements:
Ose Case ID.	7.5	Business requirements.
Priority:	HIGH	System Analysis:
Source:	Bester Strength and Conditioning	System Design:
Primary Business Actor:	Administrator	
Primary System Actor:	None	
Other Participating Actors:	None	
Other Interested Stakeholders:	None	
Description:	This use case describes the event where the administrator wants to access the schedule page. The system will display the page; allowing the administrator to access the create, search, edit and delete for schedule events.	
Pre-condition:	The administrator must be logged i	
Trigger:	The administrator requests to view	
Typical Course	Actor Action:	system Response:
	The administrator requests to view the schedule page.	The system will display the populated calendar on the screen with the:
	The administrator requests to search a schedule record.	The system invokes Use Case <u>7.2 "Search</u> <u>Schedule".</u>
Alternate Courses:	The system invokes Use Case 7.1 [ALT] Step 3b: The administrator r The system invokes Use Case 7.3 [ALT] Step 3c: The administrator r The system invokes Use Case 7.4	requests to update schedule records. "Update Schedule". equests to delete schedule records. "Delete Schedule".
Conclusion:	The system displays a calendar wit	
Post-condition:	The administrator will be able to se	
Business Rules:	 Only authorised users of the system can view the schedule page (an administrator) 	
Implementation Constraints and Specifications	None	
Assumptions	None	
Open Issues	None	



Table 2.1.7.6 Create Booking Type

Table 2.1.7.6 Create Bookii	ig Type	
Author(s): Luke Partridge		Creation Date:20/06/2022
Version: 1.0		Last Review Date:
Use Case Name:	Create Booking Type	Use case type
Use Case ID:	7.6	Business Requirements:
Priority:	HIGH	System Analysis:
Source:	Bester Strength and Conditioning	System Design:
Primary Business Actor:	Administrator	
Primary System Actor:	None	
Other Participating Actors:	None	
Other Interested Stakeholders:	None	
Description:	needed information required by the information and then stores it. The booking type record has successful	The use case begins when the w booking type record, and enters the e system. The system verifies the use case concludes when the ully been created on the system.
Pre-condition:	The administrator must be	
Trigger:	·	te a new booking type on the system
Typical Course	Actor Action:	system Response:
	9. The administrator requests to create a new booking type record on the system	10. The system prompts the administrator to enter the following details concerning the booking type from the Booking_type table - Booking_Type_Nam e - Booking_Type_Desc
	11. The administrator enters the required information into the specified fields	12. The system verifies that the required data fields are not left empty [ALT]
		13. The system will display the entered information and requests the administrator to read and confirm the entered information
	14. The administrator confirms the information entered [ALT]	15. The system verifies that the booking type record is not being duplicated in the Booking_Type table by checking if the entered attributes: - Booking_Type_Nam e - Booking_Type_Desc ription Matches any existing booking type record. [ALT]



Strengthening	Solutio
Team 7	

		16. The system will add the new booking type record to the Booking_Type table with the following attributes: - Booking_Type_ID (the value of which is the previous Booking_Type_ID incremented by 1) - Booking_Type_Nam e - Booking_Type_Desc ription 17. The System displays a success notification for the administrator that the booking type has been created
Alternate Courses:	[ALT] Step 4:All input fields are not in the correct format. The system di	filled in or the required fields are not
	with the attributes that are empty o [ALT] Step 6: The administrator de	r invalid. Return to step 3.
	correct. Return to step 3	
	[ALT] Step 7: The booking type alro system displays an error message	
	to step 5.	
Conclusion:	database and the system indicates	booking type details are added to the with a message that the details were
Doct condition:	added successfully.	oon added to the greaters in the
Post-condition:	The new booking type record has b Booking_Type table	een added to the system in the
Business Rules:	 Only administrator users are 	e able to create new booking types
Implementation Constraints and Specifications	None	
Assumptions	None	
Open Issues	None	



Table 2.1.7.7 Search Booking Type

Table 2.1.7.7 Search Bookin	g Type	
Author(s): Luke Partridge		Creation Date:20/06/2022
Version: 1.0		Last Review Date:
Use Case Name:	Search Booking Type	Use case type
Use Case ID:	7.7	Business Requirements:
Priority:	HIGH	System Analysis:
Source:	Bester Strength and Conditioning	System Design: □
Primary Business Actor:	Administrator	
Primary System Actor:	None	
Other Participating Actors:	None	
Other Interested Stakeholders:	None	
Pre-condition: Trigger:	a booking type record on the syste administrator requests to search fo case concludes when the informati booking type record. • The administrator must be I The administrator requests to search.	or a booking type record. The use on is displayed for that specific ogged into the system.
	system.	
Typical Course	Actor Action:	system Response:
	The administrator requests to search a booking type record on the system	The system prompts the administrator to enter the following details concerning the booking type record from the Booking_Type table: Booking_Type_Nam e Booking_Type_Desc ription
	The administrator enters the required information as search criteria.	4. The system retrieves and displays a list of all the booking type records which matches the search criteria entered: - Booking_Type_Nam e - Booking_Type_Desc ription [ALT]
Alternate	5. The administrator selects the specific booking type record they wish to view from the displayed results	6. The system retrieves and displays the following information regarding the booking type record from the Booking_Type table: - Booking_Type_Nam e - Booking_Type_Desc ription
Allemale	[ALT] Step 4: No results were four	iu. The use case terminates





Courses:	
Conclusion:	The use case concludes with the searched booking type record being displayed successfully.
Post-condition:	The administrator can view the searched booking type record successfully.
Business Rules:	 Only authorised users of the system will be able to search a booking type record (an administrator). The administrator must be registered on the system and logged into the system to be able to search for a booking type record.
Implementation Constraints and Specifications	None
Assumptions	None
Open Issues	None



Table 2.1.7.8 Update Booking Type

Table 2.1.7.8 Update Bookin	lg Type	
Author(s): Luke Partridge		Creation Date:20/06/2022
Version: 1.0		Last Review Date:
Use Case Name:	Update Booking Type	Use case type
Use Case ID:	7.8	Business Requirements:
Priority:	HIGH	System Analysis:
Source:	Bester Strength and Conditioning	System Design:
Primary Business Actor:	Administrator	
Primary System Actor:	None	
Other Participating Actors:	None	
Other Interested Stakeholders:	None	
Description:	update a specific booking type. The administrator requesting to update case concludes when the booking details are saved on the system.	the booking type record. The use type has been updated and the new
Pre-condition:	The administrator must be IThe booking type must exis	t on the system.
Trigger:	The administrator requests to upda system.	ite a booking type record on the
Typical Course	Actor Action:	system Response:
	The administrator requests to update a specific booking type record on the system.	The system reads and displays the following information regarding the booking type record from the Booking_Type table: Booking_Type_Nam e Booking_Type_Desc ription
		3. The system prompts the administrator to edit the pre-populated fields. The fields are populated with the information that was retrieved from the Booking_Type table.
	4. The administrator will update the relevant information of the booking type record and select to save the changes made.	5. The system validates that all the input fields are not left empty.[ALT]
		6. The system displays the new entered information and requests the administrator to read and confirm the changes that are to be made.



	-	
	7. The administrator confirms the updates made to the booking type record. [ALT]	8. The system verifies that the booking type record is not being duplicated in the Booking_Type table by checking if the entered attributes: - Booking_Type_Nam e - Booking_Type_Desc ription Matches any existing booking type record. [ALT]
		9. The system will update the booking type record to the Booking_Type table with the following attributes. - Booking_Type_ID
		(The Booking_Type_ID remain the same) - Booking_Type_Nam e
		- Booking_Type_Desc ription
		9. The system displays a success notification for the successfully updated booking type record.
Alternate Courses:	[ALT] Step 5: All input fields are not not in the correct format. The system the attributes that are empty or inv [ALT] Step 7: The administrator decrease terminates [ALT] Step 8: The booking type recommendates the system displays an error message Return to step 6.	of filled in or the required fields are in displays error messages regarding alid.Return to step 4. The enies the updates to be made. The cord already exists on the system,
Conclusion:		system notifies the administrator that en saved successfully to the system.
Post-condition:		ssfully updated in the Booking_Type
Business Rules:	booking type (an administra - A booking type cannot be up the system. - The administrator must be r	e system will be able to update a tor). pdated if it does not already exist on registered on the system and logged o update a booking type record.
Implementation Constraints and Specifications	None	
Assumptions	None	
Open Issues	None	





Table 2.1.7.9 Delete Booking Type

Table 2.1.7.9 Delete Bookin	у туре	
Author(s): Luke Partridge		Creation Date:20/06/2022
Version: 1.0		Last Review Date:
Use Case Name:	Delete Booking Type	Use case type
Use Case ID:	7.9	Business Requirements:
Priority:	HIGH	System Analysis:
Source:	Bester Strength and Conditioning	System Design: □
Primary Business Actor:	Administrator	
Primary System Actor:	None	
Other Participating Actors:	None	
Other Interested Stakeholders:	None	
Description:	delete a specific booking type reco administrator requesting to delete t case concludes when the booking the system.	the booking type record. The use type record has been deleted from
Pre-condition:	 The administrator must be I The booking type record must 	
Trigger:	The administrator requests to delet system.	te a booking type record on the
Typical Course	Actor Action:	system Response:
	The administrator requests to delete a booking type record on the system.	The system will confirm no schedule records are associated with the selected Booking_Type_ID as a FK in the Schedule table. The system will do this by performing an SQL Read query in the .Net Core controller [ALT]
		3. The system reads and displays the following attributes for the selected booking type record from the Booking_Type table: - Booking_Type_Nam e - Booking_Type_Desc ription 3. The system prompts the
	3. The administrator	administrator to confirm the deletion of the booking type record 3. The system deletes the
	confirms the deletion of the booking type record. [ALT]	booking type record from the Booking_Type table with the following attributes



	- Booking_Type_Nam e - Booking_Type_Desc ription	
	3. The system displays a success notification for the successfully deleted booking type record.	
Alternate Courses:	[ALT] Step 2: The system determines there are schedule record/s associated with the selected booking type record. The system will notify the administrator, and ask them to ensure no schedule records are. The use case terminates. [ALT] Step 5: The administrator denies the deletion of the sale category record. The use case terminates.	
Conclusion:	The use case concludes when the system notifies the administrator that the booking type record has been successfully deleted from the system.	
Post-condition:	The booking type record is deleted from the Booking_Type table in the system's database.	
Business Rules:	 Only authorised users of the system will be able to delete a booking type record (an administrator). A booking type cannot be deleted if it does not already exist on the system. The administrator must be registered on the system and logged into the system to be able to delete a booking type record. 	
Implementation Constraints and Specifications	None	
Assumptions	None	
On an Inc	Mana	

64

None



Open Issues

Table 2.1.7.10 View Booking Type

Table 2.1.7.10 View Booking	g Type	0 (; D (00/00/0000
Author(s): Luke Partridge		Creation Date:20/06/2022
Version: 1.0	1.0 B 1: -	Last Review Date:
Use Case Name:	View Booking Type	Use case type
Use Case ID:	7.10	Business Requirements:
Priority:	HIGH	System Analysis:
Source:	Bester Strength and Conditioning	System Design: □
Primary Business Actor:	Administrator	
Primary System Actor:	None	
Other Participating Actors:	None	
Other Interested Stakeholders:	None	
Description:	for booking type	e system will display the page; s the create, search, edit and delete
Pre-condition:	The administrator must be logged in	
Trigger:	The administrator requests to view	· · · · · ·
Typical Course	Actor Action:	system Response:
	The administrator requests to view the booking type page.	2. The system will display the populated list on the screen with the: - Booking_Type_Nam e - Booking_Type_Desc ription from the retrieved Booking_Type records.
	The administrator requests to search a booking type record. [ALT]	The system invokes Use Case <u>7.7 "Search Booking</u> Type".
Alternate Courses:	[ALT] Step 3a: The administrator record. The system invokes Use Case 7.6 [ALT] Step 3b: The administrator record. The system invokes Use Case 7.8 [ALT] Step 3c: The administrator record. The system invokes Use Case 7.9	"Create Booking Type". requests to update a booking type "Update Booking Type". requests to delete a booking type "Delete Booking Type".
Conclusion:	The system displays a list with all the booking type records and their information.	
Post-condition:	The administrator will be able to se	
Business Rules:	Only authorised users of the page (an administrator)	e system can view the booking type
Implementation Constraints and Specifications	None	
Assumptions	None	
Open Issues	None	
	110110	





2.1.8 Lesson Plan Subsystem

Table 2.1.8.6 Create Exercise

Author(s): Lyne' Keet Version: 1.0		Creation Date:20/06/2022 Last Review Date:
Use Case Name:	Create Exercise	Use case type
Use Case ID:	8.6	Business Requirements:
Priority:	HIGH	System Analysis:
Source:	Bester Strength and Conditioning	System Design: □
Primary Business Actor:	Trainer	
Primary System Actor:	None	
Other Participating Actors:	None	
Other Interested Stakeholders:	None	
Description:	This use case describes the event where the trainer creates a new exercise on the system. The use case begins when the trainer wants to create a exercise record, and enters the needed information required by the system. The system verifies the information and then stores it. The use case concludes when the exercise record has successfully been created on the system.	
Pre-condition:	The trainer must be logged	into the system
Trigger:	The administrator requests to creat	
Typical Course	Actor Action:	system Response:
	The administrator requests to create a exercise record on the system	The system prompts the trainer to enter the following details concerning the exercise from the Exercise table Exercise_Name Exercise_Focus Exercise_URL
		3. The system will retrieve the list of exercise categories with their: - Exercise_Category_Na me From the Exercise_Category table and populate the form. The system will prompt the trainer to select which exercise category is applicable for this exercise record.
	4. The trainer enters the required information in respective fields, and selects a exercise category	5. The system validates that all the input fields are not left empty. The system validates the following fields to ensure that valid inputs are entered The Exercise Name cannot be left empty The Exercise Url cannot be left empty



Strengthening	Solutions
Team 7	

acannot be left empty - Validate url is a valid youtube link - The Exercise category is selected [ALT] - 6. The system displayed the entered information and requests the trainer to read and confirm the entered information and requests the trainer to read and confirm the entered information. 8. The system verifies that the exercise record is not being duplicated in the Exercise table by checking if the entered attributes: - Exercise_Name Matches any existing Exercise Record. [ALT] - 9. The system will add the new exercise record to the Exerciselable with the following attributes Exercise_ID incremented by 1) - Exercise, Name - Exercise, Name - Exercise, Focus - Exercise, Focus - Exercise Category ID - 10. The system displays a success message to the trainer that the exercise record has been created. Alternate [ALT] Step 5: All input fields are not filled in or the required fields are not in the correct format. The system displays error messages under the input fields that are empty or invalid. Return to step 4 [ALT] Step 7: The administrator denies the information entered is correct. Return to step 4 [ALT] Step 8: The exercise already exists on the system, the system displays an error notification informing the administrator. Return to step 6. Conclusion: The use case concludes when the exercise details are added to the database and the system indicates with a message that the details were added successfully. Post-condition: A new exercise is created on the system in the Exercise table Only authorised users of the system will be able to create an employee (an administrator, trainer and Super User).		_	The East Control
Alternate Courses: Conclusion: Alternate Courses: Conclusion: Constraints and Specifications Conclusion: Concl			- The Exercise Focus
Secretary			
- The Exercise Category is selected [ALT] 6. The system displayed the entered information and requests the trainer to read and confirm the entered information. 8. The system verifies that the exercise record is not being duplicated in the Exercise table by checking if the entered attributes: 7. The trainer confirms the information entered [ALT] 9. The system will add the new exercise record to the Exercise table with the following attributes: - Exercise Poot to the Exercise attributes: - Exercise ID (the value of which is the previous Exercise ID) incremented by 1) - Exercise JO (the value of which is the previous Exercise To (the Exercise JO (the value of which is the previous Exercise ID (the value of which is the previous Exercise ID (the value of which is the previous Exercise ID (the value of which is the previous Exercise ID (the value of which is the previous Exercise ID (the value of which is the previous Exercise ID (the value of which is the previous Exercise ID (the value of which is the previous Exercise ID (the value of which is the previous Exercise			
Selected [ALT]			
Alternate Courses: G. The system displayed the entered information and requests the trainer to read and confirm the entered information and requests the trainer to read and confirm the entered information in the exercise record is not being duplicated in the Exercise table by checking if the entered attributes: F. Exercise Name Matches any existing Exercise Record. [ALT] 9. The system will add the new exercise record to the Exercise labele with the following attributes. Exercise ID (the value of which is the previous Exercise ID incremented by 1) Exercise Jame Exercise JuRL Exercise JuRL Exercise JuRL Exercise JuRL Exercise Category ID 10. The system displays a success message to the trainer that the exercise record has been created. Alternate [ALT] Step 5: All input fields are not filled in or the required fields are not in the correct format. The system displays error messages under the input fields that are empty or invalid. Return to step 4 [ALT] Step 7: The administrator denies the information entered is correct. Return to step 4 [ALT] Step 8: The exercise already exists on the system, the system displays are ror notification informing the administrator. Return to step 6 Conclusion: The use case concludes when the exercise details are added to the database and the system indicates with a message that the details were added successfully. A new exercise is created on the system in the Exercise table. Only authorised users of the system in the Exercise table.			1
entered information and requests the trainer to read and confirm the entered information. 7. The trainer confirms the information entered[ALT] 7. The trainer confirms the information entered[ALT] 7. The trainer confirms the information entered[ALT] 9. The system wrifing Exercise Record. [ALT] 9. The system will add the new exercise record to the Exerciseable with the following attributes. - Exercise_ID (the value of which is the previous Exercise_ID) incremented by 1) - Exercise_ID (the value of which is the previous Exercise_ID) - Exercise_ID (The value of which is the previous Exercise_ID) - Exercise_ID (The value of which is the previous Exercise_ID) - Exercise_ID (The value of which is the previous Exercise_ID) - Exercise_ID (The value of which is the previous Exercise_ID) - Exercise_ID (The value of which is the previous Exercise_ID) - Exercise_ID (The value of which is the previous Exercise_ID) - Exercise_ID (The value of which is the previous Exercise_ID) - Exercise_ID (The value of which is the previous Exercise_ID) - Exercise_ID (The value of which is the previous Exercise ecord has been created. - Exercise_ID (The value of which is the previous Exercise record has been created. - Exercise_ID (The value of which is the previous Exercise record has been created. - Exercise_ID (The value of which is the previous Exercise the information entered is correct. Return to step 4 - Exercise_ID (The value of which is the previous Exercise the information entered is correct. Return to step 4 - Exercise_ID (The value of which is the previous Exercise the information entered is correct. Return to step 4 - Exercise_ID (The value of which is the previous Exercise the information entered is correct. Return to step 4 - Exercise_ID (The value of which is the previous Exercise the information entered is correct. Return to step 4 - Exercise_ID (The value of which is the previous Exercise the information entered is correct. Return to step 4 - Exercise_ID (The value of which is the previous Exercise			is selected [ALT]
7. The trainer confirms the information entered [ALT] 7. The trainer confirms the information entered [ALT] 7. The trainer confirms the information entered [ALT] 9. The system will add the new exercise record to the Exercise Record. [ALT] 9. The system will add the new exercise record to the Exercise attributes: - Exercise D(the value of which is the previous Exercise_ID incremented by 1) - Exercise_ID (the value of which is the previous Exercise_ID incremented by 1) - Exercise_ID (the value of which is the previous Exercise_ID incremented by 1) - Exercise_ID (the value of which is the previous Exercise_ID incremented by 1) - Exercise_ID (the value of which is the previous Exercise_ID incremented by 1) - Exercise_ID (the value of which is the previous Exercise_ID incremented by 1) - Exercise_ID (the value of which is the previous Exercise_ID incremented by 1) - Exercise_INE - Exercise_ID (the value of which is the previous Exercise_ID incremented by 1) - Exercise_ID (the value of which is the previous Exercise_ID incremented by 1) - Exercise_ID (the value of which is the previous Exercise_ID incremented by 1) - Exercise_ID (the value of which is the previous Exercise the trainer that the exercise details are not filled in or the required fields are not in the correct format. The system displays error messages under the input fields that are empty or invalid. Return to step 4 [ALT] Step 5: All input fields are not filled in or the required fields are not in the correct of the system will be able to create an employee (an administrator, trainer and Super User). Post-condition: A new exercise is created on the system will be able to create an employee (an administra			
7. The trainer confirms the information entered[ALT] 8. The system verifies that the exercise record is not being duplicated in the Exercise table by checking if the entered attributes: - Exercise_Name Matches any existing Exercise Record. [ALT] 9. The system will add the new exercise record to the Exercise able with the following attributes. - Exercise_ID (the value of which is the previous Exercise_ID incremented by 1) - Exercise_Name - Exercise_Dame - Exercise_Dame - Exercise_Dame - Exercise_Dame - Exercise_Category ID 10. The system displays a success message to the trainer that the exercise record has been created. Alternate Courses: [ALT] Step 5: All input fields are not filled in or the required fields are not in the correct format. The system displays error messages under the input fields that are empty or invalid. Return to step 4 [ALT] Step 7: The administrator denies the information entered is correct. Return to step 4 [ALT] Step 7: The administrator denies the information entered is correct. Return to step 4 [ALT] Step 8: The exercise already exists on the system, the system displays an error notification informing the administrator. Return to step 6. Conclusion: Conclusion: A new exercise is created on the system will be able to create an employee (an administrator, trainer and Super User). None			entered information and
7. The trainer confirms the information entered[ALT] 8. The system verifies that the exercise rable by checking if the entered attributes: - Exercise_Name Matches any existing Exercise Record. [ALT] 9. The system will add the new exercise record to the Exercise table with the following attributes. - Exercise_ID (the value of which is the previous Exercise_ID incremented by 1) - Exercise_Name - Exercise_Varl. - Exercise_Varl. - Exercise Category ID 10. The system displays a success message to the trainer that the exercise record has been created. Alternate Courses: [ALT] Step 5: All input fields are not filled in or the required fields are not in the correct format. The system displays error messages under the input fields that are empty or invalid. Return to step 4 [ALT] Step 8: The administrator denies the information entered is correct. Return to step 4 [ALT] Step 8: The exercise already exists on the system, the system displays an error notification informing the administrator. Return to step 6. Conclusion: The use case concludes when the exercise details are added to the database and the system indicates with a message that the details were added successfully. Post-condition: A new exercise is created on the system in the Exercise table. - Only authorised users of the system in the Exercise table. None			requests the trainer to read and
7. The trainer confirms the information entered[ALT] 8. Exercise_Name Matches any existing Exercise Record. [ALT] 9. The system will add the new exercise record to the Exercise table with the following attributes. - Exercise_ID (the value of which is the previous Exercise_ID incremented by 1) - Exercise_Name - Exercise_Name - Exercise_Secrose_ID incremented by 1) - Exercise_LRL - Exercise_Category ID 10. The system displays a success message to the trainer that the exercise record has been created. Alternate Courses: [ALT] Step 5: All input fields are not filled in or the required fields are not in the correct format. The system displays error messages under the input fields that are empty or invalid. Return to step 4 [ALT] Step 7: The administrator denies the information entered is correct. Return to step 4 [ALT] Step 8: The exercise already exists on the system, the system displays an error notification informing the administrator. Return to step 6. Conclusion: The use case concludes when the exercise details are added to the database and the system indicates with a message that the details were added successfully. Post-condition: A new exercise is created on the system in the Exercise table. - Only authorised users of the system will be able to create an employee (an administrator, trainer and Super User).			confirm the entered information
7. The trainer confirms the information entered [ALT] 7. The trainer confirms the information entered [ALT] 7. The trainer confirms the information entered [ALT] 8. The system will add the new exercise record to the Exercise allo with the following attributes. 9. The system will add the new exercise record to the Exercise allo (the value of which is the previous Exercise. ID (the value of which is the previous Exercise. ID incremented by 1) 9. Exercise. ID (the value of which is the previous Exercise. ID incremented by 1) 10. Exercise Category ID 11. The system displays a success message to the trainer that the exercise record has been created. Alternate Courses: [ALT] Step 5: All input fields are not filled in or the required fields are not in the correct format. The system displays error messages under the input fields that are empty or invalid. Return to step 4 [ALT] Step 8: The administrator denies the information entered is correct. Return to step 4 [ALT] Step 8: The exercise already exists on the system, the system displays an error notification informing the administrator. Return to step 6. Conclusion: The use case concludes when the exercise details are added to the database and the system indicates with a message that the details were added successfully. Post-condition: A new exercise is created on the system in the Exercise table. Only authorised users of the system will be able to create an employee (an administrator, trainer and Super User).			8. The system verifies that the
7. The trainer confirms the information entered[ALT] 7. The trainer confirms the information entered[ALT] 8. Exercise_Name Matches any existing Exercise Record. [ALT] 9. The system will add the new exercise record to the Exercisetable with the following attributes. - Exercise_ID (the value of which is the previous Exercise_ID incremented by 1) - Exercise_Name - Exercise_Focus - Exercise_URL - Exercise_URL - Exercise_Category ID 10. The system displays a success message to the trainer that the exercise record has been created. Alternate Courses: [ALT] Step 5: All input fields are not filled in or the required fields are not in the correct format. The system displays ser or messages under the input fields that are empty or invalid. Return to step 4 [ALT] Step 7: The administrator denies the information entered is correct. Return to step 4 [ALT] Step 8: The exercise already exists on the system, the system displays an error notification informing the administrator. Return to step 6. Conclusion: The use case concludes when the exercise details are added to the database and the system indicates with a message that the details were added successfully. Post-condition: A new exercise is created on the system in the Exercise table. Uniplementation Constraints and Specifications None			exercise record is not being
information entered[ALT] attributes:			duplicated in the Exercise
information entered[ALT] attributes: - Exercise_Name Matches any existing Exercise Record. [ALT] 9. The system will add the new exercise record to the Exercisetable with the following attributes. - Exercise_ID (the value of which is the previous Exercise_ID incremented by 1) - Exercise_Name - Exercise_Value - Exercise_URL - Exercise_URL - Exercise_URL - Exercise_URL - Exercise category ID 10. The system displays a success message to the trainer that the exercise record has been created. Alternate Courses: [ALT] Step 5: All input fields are not filled in or the required fields are not in the correct format. The system displays error messages under the input fields that are empty or invalid. Return to step 4 [ALT] Step 7: The administrator denies the information entered is correct. Return to step 4 [ALT] Step 8: The exercise already exists on the system, the system displays an error notification informing the administrator. Return to step 6. Conclusion: The use case concludes when the exercise details are added to the database and the system indicates with a message that the details were added successfully. Post-condition: A new exercise is created on the system in the Exercise table. Only authorised users of the system will be able to create an employee (an administrator, trainer and Super User).		7. The trainer confirms the	table by checking if the entered
Alternate Courses: [ALT] Step 5: All input fields are not filled in or the required fields sare not in the correct format. The system displays an error notification informing the administrator. Return to step 4 [ALT] Step 7: The administrator, trainer and Super User). Conclusion: A new exercise lable. A new exercise second to the Exercise LD (the value of which is the previous Exercise ID incremented by 1) Exercise LName Exercise_IRL Exercise_URL Exercise_URL Exercise Category ID 10. The system displays a success message to the trainer that the exercise record has been created. [ALT] Step 5: All input fields are not filled in or the required fields are not in the correct format. The system displays error messages under the input fields that are empty or invalid. Return to step 4 [ALT] Step 7: The administrator denies the information entered is correct. Return to step 4 [ALT] Step 8: The exercise already exists on the system, the system displays an error notification informing the administrator. Return to step 6. Conclusion: A new exercise is created on the system in the Exercise table. - Only authorised users of the system will be able to create an employee (an administrator, trainer and Super User).		information entered[ALT]	
Matches any existing Exercise Record. [ALT] 9. The system will add the new exercise record to the Exerciseable with the following attributes. - Exercise_ID (the value of which is the previous Exercise_ID incremented by 1) - Exercise_Name - Exercise_VRL - Exercise_URL - Exercise_URL - Exercise_URL - Exercise Category_ID 10. The system displays a success message to the trainer that the exercise record has been created. Alternate Courses: [ALT] Step 5: All input fields are not filled in or the required fields are not in the correct format. The system displays error messages under the input fields that are empty or invalid. Return to step 4 [ALT] Step 7: The administrator denies the information entered is correct. Return to step 4 [ALT] Step 8: The exercise already exists on the system, the system displays an error notification informing the administrator. Return to step 6. Conclusion: The use case concludes when the exercise details are added to the database and the system indicates with a message that the details were added successfully. Post-condition: A new exercise is created on the system in the Exercise table. Business Rules: - Only authorised users of the system will be able to create an employee (an administrator, trainer and Super User). Implementation Constraints and Specifications			- Exercise Name
Record. [ALT] 9. The system will add the new exercise record to the Exerciseable with the following attributes Exercise_ID (the value of which is the previous Exercise_ID incremented by 1) - Exercise_Se_Name - Exercise_Focus - Exercise_Category_ID 10. The system displays a success message to the trainer that the exercise record has been created. Alternate Courses: [ALT] Step 5: All input fields are not filled in or the required fields are not in the correct format. The system displays error messages under the input fields that are empty or invalid. Return to step 4 [ALT] Step 7: The administrator denies the information entered is correct. Return to step 4 [ALT] Step 8: The exercise already exists on the system, the system displays an error notification informing the administrator. Return to step 6. Conclusion: The use case concludes when the exercise details are added to the database and the system indicates with a message that the details were added successfully. Post-condition: A new exercise is created on the system in the Exercise table. Unplementation Constraints and Specifications None			_
9. The system will add the new exercise record to the Exercise able with the following attributes. - Exercise_ID (the value of which is the previous Exercise_ID incremented by 1) - Exercise_Name - Exercise_Vame - Exercise_Vame - Exercise_URL - Exercise Category ID 10. The system displays a success message to the trainer that the exercise record has been created. Alternate [ALT] Step 5: All input fields are not filled in or the required fields are not in the correct format. The system displays error messages under the input fields that are empty or invalid. Return to step 4 [ALT] Step 7: The administrator denies the information entered is correct. Return to step 4 [ALT] Step 8: The exercise already exists on the system, the system displays an error notification informing the administrator. Return to step 6. Conclusion: The use case concludes when the exercise details are added to the dabase and the system indicates with a message that the details were added successfully. Post-condition: A new exercise is created on the system in the Exercise table. Business Rules: - Only authorised users of the system will be able to create an employee (an administrator, trainer and Super User). None			
exercise record to the Exercisetable with the following attributes. - Exercise_ID (the value of which is the previous Exercise_ID incremented by 1) - Exercise_Name - Exercise_Focus - Exercise_URL - Exercise_Oategory_ID 10. The system displays a success message to the trainer that the exercise record has been created. Alternate Courses: Altimate Internate Internate			<u> </u>
Exercise table with the following attributes.			1
following attributes. - Exercise_ID (the value of which is the previous Exercise_ID incremented by 1) - Exercise_Name - Exercise_Vame - Exercise_Vame - Exercise_URL - Exercise Category ID 10. The system displays a success message to the trainer that the exercise record has been created. Alternate Courses: [ALT] Step 5: All input fields are not filled in or the required fields are not in the correct format. The system displays error messages under the input fields that are empty or invalid. Return to step 4 [ALT] Step 7: The administrator denies the information entered is correct. Return to step 4 [ALT] Step 8: The exercise already exists on the system, the system displays an error notification informing the administrator. Return to step 6. Conclusion: The use case concludes when the exercise details are added to the database and the system indicates with a message that the details were added successfully. Post-condition: A new exercise is created on the system in the Exercise table. Business Rules: - Only authorised users of the system will be able to create an employee (an administrator, trainer and Super User).			
- Exercise_ID (the value of which is the previous Exercise_ID incremented by 1) - Exercise_Name - Exercise_Name - Exercise_URL - Exercise Category ID 10. The system displays a success message to the trainer that the exercise record has been created. Alternate Courses: [ALT] Step 5: All input fields are not filled in or the required fields are not in the correct format. The system displays error messages under the input fields that are empty or invalid. Return to step 4 [ALT] Step 7: The administrator denies the information entered is correct. Return to step 4 [ALT] Step 8: The exercise already exists on the system, the system displays an error notification informing the administrator. Return to step 6. Conclusion: The use case concludes when the exercise details are added to the database and the system indicates with a message that the details were added successfully. Post-condition: A new exercise is created on the system in the Exercise table. - Only authorised users of the system will be able to create an employee (an administrator, trainer and Super User). None			
of which is the previous Exercise_ID incremented by 1) - Exercise_Name - Exercise_Focus - Exercise_URL - Exercise Category ID 10. The system displays a success message to the trainer that the exercise record has been created. Alternate Courses: [ALT] Step 5: All input fields are not filled in or the required fields are not in the correct format. The system displays error messages under the input fields that are empty or invalid. Return to step 4 [ALT] Step 7: The administrator denies the information entered is correct. Return to step 4 [ALT] Step 8: The exercise already exists on the system, the system displays an error notification informing the administrator. Return to step 6. Conclusion: The use case concludes when the exercise details are added to the database and the system indicates with a message that the details were added successfully. Post-condition: A new exercise is created on the system will be able to create an employee (an administrator, trainer and Super User). Implementation Constraints and Specifications			_
Exercise_ID incremented by 1) - Exercise_Name - Exercise_Focus - Exercise_URL - Exercise Category ID 10. The system displays a success message to the trainer that the exercise record has been created. Alternate Courses: [ALT] Step 5: All input fields are not filled in or the required fields are not in the correct format. The system displays error messages under the input fields that are empty or invalid. Return to step 4 [ALT] Step 7: The administrator denies the information entered is correct. Return to step 4 [ALT] Step 8: The exercise already exists on the system, the system displays an error notification informing the administrator. Return to step 6. Conclusion: The use case concludes when the exercise details are added to the database and the system indicates with a message that the details were added successfully. Post-condition: A new exercise is created on the system will be able to create an employee (an administrator, trainer and Super User). Implementation Constraints and Specifications			
incremented by 1) - Exercise_Name - Exercise_Focus - Exercise_URL - Exercise Category ID 10. The system displays a success message to the trainer that the exercise record has been created. [ALT] Step 5: All input fields are not filled in or the required fields are not in the correct format. The system displays error messages under the input fields that are empty or invalid. Return to step 4 [ALT] Step 7: The administrator denies the information entered is correct. Return to step 4 [ALT] Step 8: The exercise already exists on the system, the system displays an error notification informing the administrator. Return to step 6. Conclusion: The use case concludes when the exercise details are added to the database and the system indicates with a message that the details were added successfully. Post-condition: A new exercise is created on the system in the Exercise table. - Only authorised users of the system will be able to create an employee (an administrator, trainer and Super User). Implementation Constraints and Specifications			
- Exercise_Name - Exercise_Focus - Exercise_URL - Exercise Category ID 10. The system displays a success message to the trainer that the exercise record has been created. [ALT] Step 5: All input fields are not filled in or the required fields are not in the correct format. The system displays error messages under the input fields that are empty or invalid. Return to step 4 [ALT] Step 7: The administrator denies the information entered is correct. Return to step 4 [ALT] Step 8: The exercise already exists on the system, the system displays an error notification informing the administrator. Return to step 6. Conclusion: The use case concludes when the exercise details are added to the database and the system indicates with a message that the details were added successfully. Post-condition: A new exercise is created on the system will be able to create an employee (an administrator, trainer and Super User). Implementation Constraints and Specifications			_
- Exercise_Focus - Exercise_URL - Exercise Category ID 10. The system displays a success message to the trainer that the exercise record has been created. Alternate Courses: [ALT] Step 5: All input fields are not filled in or the required fields are not in the correct format. The system displays error messages under the input fields that are empty or invalid. Return to step 4 [ALT] Step 7: The administrator denies the information entered is correct. Return to step 4 [ALT] Step 8: The exercise already exists on the system, the system displays an error notification informing the administrator. Return to step 6. Conclusion: The use case concludes when the exercise details are added to the database and the system indicates with a message that the details were added successfully. Post-condition: A new exercise is created on the system in the Exercise table. Only authorised users of the system will be able to create an employee (an administrator, trainer and Super User). Implementation Constraints and Specifications			
- Exercise_URL - Exercise Category ID 10. The system displays a success message to the trainer that the exercise record has been created. Alternate Courses: [ALT] Step 5: All input fields are not filled in or the required fields are not in the correct format. The system displays error messages under the input fields that are empty or invalid. Return to step 4 [ALT] Step 7: The administrator denies the information entered is correct. Return to step 4 [ALT] Step 8: The exercise already exists on the system, the system displays an error notification informing the administrator. Return to step 6. Conclusion: The use case concludes when the exercise details are added to the database and the system indicates with a message that the details were added successfully. Post-condition: A new exercise is created on the system in the Exercise table. - Only authorised users of the system will be able to create an employee (an administrator, trainer and Super User). Implementation Constraints and Specifications			
- Exercise Category ID 10. The system displays a success message to the trainer that the exercise record has been created. Alternate Courses: [ALT] Step 5: All input fields are not filled in or the required fields are not in the correct format. The system displays error messages under the input fields that are empty or invalid. Return to step 4 [ALT] Step 7: The administrator denies the information entered is correct. Return to step 4 [ALT] Step 8: The exercise already exists on the system, the system displays an error notification informing the administrator. Return to step 6. Conclusion: The use case concludes when the exercise details are added to the database and the system indicates with a message that the details were added successfully. Post-condition: A new exercise is created on the system in the Exercise table. - Only authorised users of the system will be able to create an employee (an administrator, trainer and Super User). Implementation Constraints and Specifications			_
Alternate Courses: [ALT] Step 5: All input fields are not filled in or the required fields are not in the correct format. The system displays error messages under the input fields that are empty or invalid. Return to step 4 [ALT] Step 7: The administrator denies the information entered is correct. Return to step 4 [ALT] Step 8: The exercise already exists on the system, the system displays an error notification informing the administrator. Return to step 6. Conclusion: The use case concludes when the exercise details are added to the database and the system indicates with a message that the details were added successfully. Post-condition: A new exercise is created on the system in the Exercise table. - Only authorised users of the system will be able to create an employee (an administrator, trainer and Super User). None			_
Alternate Courses: [ALT] Step 5: All input fields are not filled in or the required fields are not in the correct format. The system displays error messages under the input fields that are empty or invalid. Return to step 4 [ALT] Step 7: The administrator denies the information entered is correct. Return to step 4 [ALT] Step 8: The exercise already exists on the system, the system displays an error notification informing the administrator. Return to step 6. Conclusion: The use case concludes when the exercise details are added to the database and the system indicates with a message that the details were added successfully. Post-condition: A new exercise is created on the system in the Exercise table. - Only authorised users of the system will be able to create an employee (an administrator, trainer and Super User). None			
Alternate Courses: [ALT] Step 5: All input fields are not filled in or the required fields are not in the correct format. The system displays error messages under the input fields that are empty or invalid. Return to step 4 [ALT] Step 7: The administrator denies the information entered is correct. Return to step 4 [ALT] Step 8: The exercise already exists on the system, the system displays an error notification informing the administrator. Return to step 6. Conclusion: The use case concludes when the exercise details are added to the database and the system indicates with a message that the details were added successfully. Post-condition: A new exercise is created on the system in the Exercise table. Only authorised users of the system will be able to create an employee (an administrator, trainer and Super User). None			
Alternate Courses: Deen created.			
Alternate Courses: [ALT] Step 5: All input fields are not filled in or the required fields are not in the correct format. The system displays error messages under the input fields that are empty or invalid. Return to step 4 [ALT] Step 7: The administrator denies the information entered is correct. Return to step 4 [ALT] Step 8: The exercise already exists on the system, the system displays an error notification informing the administrator. Return to step 6. Conclusion: The use case concludes when the exercise details are added to the database and the system indicates with a message that the details were added successfully. Post-condition: A new exercise is created on the system in the Exercise table. Only authorised users of the system will be able to create an employee (an administrator, trainer and Super User). Implementation Constraints and Specifications			
the correct format. The system displays error messages under the input fields that are empty or invalid. Return to step 4 [ALT] Step 7: The administrator denies the information entered is correct. Return to step 4 [ALT] Step 8: The exercise already exists on the system, the system displays an error notification informing the administrator. Return to step 6. Conclusion: The use case concludes when the exercise details are added to the database and the system indicates with a message that the details were added successfully. Post-condition: A new exercise is created on the system in the Exercise table. Only authorised users of the system will be able to create an employee (an administrator, trainer and Super User). None None	Alternate	[ALT] Stop 5: All input fields are no	
fields that are empty or invalid. Return to step 4 [ALT] Step 7: The administrator denies the information entered is correct. Return to step 4 [ALT] Step 8: The exercise already exists on the system, the system displays an error notification informing the administrator. Return to step 6. Conclusion: The use case concludes when the exercise details are added to the database and the system indicates with a message that the details were added successfully. Post-condition: A new exercise is created on the system in the Exercise table. Only authorised users of the system will be able to create an employee (an administrator, trainer and Super User). Implementation Constraints and Specifications			
[ALT] Step 7: The administrator denies the information entered is correct. Return to step 4 [ALT] Step 8: The exercise already exists on the system, the system displays an error notification informing the administrator. Return to step 6. Conclusion: The use case concludes when the exercise details are added to the database and the system indicates with a message that the details were added successfully. Post-condition: A new exercise is created on the system in the Exercise table. Only authorised users of the system will be able to create an employee (an administrator, trainer and Super User). None None	Courses.		
Return to step 4 [ALT] Step 8: The exercise already exists on the system, the system displays an error notification informing the administrator. Return to step 6. Conclusion: The use case concludes when the exercise details are added to the database and the system indicates with a message that the details were added successfully. Post-condition: A new exercise is created on the system in the Exercise table. Only authorised users of the system will be able to create an employee (an administrator, trainer and Super User). None None			
[ALT] Step 8: The exercise already exists on the system, the system displays an error notification informing the administrator. Return to step 6. Conclusion: The use case concludes when the exercise details are added to the database and the system indicates with a message that the details were added successfully. Post-condition: A new exercise is created on the system in the Exercise table. - Only authorised users of the system will be able to create an employee (an administrator, trainer and Super User). Implementation Constraints and Specifications None		1	and the internation official to correct.
displays an error notification informing the administrator. Return to step 6. Conclusion: The use case concludes when the exercise details are added to the database and the system indicates with a message that the details were added successfully. Post-condition: A new exercise is created on the system in the Exercise table. Only authorised users of the system will be able to create an employee (an administrator, trainer and Super User). Implementation Constraints and Specifications None		· ·	v exists on the system the system
Conclusion: The use case concludes when the exercise details are added to the database and the system indicates with a message that the details were added successfully. Post-condition: A new exercise is created on the system in the Exercise table. Only authorised users of the system will be able to create an employee (an administrator, trainer and Super User). Implementation Constraints and Specifications None			,
database and the system indicates with a message that the details were added successfully. Post-condition: A new exercise is created on the system in the Exercise table. - Only authorised users of the system will be able to create an employee (an administrator, trainer and Super User). Implementation Constraints and Specifications None	Conclusion:		
added successfully. Post-condition: Business Rules: Only authorised users of the system will be able to create an employee (an administrator, trainer and Super User). Implementation Constraints and Specifications None			
Post-condition: Business Rules: Only authorised users of the system will be able to create an employee (an administrator, trainer and Super User). Implementation Constraints and Specifications None		•	
Business Rules: Only authorised users of the system will be able to create an employee (an administrator, trainer and Super User). Implementation Constraints and Specifications None	Post-condition:		ystem in the Exercise table.
employee (an administrator, trainer and Super User). Implementation Constraints and Specifications None			
Implementation Constraints and Specifications None		1	-
Specifications	Implementation Constraints and		/
	•	None	
1.000	Assumptions	None	
Open Issues None		None	





Table 2.1.8.7 Search Exercise

Table 2.1.8.7 Search Exerc	186	
Author(s): Lyne' Keet		Creation Date:20/06/2022
Version: 1.0		Last Review Date:
Use Case Name:	Search Exercise	Use case type
Use Case ID:	8.7	Business Requirements:
Priority:	HIGH	System Analysis:
Source:	Bester Strength and Conditioning	System Design:
Primary Business Actor:	Trainer	
Primary System Actor:	None	
Other Participating Actors:	None	
Other Interested Stakeholders:	None	
Description:	This use case describes the event exercise record on the system. The requests to search for an exercise when the information is displayed f	e use case begins when the trainer record. The use case concludes or that specific exercise.
Pre-condition:	The trainer must be logged	into the system
Trigger:	The administrator requests to sear	ch an exercise record on the system
Typical Course	Actor Action:	system Response:
	The trainer requests to search an exercise record on the system	The system prompts the trainer to enter a search criteria related to any of the attributed founded in the Exercise table
	The administrator enters the information as search criteria.	4. The system retrieves and displays a list of all the exercise records which matches the search criteria entered from the Exercise The system retrieves and displays the Exercise_Category_Name from the Exercise_Category table using the Exercise_Category_ID FK for each exercise record. [ALT]
	5. The trainer selects the specific exercise they wish to view from the displayed results.	6. The system retrieves and displays the following information regarding the exercise from the Exercise table: - Exercise_Name - Exercise_Focus - Exercise_URL from the Exercise_Category table: - Exercise_Category_Name

68



11 ~ 11 0	
۳اا ∩ اا	
Д	
\sim	
11 11	

Alternate Courses:	[ALT] Step 4: No results were found. The use case terminates
Conclusion:	The use case concludes when the searched exercise is displayed successfully
Post-condition:	The trainer can view the searched exercise successfully
Business Rules:	 Only authorised users of the system will be able to search a exercise (an administrator, trainer or Super User)
Implementation Constraints and Specifications	None
Assumptions	 The trainer user searching for the exercise is the rightful owner of that account. Trainers will need internet access to the system to view the list of exercises that are on the system.
Open Issues	None



Table 2.1.8.8 Update Exercise

Table 2.1.8.8 Update Exercis		
Author(s): Lyne' Keet		Creation Date:20/06/2022
Version: 1.0		Last Review Date:
Use Case Name:	Update Exercise	Use case type
Use Case ID:	8.8	Business Requirements:
Priority:	HIGH	System Analysis:
Source:	Bester Strength and Conditioning	System Design: □
Primary Business Actor:	Trainer	-
Primary System Actor:	None	
Other Participating Actors:	None	
Other Interested Stakeholders:	None	
Description:	update a specific employee type. T	the employee type record. The use e type has been updated and the
Pre-condition:	The trainer must be logged	
	The exercise must exist on	
Trigger:	The trainer requests to update an e	
Typical Course	Actor Action:	system Response:
	The trainer requests to update a specific exercise record on the system.	2. The system reads and displays the following information regarding the selected exercise record from the Exercise_Name - Exercise_Focus - Exercise_URL from the Exercise_Category table: - Exercise_Category_Name -
	4. The trainer will update the relevant information of the exercise record and select to save the changes made.	 3. The system prompts the trainer to edit pre-populated fields. The fields are populated with the information that was retrieved from the Exercise table. 5. The system validates that all the input fields are not left empty and that the exercise url is a valid youtube link [ALT] 6. The system displays the new entered information and requests the trainer to read and confirm the



		changes that are to be made.
	7. The trainer confirms the update [ALT]	8. The system verifies that the exercise record is not being duplicated in the Exercise table by checking if the entered attributes: - Exercise_Name Matches any existing Exercise Record. [ALT]
		9. The system will update the exercise record to the Exercise table with the following attributes. - Exercise_ID (remains the same) - Exercise_Name - Exercise_Focus - Exercise_URL - Exercise_Category_I D
		10. The system displays a success notification for the successfully updated exercise
Alternate Courses:	step 4 [ALT] Step 8: The exercise record	m displays error messages under valid. Return to step 4. pdating the employee type. Return to
Conclusion:	The use case concludes when the completed and a successful alert b	
Post-condition:	The Exercise record has been updated in the Exercise table has been updated	
Business Rules:	 Only administrators and train 	ners can update exercises
Implementation Constraints and Specifications	None	
Assumptions	 The administrator/trainer updating the exercise is the rightful owner of that account. Administrators/Trainers will need internet access to the system to view the list of exercises that are on the system. 	
Open Issues	None	



Table 2.1.8.9 Delete Exercise

Author(s): Lyne' Keet Version: 1.0		Creation Date:20/06/2022 Last Review Date:
Use Case Name:	Delete Exercise	Use case type
Use Case ID:	8.9	Business Requirements:
Priority:	HIGH	System Analysis:
Source:	Bester Strength and Conditioning	System Design: □
Primary Business Actor:	Trainer	
Primary System Actor:	None	
Other Participating Actors:	None	
Other Interested Stakeholders:	None	
Description:	a specific exercise record. The use	ecord. The use case concludes when
Pre-condition:	The administrator must be IThe exercise record must e	,
Trigger:	The administrator requests to delet system	e an exercise record from the
Typical Course	Actor Action:	system Response:
	The administrator requests to delete a specific exercise record on the system.	2. The system will confirm no lesson records are associated with the selected Exercise_ID as a FK in the Lesson_Plan table, and the Lesson_ID as a FK in the Lesson_Plan entity, the Exercise_ID as a PK in the Exercise entity and the Lesson_ID as a PK in the Lesson_ID as a PK in the Lesson_entity[ALT] 3. The system reads and displays the following attributes for selected exercise from the Exercise table: - Exercise_Name
		- Exercise_Focus - Exercise_URL from the Exercise_Category table: - Exercise_Category_ Name 4. The system prompts the trainer to confirm the deletion of the exercise record



Strengthening	Solutions
Team 7	

	5. The trainer confirms the deletion of the exercise record. [ALT]	6. The system will delete the exercise record to the Exercise table with the following attributes. - Exercise_ID - Exercise_Name - Exercise_Focus - Exercise_URL - Exercise_Category_I D
		The system displays a success notification for the successfully deleted exercise.
Alternate	[ALT] Step 2: The system determines there are lesson record/s	
Courses:	associated with the selected exercise record. The system will notify the trainer, and ask them to ensure no lesson plan records are associated before deletion. The use case terminates	
	[ALT] Step 5: The trainer denies th	e deletion of the exercise record.
On a division.	The use case terminates	
Conclusion:	The use case concludes when the	
Post-condition:	exercise record has been successf	
F USI-COHUILIOH.	database.	m the Exercise table in the system's
Business Rules:		e system will be able to delete an
Dustrioso i (dios.		strator, trainer or super user).
		nust be registered on the system and
		e able to delete an exercise record.
Implementation Constraints and Specifications	None	
Assumptions	None	
Open Issues	None	



Table 2.1.8.10 View Exercise

Author(s): Lyne' Keet		Creation Date:20/06/2022
Version: 1.0		Last Review Date:
Use Case Name:	View Exercise	Use case type
Use Case ID:	8.10	Business Requirements:
Priority:	HIGH	System Analysis:
Source:	Bester Strength and Conditioning	System Design:
Primary Business Actor:	Trainer	
Primary System Actor:	None	
Other Participating Actors:	None	
Other Interested Stakeholders:	None	
Description:	This use case describes the event access the exercise page. The sys the trainer to access the create, se	tem will display the page; allowing arch, edit and delete for exercises.
Pre-condition:	The trainer must be logged	into the system
Trigger:	The trainer requests to view the ex	ercise page
Typical Course	Actor Action:	system Response:
	The trainer requests to view the exercise page. 3. The trainer requests to	The system will display the populated list on the screen from the Exercise table: - Exercise_Name - Exercise_Focus - Exercise_URL from the Exercise_Category table: - Exercise_Category_Name
	search an exercise record [ALT]	4. The system invokes Use Case <u>8.6 "Search Exercise".</u>
Alternate Courses:	[ALT] Step 3a: The trainer requests to create an exercise record. The system invokes Use Case 8.5 "Create Exercise". [ALT] Step 3b: The trainer requests to update an exercise record. The system invokes Use Case 8.7 "Update Exercise". [ALT] Step 3c: The trainer requests to delete an exercise record. The system invokes Use Case 8.8 "Delete Exercise".	
Conclusion:	The system displays a list with all the exercise records and their information with all correlated exercise attributes.	
Post-condition:	The trainer will be able to search on the exercise page.	
Business Rules:	Only authorised users of the system can view the exercise page (an administrator, a trainer and Super user)	
Implementation Constraints and Specifications	None	,
Assumptions	None	
Open Issues	None	





Table 2.1.8.11 Create Exercise Category

Author(s): George Liatos	ioc outogory	Creation Date:20/06/2022
Version: 1.0		Last Review Date:
Use Case Name:	Create Exercise Category	Use case type
Use Case ID:	8.11	Business Requirements:
Priority:	HIGH	System Analysis:
Source:	Bester Strength and Conditioning	System Design: □
Primary Business Actor:	Administrator	, ,
Primary System Actor:	None	
Other Participating Actors:	None	
Other Interested Stakeholders:	None	
Description:	This use case describes the event where the administrator creates a new exercise category on the system. The use case begins when the administrator wants to create a new exercise category record, and enters the needed information required by the system. The system verifies the information and then stores it. The use case concludes when the exercise category record has successfully been created on the system.	
Pre-condition:	The administrator must be I	logged into the system.
Trigger:	The administrator requests to creat system	
Typical Course	Actor Action:	system Response:
	The administrator requests to create a new exercise category record on the system	The system prompts the administrator to enter the following details concerning the exercise category from the Exercise_Category table Exercise_Category_Name Exercise_Category_Description
	The administrator enters the required information into the specified fields	The system verifies that the required data fields are not left empty [ALT]
		5. The system will display the entered information and requests the administrator to read and confirm the entered information
	The administrator confirms the information entered [ALT]	7. The system verifies that the exercise category record is not being duplicated in the Exercise_Category table by checking if the entered attributes: - Exercise_Category_ Name





		- Exercise_Category_
		Description
		Matches any existing exercise
		category record. [ALT]
		8. The system will add the new
		exercise category record to
		the Exercise_Category
		table with the following
		attributes: - Exercise_Category_I
		D (the value of which
		is the previous
		Exercise_Category_I
		D incremented by 1)
		 Exercise_Category_
		Name
		- Exercise_Category_
		Description 9. The System displays a
		success notification for the
		administrator that the
		exercise category has been
		created
Alternate		filled in or the required fields are not
Courses:	in the correct format. The system displays error messages associated	
	with the attributes that are empty o	r Invalid. Return to step 3.
	[ALT] Step 6: The administrator de	nies the information entered is
	correct. Return to step 3	mes the information entered to
	'	
	[ALT] Step 7: The exercise categor	
	system displays an error message	informing the administrator. Return
Canalysian	to step 5.	
Conclusion:	to the database and the system ind	exercise category details are added
	details were added successfully.	icalco willi a meosage mat me
Post-condition:	The new exercise category record has been added to the system in the	
	Exercise_Category table	
Business Rules:	- Only administrator users are able to create new exercise	
levels as a station Co. 1 is 1	categories	
Implementation Constraints and	None	
Specifications Assumptions	None	
Open Issues	None	
C POIT 100000	110110	



Table 2.1.8.12 Search Exercise Category

Table 2.1.8.12 Search Exer	cise Calegory	
Author(s): George Liatos		Creation Date:20/06/2022
Version: 1.0		Last Review Date:
Use Case Name:	Search Exercise Category	Use case type
Use Case ID:	8.12	Business Requirements:
Priority:	HIGH	System Analysis:
Source:	Bester Strength and Conditioning	System Design:
Primary Business Actor:	Administrator	
Primary System Actor:	None	
Other Participating Actors:	None	
Other Interested Stakeholders:	None	
Description: Pre-condition:		information is displayed for that
Trigger:	The administrator requests to search	
mgger.	the system.	on an exercise category record off
Typical Course	Actor Action:	system Response:
Typical Course	Actor Action.	2. The system prompts the
	The administrator requests to search a exercise category record on the system	administrator to enter the following details concerning the exercise category record from the Exercise_Category table: - Exercise_Category_ Name - Exercise_Category_ Description
	The administrator enters the required information as search criteria.	4. The system retrieves and displays a list of all the exercise category records which matches the search criteria entered: - Exercise_Category_Name - Exercise_Category_Description [ALT]
	5. The administrator selects the specific exercise category record they wish to view from the displayed results	6. The system retrieves and displays the following information regarding the exercise category record from the Exercise_Category table: - Exercise_Category_ Name - Exercise_Category_ Description



	-
Alternate Courses:	[ALT] Step 4: No results were found. The use case terminates
Conclusion:	The use case concludes with the searched exercise category record being displayed successfully.
Post-condition:	The administrator can view the searched exercise category record successfully.
Business Rules:	 Only authorised users of the system will be able to search an exercise category record (an administrator). The administrator must be registered on the system and logged into the system to be able to search for an exercise category record.
Implementation Constraints and Specifications	None
Assumptions	None
Open Issues	None

78



Table 2.1.8.13 Update Exercise Category

Table 2.1.8.13 Update Exerc	cise Calegory	
Author(s): George Liatos		Creation Date:20/06/2022
Version: 1.0	I.,	Last Review Date:
Use Case Name:	Update Exercise Category	Use case type
Use Case ID:	8.13	Business Requirements:
Priority:	HIGH	System Analysis:
Source:	Bester Strength and Conditioning	System Design:
Primary Business Actor:	Administrator	
Primary System Actor:	None	
Other Participating Actors:	None	
Other Interested Stakeholders:	None	
Description: Pre-condition:	This use case describes the event where the administrator would like to update a specific exercise category. The use case begins with the administrator requesting to update the exercise category record. The use case concludes when the exercise category has been updated and the new details are saved on the system.	
1 re-condition.	The administrator must be IThe exercise category must	
Trigger:	The administrator requests to updathe system.	
Typical Course	Actor Action:	system Response:
	The administrator requests to update a specific exercise category record on the system.	The system reads and displays the following information regarding the exercise category record from the Exercise_Category table:
		3. The system prompts the administrator to edit the pre-populated fields. The fields are populated with the information that was retrieved from the Exercise_Category table.
	4. The administrator will update the relevant information of the exercise category record and select to save the changes made.	The system validates that all the input fields are not left empty.[ALT]
		The system displays the new entered information and requests the administrator to read and





		confirm the changes that are to be made.
	7. The administrator confirms the updates made to the exercise category record. [ALT]	8. The system verifies that the exercise category record is not being duplicated in the Exercise_Category table by checking if the entered attributes: - Exercise_Category_ Name - Exercise_Category_ Description Matches any existing exercise category record. [ALT]
		9. The system will update the exercise category record to the Exercise_Category table with the following attributes. - Exercise_Category_I D (The Exercise_Category_I D remain the same) - Exercise_Category_ Name - Exercise_Category_ Description
		10. The system displays a success notification for the successfully updated exercise category record.
Alternate Courses:	[ALT] Step 5: All input fields are not filled in or the required fields are not in the correct format. The system displays error messages regarding the attributes that are empty or invalid. Return to step 4.	
	[ALT] Step 7: The administrator de use case terminates	enies the updates to be made. The
	[ALT] Step 8: The exercise category system, the system displays an error administrator. Return to step 6.	
Conclusion:	The use case concludes when the system notifies the administrator that the exercise category changes have been saved successfully to the system.	
Post-condition:	The exercise category details are successfully updated in the Exercise_Category table in the system's database.	
Business Rules:	 Only authorised users of the system will be able to update an exercise category (an administrator). An exercise category cannot be updated if it does not already exist on the system. 	



	 The administrator must be registered on the system and logged into the system to be able to update an exercise category record.
Implementation Constraints and Specifications	None
Assumptions	None
Open Issues	None



Table 2.1.8.14 Delete Exercise Category

Table 2.1.8.14 Delete Exerc	isc Category	
Author(s): George Liatos		Creation Date:20/06/2022
Version: 1.0		Last Review Date:
Use Case Name:	Delete Exercise Category	Use case type
Use Case ID:	8.14	Business Requirements:
Priority:	HIGH	System Analysis:
Source:	Bester Strength and Conditioning	System Design:
Primary Business Actor:	Administrator	
Primary System Actor:	None	
Other Participating Actors:	None	
Other Interested Stakeholders:	None	
Description:	delete a specific exercise category the administrator requesting to delete use case concludes when the exert deleted from the system.	ete the exercise category record. The cise category record has been
Pre-condition:	 The administrator must be I The exercise category reco 	ogged into the system. rd must exist on the system.
Trigger:	The administrator requests to delet system.	e an exercise category record on the
Typical Course	Actor Action:	system Response:
	The administrator requests to delete an exercise category record on the system.	2. The system will confirm no exercise records are associated with the selected Exercise_Category_ID as a FK in the Exercise table. The system will do this by performing an SQL Read query in the .Net Core controller [ALT]
		3. The system reads and displays the following attributes for the selected exercise category record from the Exercise_Category table: - Exercise_Category_ Name - Exercise_Category_ Description
		The system prompts the administrator to confirm the deletion of the exercise category record The system deletes the
	The administrator confirms the deletion of	exercise category record from the



	the exercise category	Exercise_Category table
	record. [ALT]	with the following attributes
		- Exercise_Category_
		Name
		- Exercise_Category_
		Description
		7. The system displays a
		success notification for the
		successfully deleted
Alta wa ata	[A] T 01 0 T 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	exercise category record.
Alternate Courses:	[ALT] Step 2: The system determin	
Courses.		se category record. The system will
	associated. The use case terminate	em to ensure no exercise records are
	associated. The use case terminate	55.
	[ALT] Stan 5: The administrator de	nies the deletion of the sale category
	record. The use case terminates.	The deletion of the sale category
Conclusion:	The use case concludes when the system notifies the administrator that	
	the exercise category record has been successfully deleted from the	
	system.	
	,	
Post-condition:	The exercise category record is de	eted from the Exercise_Category
	table in the system's database.	
Business Rules:	Only authorised users of the system will be able to delete an	
	exercise category record (an administrator).	
	- An exercise category cannot be deleted if it does not already	
	exist on the system.	
	- The administrator must be registered on the system and logged	
	into the system to be able to	delete an exercise category record.
Implementation Constraints and	None	
Specifications	Naca	
Assumptions	None	
Open Issues	None	



Table 2.1.8.15 View Exercise Category

Author(a): Coorgo Listos		Creation Date: 20/06/2022
Author(s): George Liatos Version: 1.0		Creation Date:20/06/2022 Last Review Date:
	View Eversion October	
Use Case Name:	View Exercise Category	Use case type
Use Case ID:	8.11	Business Requirements:
Priority:	HIGH	System Analysis:
Source:	Bester Strength and Conditioning	System Design:
Primary Business Actor:	Administrator	
Primary System Actor:	None	
Other Participating Actors:	None	
Other Interested Stakeholders:	None	
Description:	for exercise category	. The system will display the page; s the create, search, edit and delete
Pre-condition:	The administrator must be logged i	
Trigger:	The administrator requests to view	
Typical Course	Actor Action:	system Response:
	The administrator requests to view the exercise category page.	The system will display the populated list on the screen with the:
	The administrator requests to search an exercise category record. [ALT]	The system invokes Use Case <u>8.8 "Search exercise</u> category".
Alternate Courses:	[ALT] Step 3a: The administrator requests to create an exercise category record. The system invokes Use Case 8.7 "Create exercise category". [ALT] Step 3b: The administrator requests to update an exercise category record. The system invokes Use Case 8.9 "Update exercise category". [ALT] Step 3c: The administrator requests to delete an exercise category record. The system invokes Use Case 8.10 "Delete exercise category".	
Conclusion:	The system displays a list with all the exercise category records and their information.	
Post-condition:	The administrator will be able to search on the exercise category page.	
Business Rules:	Only authorised users of the system can view the exercise category page (an administrator)	
Implementation Constraints and Specifications	None	
Assumptions	None	
Open Issues	None	
	110.10	







2.1.9 Inventory Subsystem

Table 2.1.9.4 Create Write Off Reason

Author(s): Joshua Bester Version: 1.0		Creation Date:20/06/2022 Last Review Date:
Use Case Name:	Create Write-Off Reason	Use case type
Use Case ID:	9.4	Business Requirements:
Priority:	HIGH	System Analysis:
Source:	Bester Strength and Conditioning	System Design:
Primary Business Actor:	Administrator	
Primary System Actor:	None	
Other Participating Actors:	None	
Other Interested Stakeholders:	None	
Pre-condition:	the needed information required by information and then stores it. The write-off reason record has success • The administrator must be I	n. The use case begins when the w write-off reason record, and enters the system. The system verifies the use case concludes when the sfully been created on the system. ogged into the system
Trigger:	The administrator requests to creat system	te a new write-on reason on the
Typical Course	Actor Action:	system Response:
	The administrator requests to create a new write-off reason record on the system	The system prompts the administrator to enter the following details concerning the write-off reason from the Write_Off_Reason table Write_Off_Reason_Description
	The administrator enters the required information in the field provided	The system validates that the Description input field is not left empty. [ALT]
		5. The system displayed the entered information and requests the administrator to read and confirm the entered information
	6. The administrator confirms the information entered [ALT]	7. The system verifies that the write-off reason record is not duplicated in the Write_Off_Reason table by checking if the attribute - Write_Off_Reason_



	Strengthening	Solutio
\Box	Team 7	

	<u> </u>
	8. The system will add the new write-off reason record to the Write_Off_Reason table with the following attributes - Write_Off_Reason_I D(the value of which is the previous Write_Off_Reason_I D incremented by 1) - Write_Off_Reason_ Description
	9. The system displayed a success notification to the administrator that the write-off reason has been created
Alternate Courses:	[ALT] Step 4: All input fields are not filled in or the required fields are not in the correct format. The system displays error messages under the input fields that are empty or invalid. Return to step 3. [ALT] Step 6: The administrator denies the information entered is correct. Return to step 3 [ALT] Step 7: The write-off reason record already exists on the system, the system displays an error message informing the administrator. Return to step 5.
Conclusion:	The use case concludes when the write-off reason details are added to the database and the system indicates with a message that the details were added successfully
Post-condition:	The new write-off reason record has been added to the system in the Write_Off_Reason table
Business Rules:	-
Implementation Constraints and Specifications	None
Assumptions	None
Open Issues	None



Table 2.1.9.5 Search Write Off Reason

Table 2.1.9.5 Search Write (JII Reason	0 11 5 1 52/22/25
Author(s): Joshua Bester		Creation Date:20/06/2022
Version: 1.0		Last Review Date:
Use Case Name:	Search Write-Off Reason	Use case type
Use Case ID:	9.5	Business Requirements:
Priority:	HIGH	System Analysis:
Source:	Bester Strength and Conditioning	System Design:
Primary Business Actor:	Administrator	
Primary System Actor:	None	
Other Participating Actors:	None	
Other Interested Stakeholders:	None	
Description:	a write-off reason on the system. T	search for a write-off reason. The use
Pre-condition:	 The administrator must be I The write-off reason must e 	
Trigger:	The administrator requests to search	ch a write-off reason on the system
Typical Course	Actor Action:	system Response:
	The administrator requests to search for a write-off reason on the system.	The system prompts the administrator to enter the following details concerning the write-off reason from the Write_Off_Reason table: Write_Off_Reason_Description.
	The administrator enters the required information as search criteria.	4. The system retrieves and displays a list of all the write-off reasons with the Write_Off_Reason_Descript ion from the Write_Off_Reason table that matches the search criteria entered. [ALT]
	The administrator selects the specific write-off reason they wish to view from the displayed results.	6. The system retrieves and displays the following information regarding the write-off reason from the Write_Off_Reason table: - Write_Off_Reason_Description
Alternate Courses:	[ALT] Step 4: No results were four	nd. The use case terminates
Conclusion:	The use case concludes with the searched write-off reason being displayed successfully.	
Post-condition:	The administrator can view the searched write-off reason successfully.	
Business Rules:	 Only authorised users of the write-off reason (an administration) 	e system will be able to search a strator).







Logical Narratives

Iteration 6 2022/08/01

Implementation Constraints and Specifications	None
Assumptions	None
Open Issues	None





Table 2.1.9.6 Update Write Off Reason

Table 2.1.9.6 Update Write	Oli Reasoli	0 " D + 00/00/0000
Author(s): Joshua Bester		Creation Date:20/06/2022
Version: 1.0	111 1 1 M " O" D	Last Review Date:
Use Case Name:	Update Write-Off Reason	Use case type
Use Case ID:	9.6	Business Requirements:
Priority:	HIGH	System Analysis:
Source:	Bester Strength and Conditioning	System Design:
Primary Business Actor:	Administrator	
Primary System Actor:	None	
Other Participating Actors:	None	
Other Interested Stakeholders:	None	
Description:	This use case describes the event where the administrator would like to update a specific write-off reason. The use case begins with the administrator requesting to update the write-off reason. The use case concludes when the write-off reason has been updated and the new details are saved on the system.	
Pre-condition:	The administrator must be larger and the surite off reason must be larger.	
Triggor	The write-off reason must e The administrator requests to under	ate a write-off reason on the system.
Trigger: Typical Course	Actor Action:	
Typical Gourse	The administrator requests to update a write-off reason on the system. 4. The administrator will update the relevant information of the write-off reason and select the	2. The system retrieves and displays the following information regarding the write-off reason record from the Write_Off_Reason table: - Write_Off_Reason_Description. 3. The system will prompt the administrator to edit pre-populated fields with the information that was retrieved from the Write_Off_Reason table. 5. The system verifies that the required data fields are not empty. [ALT]
	7. The administrator confirms the updates made to the write-off reason record. [ALT]	6. The system displays the new entered information and requests the administrator to read and confirm the changes that are to be made. 8. The system verifies that the write-off reason record is not being duplicated in the Write_Off_Reason table by



		confirming that the entered attributes: - Refind_Reason_Des cription Do not match any existing write-off reason record. [ALT]
		 The system saves the relevant changes to the Write_Off_Reason_Descript ion in the Write_Off_Reason table.
		10. The system notifies the administrator that the changes have been successfully made.
Alternate Courses:	[ALT] Step 5: The required fields are not entered. The system displays an error notification informing the administrator. Return to step 4. [ALT] Step 7: The administrator denies the updates to be made. The	
	use case terminates [ALT] Step 8:The entered attribute records. The system displays a notification i step	es match existing write-off reason nforming the administrator. Return to
Conclusion:	The use case concludes when the system notifies the administrator that the write-off reason changes have been saved successfully to the system.	
Post-condition:	The write-off reason details are successfully updated in the Write_Off_Reason table in the system's database.	
Business Rules:	 Only authorised users of the system will be able to update a write-off reason (an administrator). 	
Implementation Constraints and Specifications	None	
Assumptions	None	
Open Issues	None	



Table 2.1.9.7 Delete Write Off Reason

Author(s): Joshua Bester		Creation Date:20/06/2022	
Version: 1.0		Last Review Date:	
Use Case Name:	Delete Write-Off Reason	Use case type	
Use Case ID:		Business Requirements:	
	9.7	·	
Priority:	HIGH	System Analysis:	
Source:	Bester Strength and Conditioning	System Design:	
Primary Business Actor:	Administrator		
Primary System Actor:	None		
Other Participating Actors:	None		
Other Interested Stakeholders:	None		
Description:	delete a specific write-off reason re administrator requesting to delete t case concludes when the write-off the system.	This use case describes the event where the administrator would like to delete a specific write-off reason record. The use case begins with the administrator requesting to delete the write-off reason record. The use case concludes when the write-off reason record has been deleted from the system.	
Pre-condition:	The administrator must be IThe write-off reason record		
Trigger:	The administrator requests to delet system.	te a write-off reason record on the	
Typical Course	Actor Action:	system Response:	
	The administrator requests to delete a write-off reason record on the system.	The system will confirm no write-off lines are associated with the selected Write_Off_Reason_ID as a FK in the Write_Off_Line table. [ALT] The system will retrieve and display the:	
		- Write_Off_Reason_ Description from the Write Off_Reason table	
		The system prompts the administrator to confirm the deletion of the write-off reason record	
	5. The administrator confirms the deletion of the write-off reason record. [ALT]	6. The system deletes the write-off reason record from the Write_Off_Reason table with the following attribute: Write_Off_Reason_Descript ion.	
Alternate	[ALT] Step 2: The system determine	7. The system notifies the administrator that the write-off reason record has been successfully deleted.	
Courses:	associated with the selected write-		





	notify the administrator, and ask them to ensure no write-off line records are associated before deletion. The use case terminates.	
	[ALT] Step 5: The administrator denies the deletion of the write-off reason record. The use case terminates.	
Conclusion:	The use case concludes when the system notifies the administrator that the write-off reason record has been successfully deleted from the system.	
Post-condition:	The write-off reason record is deleted from the Write_Off_Reason table in the system's database	
Business Rules:	 Only authorised users of the system will be able to delete a write-off reason record (an administrator). 	
Implementation Constraints and Specifications	None	
Assumptions	None	
Open Issues	None	

92



Table 2.1.9.8 ViewWrite Off Reason

Author(a): Joshua Postor		Creation Date:20/06/2022
Author(s): Joshua Bester Version: 1.0		Last Review Date:
Use Case Name:	View Write-Off Reason	Use case type
Use Case ID:	View Write-Oil Reason	Business Requirements:
	9.8	<u> </u>
Priority:	HIGH	System Analysis:
Source:	Bester Strength and Conditioning	System Design:
Primary Business Actor:	Administrator	
Primary System Actor:	None	
Other Participating Actors:	None	
Other Interested Stakeholders:	None	
Description:	This use case describes the event	where the administrator wants to
·	access the write-off reason page.	The system will display the page;
		s the create, search, edit and delete
	for write-off reason.	
Pre-condition:	The administrator must be logged	into the system.
Trigger:	The administrator requests to view	the write-off reason page.
Typical Course	Actor Action:	system Response:
		2. The system will display the
		populated table on the
	The administrator	screen with the:
	requests to view the	Write_Off_Reason_
	write-off reason page.	Description
		from the retrieved
		Write_Off_Reason records
	3. The administrator	The system invokes Use
	requests to search a	Case <u>9.5 "Search Write-Off</u>
	write-off reason record.	Reason".
Alternate Courses:	[ALT] Step 3a: The administrator requests to create a write-off reason	
Courses.	record. The system invokes Use Case <u>9.4 "Create Write-Off Reason</u> ".	
	[ALT] Step 3b: The administrator i	requests to update write-off reason
	records.	·
	The system invokes Use Case <u>9.6</u>	"Update Write-Off Reason".
	[ALT] Step 3c: The administrator r	requests to delete write-off reason
	records.	<u></u>
	The system invokes Use Case 9.7	
Conclusion:	The system displays a table with all the write-off reasons records and their information.	
Post-condition:	The administrator will be able to search on the write-off reason page	
Business Rules:	- Only authorised users of the system can view the write-off	
	reason page (an administrator)	
Implementation Constraints and Specifications	None	
Assumptions	None	
Open Issues	None	
Орен юмее	I NOTIC	





Table 2.1.9.14 Receive Stock

Author(s): Joshua Bester		Creation Date:20/06/2022
Version: 1.0		Last Review Date:
Use Case Name:	Receive Stock	Use case type
Use Case ID:	9.14	Business Requirements:
Priority:	HIGH	System Analysis:
Source:	Bester Strength and Conditioning	System Design:
Primary Business Actor:		
Primary System Actor:		
Other Participating Actors:		
Other Interested Stakeholders:		
Description:		
Pre-condition:		
Trigger:		
Typical Course	Actor Action:	system Response:
.		
Alternate		
Courses:		
Conclusion: Post-condition:		
Business Rules:		
Implementation Constraints and	-	
Specifications	None	
Assumptions	None	
Open Issues	None	



3. DOCUMENT CONCLUSION

The above document contained the logical narratives for the Administrator Subsystem, Employee Subsystem, Login Subsystem, Sale Subsystem, Booking Subsystem, Client/Member Subsystem, Inventory Subsystem, Lesson Plan Subsystem. These narratives provided a full insight into the processes of these subsystems and the entities, actors, and attributes involved according to the needs of the client.

95

