# H.I.I.T

A scheduling/booking system by:

**TEAM 7**

STRENGTHENING SOLUTIONS

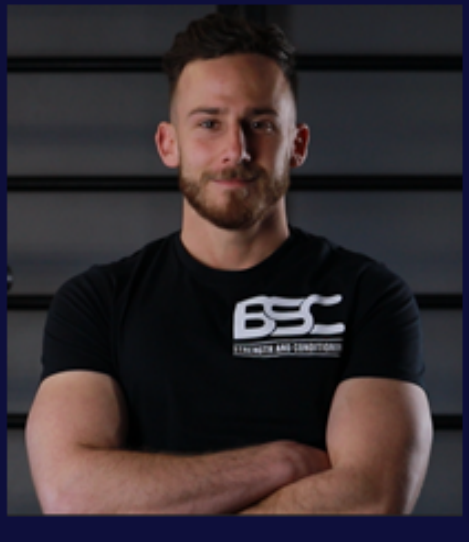

**Luke Partridge**
U19010240
u19010240@tuks.co.za
+27 64 682 0521

**Shannon Noel**
U20502746
u20502746@tuks.co.za
+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

Table of Contents

# 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.

# 2. LOGICAL NARRATIVES

## 2.1 Logical Narratives

### 2.1.2 Employee Subsystem

*Table 2.1.2.1 Create Employee*

| Author(s): Shannon Noel<br>Version: 1.0 | | Creation Date:20/05/2022<br>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 logged into the system. | |
| Trigger: | The administrator requests to create 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:<br>- Employee_Photo<br>- Employee_SA_ID<br>- Employee_Contract_File<br><br>from the **Employee** table<br><br>- Firstname<br>- Lastname<br>- Email<br>- PhoneNumber<br>From the **AspNetUser** table (The Primary Key: User_ID is the Foreign Key in the **Employee** table).<br><br>- Employee_Type_Name<br>from the **Employee_Type** table (the Primary Key: Employee_Type_ID is the Foreign Key in **Employee** table) |

| | | |
|---|---|---|
| | | - Qualification_Description<br>from the **Qualification** table (the Primary Key: Qualification_ID is the Foreign Key in **Employee** table)<br><br>- Name<br>(representing role name)<br>From the **AspNetRoles** table (the Primary Key: RoleId 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).<br><br>- Title_Description<br><br>From the **Title** table (The Primary Key: Title_ID is the Foreign Key in the **User** table). |
| | 2. 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<br>- Name should be text<br>- Surname should be text<br>- Photo is optional<br>- SA_ID should be validated according to a valid South African ID (13 digits).<br>- Email should have an @<br>- Phone Number should be 10 digits.<br>[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:<br>- Employee_SA_ID<br>- Email<br>- PhoneNumber<br>do not match an existing employee record. [ALT] |

| | | |
|---|---|---|
| | | 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:<br>- Employee_ID (the system will auto generate an Employee_ID by taking the last ID and adding one.)<br>- Employee_Photo<br>- Employee_SA_ID<br>- Employee_Contract_File<br>- Qualification_ID<br>- Employee_Type_ID<br>- Title_ID<br>- Role_ID<br>and **AspNetUser** table:<br>- User_ID (the system will auto generate a User_ID by taking the last ID and adding one.)<br>- Firstname<br>- Lastname<br>- Email<br>- 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 Courses: | **[ALT] Step 4**: A required field is not filled in or it is not in the correct format. The system displays an error message to the administrator. Return to step 3.<br>**[ALT] Step 6**: The administrator denies the information entered is correct. Return to step 3.<br>**[ALT] Step 7**: The employee already exists on the system, the system displays an error notification informing the administrator. Return to step 3. | |
| Conclusion: | The use case concludes when the employee has been created in the database and the success message is displayed to the administrator. | |
| Post-condition: | A new employee is created on the system in the **Employee** table.<br>This employee can now access the system. | |
| Business Rules: | - Only authorised users of the system will be able to create an employee (an administrator). | |
| Implementation Constraints and Specifications | None | |
| Assumptions | None | |
| Open Issues | None | |

*Table 2.1.2.2 Search Employee*

| | | |
|---|---|---|
| Author(s): Shannon Noel<br>Version: 1.0 | | Creation Date:20/05/2022<br>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: | This use case describes the event where the administrator wishes to search for an employee on the system. The use case begins when the administrator requests to search for an employee. The use case concludes when the information is displayed for that specific employee. | |
| Pre-condition: | The administrator must be logged into the system.<br>The employee must exist on the system. | |
| Trigger: | The administrator requests to search an employee on the system. | |
| Typical Course | Actor Action: | system Response: |
| | 1. The administrator requests to search for an employee on the system. | 2. The system prompts the administrator to enter any details concerning the employee. |
| | 3. The administrator enters the search criteria. | 4. The system retrieves and displays a list of all the employees which matches the search criteria entered with the following attributes:<br>- Employee_Photo<br>- Employee_SA_ID<br>- Employee_Contract_File<br>from the **Employee** table<br><br>- Employee_Type_Name<br>from the **Employee_Type** table (the Primary Key: Employee_Type_ID is the Foreign Key in **Employee** table)<br><br>- Qualification_Description<br>from the **Qualification** table (the Primary Key: Qualification_ID is the Foreign Key in **Employee** table)<br><br>- Firstname<br>- Lastname<br>- Email<br>- PhoneNumber |

| | | |
|---|---|---|
| | | From the **AspNetUser** table (The Primary Key: User_ID is the Foreign Key in the **Employee** table).<br><br>- Title_Description<br><br>From the **Title** table (The Primary Key: Title_ID is the Foreign Key in the **User** table). [ALT] |
| | 5. The administrator selects the specific employee record they wish to view from the displayed results. | 6. The system retrieves and displays the following information regarding the specific employee record:<br>- Employee_Photo<br>- Employee_SA_ID<br>- Eployee_Contract_File<br>from the **Employee** table<br><br>- Employee_Type_Name<br>from the **Employee_Type** table (the Primary Key: Employee_Type_ID is the Foreign Key in **Employee** table)<br><br>- Qualification_Description<br>from the **Qualification** table (the Primary Key: Qualification_ID is the Foreign Key in **Employee** table)<br><br>- Firstname<br>- Lastname<br>- Email<br>- PhoneNumber<br>From the **AspNetUser** table (The Primary Key: User_ID is the Foreign Key in the **Employee** table).<br><br>- Title_Description<br><br>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 displayed successfully. | |
| Post-condition: | The administrator can view the searched employee record successfully. | |
| Business Rules: | - Only authorised users of the system will be able to search an employee (an administrator).<br>- 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 | |

| | |
|---|---|
| Assumptions | None |
| Open Issues | None |

*Table 2.1.2.3 Update Employee*

| | | |
|---|---|---|
| Author(s): Shannon Noel<br>Version: 1.0 | | Creation Date:20/05/2022<br>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<br>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 logged into the system.<br>• The employee must exist 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:<br><br>- Employee_Photo<br>- Employee_SA_ID<br>- Employee_Contract_File<br><br>from the **Employee** table<br><br>- Qualification_Description<br>from the **Qualification** table (the Primary Key: Qualification_ID is the Foreign Key in **Employee** table)<br><br>- Firstname<br>- Lastname<br>- Email<br>- PhoneNumber<br>From the **AspNetUser** table (The Primary Key: User_ID is the Foreign Key in the **Employee** table). |

| | | |
|---|---|---|
| | | - Title_Description<br><br>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<br>- Name should be text<br>- Surname should be text<br>- Photo is optional<br>- SA_ID should be validated according to a valid South African ID (13 digits).<br>- User_Email should have an @<br>- User_Cell should be 10 digits.<br>[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:<br>- Employee_SA_ID<br>- Firstname<br>- Lastname<br>- Email<br>- PhoneNumber |

| | | |
|---|---|---|
| | | do not match an existing employee record. [ALT] |
| | | 9. The system will update the employee record to the **Employee** table with the following attributes:<br>- Employee_ID (remain the same.)<br>- Employee_Photo<br>- Employee_SA_ID<br>- Employee_Contract_File<br>- Qualification_ID<br>- Employee_Type_ID<br>and **AspNetUser** table:<br>- User_ID (remain the same.)<br>- Firstname<br>- Lastname<br>- Email<br>- PhoneNumber |
| | | 10. The system displays a success notification for the successfully updated employee record. |
| Alternate Courses: | **[ALT] Step 5:** The required 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.<br>**[ALT] Step 7**: The user denies the information entered is correct. Return to step 4.<br>**[ALT] Step 8:** The employee already exists on the system, the system displays an error notification informing 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*

| | | |
|---|---|---|
| Author(s): Shannon Noel<br>Version: 1.0 | | Creation Date:20/05/2022<br>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: | 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: | • The administrator must be logged into the system.<br>• The employee must exist on the system. | |
| Trigger: | The administrator requests to 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:<br>- Employee_Photo<br>- Employee_SA_ID<br>- Employee_Contract_File<br><br>from the **Employee** table<br><br>- Qualification_Description<br>from the **Qualification** table (the Primary Key: Qualification_ID is the Foreign Key in **Employee** table)<br><br>- Firstname<br>- Lastname<br>- Email<br>- PhoneNumber |

| | | |
|---|---|---|
| | | From the **AspNetUser** table (The Primary Key: User_ID is the Foreign Key in the **Employee** table).<br><br>- Name (Role Name) from the **AspNetRoles** table where the User_ID FK is the PK in the **AspNetUsers** table.<br><br>- Title_Description<br><br>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:<br>- Employee_Photo<br>- Employee_SA_ID<br>- Employee_Contract_File<br><br>from the **Employee** table<br><br>- Firstname<br>- Lastname<br>- Email<br>- PhoneNumber<br>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<br><br>**[ALT] Step 4**: The administrator denies the deletion of the employee.The use case terminates. | |
| 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 contain the information of the deleted employee.The deleted employee cannot gain access into the system. | |

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

| | |
|---|---|
| | **[ALT] Step 3b:** The administrator requests to update employee records.<br>The system invokes Use Case 2.3 “Update Employee”.<br><br>**[ALT] Step 3c:** The administrator requests to delete employee records.<br>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 |

### 2.1.3 Login Subsystem

*Table 2.1.3.1 Forgot Password*

| | | |
|---|---|---|
| Author(s): Lyne Keet, Shannon Noel<br>Version: 1.0 | | Creation Date:12/04/2022<br>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 | |
| Description: | This use case describes the event when the user forgot their password and would like to reset the password.The use case begins with the user selecting the forgot password option from the Login screen. The system will request the user to enter their email address, then the system searches for a user profile with an email address that matches the email address entered by the user. Once a match has been made, the system auto-generates a One-Time-Pin that is sent to the cellphone number that is retrieved from the system. The user receives the One-Time-Pin SMS and enters the sequence onto the system. The system verifies if the entered pin matches the sent sequence. The system then prompts the user to enter a new password and confirm the new password. The use case concludes once the system notifies the user that their password has been successfully reset. | |
| 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: |
| | 1. The user will select the forgot password function on the login screen. | 2. 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:<br>- Email<br>- NormalizedEmail<br>[ALT] |
| | | 6. The system retrieves the matching user profile and the following information is read from the **AspNetUser** table<br>- phoneNumber |
| | | 7. The system generates a random One-Time-Pin to send to the user for them to gain access to the system. |
| | | 8. The system will SMS the OTP to the retrieved phoneNumber record. [ALT] |
| | | 9. The system will load a screen prompting the user to enter the OTP sent via SMS |
| | 10. The user accesses the auto-generated SMS and enters the received One-TimePin in a designated input area | |
| | | 11. The system validates the entered One-Time-Pin sequence against the system-generated pin that was sent, confirming a match [ALT] |
| | | 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 password and confirms the new password in the specified fields. | |
| | | 14. The system will validate the entered password to ensure the password matches the confirmed password entered. [ALT] |
| | | 15. The system also validates the password requirements, validating that:<br>● The new password meets the minimum length requirement of 8 characters<br>● The new password contains at least one special character<br>● The new password has one upper case letter.<br>[ALT] |
| | | 16. The system hashes the user's new password and creates a new record in the **Password_History** table, using the following attributes: |

| | | |
|---|---|---|
| Alternate Courses: | | - Password_ID(the value of which is the previous Password_ID incremented by 1)<br>- Password_Date<br>- Hashed_Password<br>- 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. |
| | [**ALT**] **Step 4:** the email field has been left empty or the email data entered have incorrect data formats. The system displays a notification stating the error. Return to step 3.<br>**[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.<br>**[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.<br>**[ALT] Step 11:**The system's One-Time-Pin validation was unsuccessful. The system will display an error message. Return to step 10<br>**[ALT] Step 14:** The new password and confirm password fields do not match. The system will display an error message. Return to step 13<br>**[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.<br>- Passwords should contain at least one special character and one upper case letter.<br>- New password cannot be the same as the old password. | |
| Implementation Constraints and Specifications | - The password must be hashed before it is stored in the database.<br>- 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 | |

*Table 2.1.3.2 Change Password*

| | | |
|---|---|---|
| Author(s): Lyne Keet, Shannon Noel<br>Version: 1.0 | | Creation Date:12/04/2022<br>Last Review Date:26/07/2022 |
| 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 logged in to the system. | |
| Trigger: | The user requests to change their password. | |
| Typical Course | Actor Action: | system Response: |
| | 1. The user requests to change their password. | 2. 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<br>- Hashed_Password<br><br>The system will successfully validate if the current password entered matches the current password associated with the user's profile [ALT] |
| | | 5. The system will validate the new password and the confirmed password matches [ALT] |
| | | 6. The system also validates the password requirements, validating that:<br>• The new password meets the minimum length requirement of 8 characters<br>• The new password contains at least one special characters |

| | | |
|---|---|---|
| | | • The new password has at least one upper case letter.<br>[ALT] |
| | | 7. The system will validate that the new password entered in the field does not match the user's current password stored.<br><br>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.<br>[ALT] |
| | | 8. The system hashes the user's new password and creates a new record in the **Password_History** table, using the following attributes:<br>- Password_ID(the value of which is the previous Password_ID incremented by 1)<br>- Password_Date<br>- Hashed_Password<br>- 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.<br>**[ALT] Step 5:** The new password and confirm password fields do not match. The system will display an error message. Return to step 3<br>**[ALT] Step 6:** The new password does not meet the requirements. The system will display an error message. Return to step 3<br>**[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. | |
| Business Rules: | New password cannot be the same as the old password. | |

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

### 2.1.4 Sale Subsystem

*Table 2.1.4.1 Search Sale Item*

<table>
<tr><td colspan="2">Author(s): Luke Partridge<br>Version: 1.0</td><td colspan="2">Creation Date:12/04/2022<br>Last Review Date:</td></tr>
<tr><td>Use Case Name:</td><td>Search Sales Item</td><td colspan="2">Use case type</td></tr>
<tr><td>Use Case ID:</td><td>4.1</td><td colspan="2" rowspan="3">Business Requirements: □<br><br>System Analysis:<br><br>System Design: □</td></tr>
<tr><td>Priority:</td><td>HIGH</td></tr>
<tr><td>Source:</td><td>Bester Strength and Conditioning</td></tr>
<tr><td>Primary Business Actor:</td><td colspan="3">User</td></tr>
<tr><td>Primary System Actor:</td><td colspan="3">None</td></tr>
<tr><td>Other Participating Actors:</td><td colspan="3">None</td></tr>
<tr><td>Other Interested Stakeholders:</td><td colspan="3">None</td></tr>
<tr><td>Description:</td><td colspan="3">This use case describes the event where the user searches for sale item records on the system. The use case begins when the user requests to search for sale item records. The use case concludes when the information is displayed for the sales item records.</td></tr>
<tr><td>Pre-condition:</td><td colspan="3">None</td></tr>
<tr><td>Trigger:</td><td colspan="3">The user requests to search for sales item records on the system.</td></tr>
<tr><td rowspan="4">Typical Course</td><td>Actor Action:</td><td colspan="2">system Response:</td></tr>
<tr><td>1. The user requests to search for sales item records on the system.</td><td colspan="2">2. The system prompts the user to enter any a search criteria related to any of the sale item attributes</td></tr>
<tr><td>3. The user enters the information as search criteria.</td><td colspan="2">4. The system retrieves and displays a list of all the sale item records that matches with the search criteria entered. [ALT]</td></tr>
<tr><td>5. The user selects the specific sale item record they wish to view from the displayed results.</td><td colspan="2">6. The system retrieves and displays the following information regarding the selected sale item record from the Sale_Item table:<br>- Sale_Item_Name<br>- Sale_Item_Photo<br>- Sale_Item_Description<br>- Sale_Item_Quotable<br>- Sale_Item_Price?<br>- Sale_Item_Quantity?<br><br>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”</td></tr>
</table>

| | |
|---|---|
| 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(s): Luke Partridge<br>Version: 1.0 | | Creation Date:12/04/2022<br>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: | This use case describes the event where the user searches for sale category records on the system. The use case begins when the user requests to search for sale category records. The use case concludes when the information is displayed for the sales category records. | |
| Pre-condition: | None | |
| Trigger: | The user requests to search for sales category records on the system. | |
| Typical Course | Actor Action: | system Response: |
| | 1. The user requests to search for sales category records on the system. | 2. The system prompts the user to enter any one of the following details concerning the sales category records from the **Sale_Category** table:<br>- Sale_Category_Name<br>- Sale_Category_Description |
| | 3. The administrator enters the required information as search criteria. | 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:<br>- Sale_Category_Name<br>- Sale_Category_Description |

| | |
|---|---|
| Alternate Courses: | **[ALT] Step 4:** There are no matching search results. The use case 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 |

*Table 2.1.4.3 Add to Cart*

| | | |
|---|---|---|
| Author(s): Joshua Bester<br>Version: 1.0 | | Creation Date:20/06/2022<br>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: | The use case describes the event where the user would like to add a specific sale item to their cart. The use case starts where the user selects the specific item they would like to add to their cart. The system would retrieve that specific product information from the relevant entities and display it for the user. The use case concludes when the item is successfully added to their cart and a success notification is shown. | |
| Pre-condition: | | |
| Trigger: | The user requests to add a product to their cart | |
| Typical Course | Actor Action: | system Response: |
| | 1. 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)<br>- Sale_Item_Name<br>- Sale_Item_Price<br>- Sale_Item_Stock<br>- Sale_Item_Description |
| | | 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.<br>- Sale_Line_ID(the value of which is the previous Employee_Type_ID incremented by 1)<br>- Sale_Line_Quantity<br>- Sale_Item_ID |
| | | 7. The system displays a success notification to the user that the sale item has been added to the cart. |
| Alternate Courses: | **[ALT] Step 3:** No items in the stock associated with the Sale_Item_ID. The system will notify the user that there is no stock available<br>The use case terminates.<br><br>**[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*

| Author(s): Joshua Bester<br>Version: 1.0 | | Creation Date:20/06/2022<br>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: | The use case describes the event where the user would like to view the content of their cart. The user requests to view their cart, the system then responds by reading the cart information from the relevant entities and displaying the information as well as the total amount due by the user. The use case concludes when the items of the cart is successfully viewed by the user | |
| Pre-condition: | | |
| Trigger: | The user requests to view their cart | |
| Typical Course | Actor Action: | System Response: |
| | 1. 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<br><br>The system will then search for the matching Sale_Line_ID in the **Sale_Line** table and retrieve the following information:<br>- Sale_Line_Quantity<br>- Sale_ID<br>- Sale_Item_ID<br><br>The system will then search for the matching Sale_Item_ID (primary key) in the **Sale_Item** table and retrieve the following information:<br>- Sale_Item_Name<br>- Sale_Item_Price<br>(Using the foreign key Sale_Item_ID from the **Sale_Line** table)<br><br>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<br>[ALT] |
| | | 3. The system will perform the following calculations, and display the |

| | | |
|---|---|---|
| Alternate Courses: | | calculations at the bottom of the listed items in the cart<br>● Cart Subtotal VAT exclusive (Card Total minus the vat amount)<br>● VAT amount (The Cart total vat inclusive multiplies with 15%<br>● Cart Total Vat inclusive (The total of each Sale_Item price multiply by the quantity selected by the user) |
| | 4. The user changes the quantities of the sale_items in their cart [ALT] | 5. The system invokes Use Case 4.5 Edit Cart |
| | **[ALT] Step 2:** No items in the cart associated with the User_ID. The system will notify the user that the cart is empty.<br>The use case terminates.<br><br>**[ALT] Step 4:** The user requests to check out the cart.<br>The system invokes Use Case 4.6 “Checkout Cart ”.<br>The use case terminates. | |
| Conclusion: | The system displays a list with all the sale item records and their quantity in the cart | |
| Post-condition: | All cart information is successfully displayed to the user | |
| Business Rules: | - All users of the system can view their cart | |
| Implementation Constraints and Specifications | None | |
| Assumptions | None | |
| Open Issues | None | |

*Table 2.1.4.5 Edit Cart*

| | | |
|---|---|---|
| Author(s): Joshua Bester<br>Version: 1.0 | | Creation Date:20/06/2022<br>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 in the cart | |
| Trigger: | | |
| Typical Course | Actor Action: | system Response: |
| | 1. 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 |
| | | 5. 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<br>• Cart Subtotal VAT exclusive (Cart Total minus the vat amount)<br>• VAT amount (The Cart total vat inclusive multiplies with |

| | | |
|---|---|---|
| Alternate Courses: | | • Vat_percentage last date retrieved from the **VAT** entity<br>• Cart Total Vat inclusive (The total of each Sale_Item price multiply by the quantity selected by the user) |
| | **[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<br><br>**[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 Specifications | None | |
| Assumptions | None | |
| Open Issues | None | |

*Table 2.1.4.7 View Shop*

| | | |
|---|---|---|
| Author(s): Joshua Bester<br>Version: 1.0 | | Creation Date:20/06/2022<br>Last Review Date: |
| Use Case Name: | View Shop | Use case type |
| Use Case ID: | 4.7 | 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 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 page | |
| Typical Course | Actor Action: | system Response: |
| | 1. The administrator requests to view the shop page | 2. The system will read information regarding each item on the shop screen with their<br>- Sale_Item_Name<br>- Sale_Ite_Photo<br>- Sale_Item_Price<br>- Sale_Item_Quaotable<br>Retrieved from the **Sale_Item** table<br><br>The system will display the retrieved information in cards for the user to view.<br>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 Courses: | **[ALT] Step 3a:** The user requests to add an item to their cart.<br>The system invokes Use Case 4.3 “Add to Cart”.<br><br>**[ALT] Step 3b:** The requests for a quote<br>The system invokes Use Case 4.8 “Request For Quote”. | |
| 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*

| | | |
|---|---|---|
| Author(s): Lyne' Keet<br>Version: 1.0 | | Creation Date:20/06/2022<br>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: | • The administrator must be logged into the system | |
| Trigger: | The administrator requests to create a new refund reason on the system | |
| Typical Course | Actor Action: | system Response: |
| | 1. The administrator requests to create a new refund reason record on the system | 2. The system prompts the administrator to enter the following details concerning the refund reason from the **Refund_Reason** table<br>- Refund_Reason_Description |
| | 3. The administrator enters the required information in the field provided | 4. The system validates that the Description input field is not left empty.<br>[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<br>[ALT] | 7. The system verifies that the refund reason record is not duplicated in the Refund_Reason table by checking if the attribute<br>- Refund_Reason_Description<br>Does not match any existing refund reason record<br>[ALT] |
| | | 8. The system will add the new refund reason record to the **Refund_Reason** table with the following attributes<br>- Refund_Reason_ID(the value of which is the previous Refund_Reason_ID incremented by 1) |

| | | |
|---|---|---|
| Alternate Courses: | | - Refund_Reason_Description |
| | | 9. The system displayed a success notification to the administrator that the refund reason has been created |
| | **[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.<br>**[ALT] Step 6**: The administrator denies the information entered is correct. Return to step 3<br>**[ALT] Step 7**: The refund 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 refund reason details are added to the database and the system indicates 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 Specifications | None | |
| Assumptions | None | |
| Open Issues | None | |

*Table 2.1.4.11 Search Refund Reason*

| | | |
|---|---|---|
| Author(s): Lyne' Keet<br>Version: 1.0 | | Creation Date:20/06/2022<br>Last Review Date: |
| Use Case Name: | Search Refund Reason | Use case type |
| Use Case ID: | 4.11 | 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 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.<br>• The title must exist on the system. | |
| Trigger: | The administrator requests to search a refund reason on the system | |
| Typical Course | Actor Action: | system Response: |
| | 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** table:<br>- Refund_Reason_Description. |
| | 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_Refund** table that matches the search 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** table:<br>- Refund_Reason_Description |
| 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. | |
| Business Rules: | - Only authorised users of the system will be able to search a refund reason (an administrator). | |

| | |
|---|---|
| Implementation Constraints and Specifications | None |
| Assumptions | None |
| Open Issues | None |

*Table 2.1.4.12 Update Refund Reason*

| | | |
|---|---|---|
| Author(s): Lyne' Keet<br>Version: 1.0 | | Creation Date:20/06/2022<br>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: | This use case describes the event where the administrator would like to update a specific refund reason. The use case begins with the administrator requesting to update the refund reason. The use case concludes when the refund reason has been updated and the new details are saved on the system. | |
| Pre-condition: | ● The administrator must be logged into the system.<br>● The refund reason must exist on the system. | |
| Trigger: | The administrator requests to update a refund reason on the system. | |
| Typical Course | Actor Action: | system Response: |
| | 1. The administrator requests to update a refund reason on the system. | 2. The system retrieves and displays the following information regarding the refund reason record from the **Refund_Reason** table:<br>- Refund_Reason_Description. |
| | | 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. | 5. 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: |

| | | |
|---|---|---|
| | | - Refind_Reason_Description<br>Do not match any existing refund reason record.<br>[ALT] |
| | | 9. 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 Courses: | **[ALT] Step 5**: The required fields are not entered. The system displays an error notification informing the administrator. Return to step 4.<br>**[ALT] Step 7**: The administrator denies the updates to be made. The use case terminates<br>**[ALT] Step 8**:The entered attributes match existing refund reason records. The system displays a notification informing the administrator. Return to step | |
| Conclusion: | The use case concludes when the system notifies the administrator that the refund reason changes have been saved successfully to the system. | |
| Post-condition: | The refund reason details are successfully updated in the **Refund_Reason** table in the system's database. | |
| Business Rules: | - Only authorised users of the system will be able to update a refund reason (an administrator). | |
| Implementation Constraints and Specifications | None | |
| Assumptions | None | |
| Open Issues | None | |

*Table 2.1.4.13 Delete Refund Reason*

| | | |
|---|---|---|
| Author(s): Lyne' Keet<br>Version: 1.0 | | Creation Date:20/06/2022<br>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: | This use case describes the event where the administrator would like to delete a specific refund reason record. The use case begins with the administrator requesting to delete the refund reason record. The use case concludes when the refund reason record has been deleted from the system. | |
| Pre-condition: | • The administrator must be logged into the system.<br>• The refund reason record must exist on the system. | |
| Trigger: | The administrator requests to delete a refund reason record on the system. | |
| Typical Course | Actor Action: | System Response: |
| | 1. The administrator requests to delete a refund reason record on the system. | 2. The system will confirm no refund records are associated with the selected Refund_Reason_ID as a FK in the **Refund** table. [ALT] |
| | | 3. The system will retrieve and display the:<br>- Refund_Reason_Description from the **Refund_Reason** table |
| | | 4. The system prompts the administrator to confirm the deletion of the refund reason record |
| | 5. The administrator confirms the deletion of the refund reason record. [ALT] | 6. The system deletes the refund 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.<br><br>**[ALT] Step 5**: The administrator denies the deletion of the refund reason record. The use case terminates. | |
| Conclusion: | The use case concludes when the system notifies the administrator that the refund reason record has been successfully deleted from the system. | |

| | |
|---|---|
| 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 |

*Table 2.1.4.14 View Refund Reason*

| | | |
|---|---|---|
| Author(s): Lyne' Keet<br>Version: 1.0 | | Creation Date:20/06/2022<br>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: | This use case describes the event where the administrator wants to access the refund reason page. The system will display the page; allowing the administrator to access the create, search, edit and delete for refund reasons. | |
| Pre-condition: | The administrator must be logged into the system. | |
| Trigger: | The administrator requests to view the refund reason page. | |
| Typical Course | Actor Action: | system Response: |
| | 1. The administrator requests to view the refund reason page. | 2. The system will display the populated table on the screen with the:<br>- Refund_Reason_Description<br>from the retrieved **Refund_Reason** records |
| | 3. The administrator requests to search a refund reason record. | 4. The system invokes Use Case 4.11 "Search Refund Reason". |
| Alternate Courses: | **[ALT] Step 3a:** The administrator requests to create a refund reason record.<br>The system invokes Use Case 4.10 "Create Refund Reason".<br><br>**[ALT] Step 3b:** The administrator requests to update refund reason records.<br>The system invokes Use Case 4.12 "Update Refund Reason".<br><br>**[ALT] Step 3c:** The administrator requests to delete refund reason records.<br>The system invokes Use Case 4.13 "Delete Refund Reason". | |
| Conclusion: | The system displays a table with all the refund reasons records and their information. | |
| Post-condition: | The administrator will be able to search on the refund reason page | |
| Business Rules: | - Only authorised users of the system can view the refund reason page (an administrator) | |
| Implementation Constraints and Specifications | None | |
| Assumptions | None | |
| Open Issues | None | |

### 2.1.5 Client/Member Subsystem

*Table 2.1.5.1 Register Client*

| Author(s): Lyne' Keet<br>Version: 1.0 | | Creation Date:20/06/2022<br>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: | The use case describes the event where a potential client requests to register themselves on The BSC system. The use case begins when the potential customer wishes to register on the system. The system will prompt the potential client to enter their information. The potential client will enter their details, where it will be validated by the system..The use case concludes when the system indicates that the potential client's details were added successfully to system | |
| Pre-condition: | | |
| Trigger: | The potential customer wishes to register themselves onto the BSC system | |
| Typical Course | Actor Action: | system Response: |
| | 1. The potential client requests to add register themselves onto the BSC system | 2. The system prompts the client to enter the following details<br>- FirstName<br>- LastName<br>- PhoneNumber<br>- Title<br>- Email<br>- Password<br>concerning the client from the **AspNetUsers** table<br>And the<br>- Title_Name<br>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<br>- all the input fields are not left empty<br>- Phone number consists of 10 digits<br>- Email consists a "@' sign<br>- The password length is more than 8 characters<br>- The password consists of 1 special character<br>- The password consists of one capital letter |

| | | |
|---|---|---|
| | | - Confirms that the password and confirm password fields are matched<br>[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<br>- Id(the value of which is the previous Id incremented by 1)<br>- FirstName<br>- LastName<br>- PhoneNumber<br>- Email<br>- Title<br>- 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 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. | |
| 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*

| Author(s): Shannon Noel<br>Version: 1.0 | | Creation Date:20/06/2022<br>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 where the administrator creates a new schedule on the system. The use case begins when the administrator wants to create a new schedule 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 schedule record has successfully been created on the system. | |
| Pre-condition: | ● The administrator must be logged into the system | |
| Trigger: | The administrator requests to create a new schedule on the system | |
| Typical Course | Actor Action: | system Response: |
| | 1. 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<br>- Venue_Name<br>- Booking_Type_Name<br>- Lesson_Name<br>- Date_Session<br>  - Date_session_StartTime<br>  - Date_Session_EndTime<br>- Employee_Name |
| | 3. The administrator enters the required information in the fields provided | 4. The system validates that the input fields are not left empty.<br>The system validates the following fields to ensure that valid inputs are entered<br>- The date selected does not precede the current date at |

| | | |
|---|---|---|
| | | the time of creation [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 will add the new schedule record to the **Schedule** table with the following attributes<br>- Schedule_ID(the value of which is the previous Schedule_ID incremented by 1)<br>- Venue_ID (which is retrieved from the **Venue** table using the Venue_Name; selected from the dropdown list)<br>- Booking_Type_ID (which is retrieved from the **Booking_Type** table using the Booking_Type_Name; selected from the dropdown list)<br>- Lesson_ID (which is retrieved from the **Lesson** table using the Lesson_Name; selected from the dropdown list)<br>- 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)<br>- Employee_ID (which is retrieved from the **Employee** table using the Employee_Name; selected from the dropdown list) |

| | | |
|---|---|---|
| Alternate Courses: | | 8. The system displayed a success notification to the administrator that the schedule event record has been created |
| | **[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.<br>**[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>
<tr><td>Author(s): Shannon Noel<br>Version: 1.0</td><td></td><td colspan="2">Creation Date:20/06/2022<br>Last Review Date:</td></tr>
<tr><td>Use Case Name:</td><td>Search Schedule</td><td colspan="2">Use case type</td></tr>
<tr><td>Use Case ID:</td><td>7.2</td><td colspan="2">Business Requirements: □</td></tr>
<tr><td>Priority:</td><td>HIGH</td><td colspan="2">System Analysis:</td></tr>
<tr><td>Source:</td><td>Bester Strength and Conditioning</td><td colspan="2">System Design: □</td></tr>
<tr><td>Primary Business Actor:</td><td colspan="3">Administrator</td></tr>
<tr><td>Primary System Actor:</td><td colspan="3">None</td></tr>
<tr><td>Other Participating Actors:</td><td colspan="3">None</td></tr>
<tr><td>Other Interested Stakeholders:</td><td colspan="3">None</td></tr>
<tr><td>Description:</td><td colspan="3">This use case describes the event where the administrator searches for a schedule on the system. The use case begins when the administrator selects the option to search for a schedule. The use case concludes when the information is displayed for that specific schedule.</td></tr>
<tr><td>Pre-condition:</td><td colspan="3">• The administrator must be logged into the system.<br>• The schedule must exist on the system.</td></tr>
<tr><td>Trigger:</td><td colspan="3">The administrator requests to search a schedule on the system</td></tr>
<tr><td rowspan="4">Typical Course</td><td>Actor Action:</td><td colspan="2">system Response:</td></tr>
<tr><td>1. The administrator requests to search for a schedule on the system.</td><td colspan="2">2. The system prompts the administrator to enter the following details concerning the schedule from the Schedule table:<br>- Date_Session.</td></tr>
<tr><td>3. The administrator enters the required information as search criteria.</td><td colspan="2">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]</td></tr>
<tr><td>5. The administrator selects the specific schedule they wish to view from the displayed results.</td><td colspan="2">6. The system retrieves and displays the following information regarding the schedule from the Schedule table:<br>- Venue_Name<br>- Booking_Type_Name<br>- Lesson_Name<br>- Date_Session<br>- Employee_Name</td></tr>
<tr><td>Alternate Courses:</td><td colspan="3">[ALT] Step 4: No results were found. The use case terminates</td></tr>
<tr><td>Conclusion:</td><td colspan="3">The use case concludes with the retrieved schedule being displayed successfully.</td></tr>
<tr><td>Post-condition:</td><td colspan="3">The administrator can view the retrieved schedule successfully.</td></tr>
</table>

| | |
|---|---|
| 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*

| | | |
|---|---|---|
| Author(s): Shannon Noel<br>Version: 1.0 | | Creation Date:20/06/2022<br>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: | This use case describes the event where the administrator would like to update a specific schedule. The use case begins with the administrator requesting to update the schedule. The use case concludes when the schedule has been updated and the new details are saved on the system. | |
| Pre-condition: | • The administrator must be logged into the system.<br>• The schedule must exist on the system. | |
| Trigger: | The administrator requests to update a schedule on the system. | |
| Typical Course | Actor Action: | system Response: |
| | 1. 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:<br>- Venue_Name<br>- Booking_Type_Name<br>- Lesson_Name<br>- Date_Session<br>- Date_Session_Start_Time<br>- Date_Session_End_Time<br>- Employee_Name |
| | | 3. The system will prompt the administrator to edit pre-populated fields with the information that was retrieved from the **Schedule** table. |
| | 4. The administrator will update the relevant information of the schedule and select the option to save the changes made. | 5. 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 schedule record. [ALT] | 8. The system saves the relevant changes to the **Schedule** table.<br>- Venue_ID (which is retrieved from the **Venue** table using the Venue_Name; selected from the dropdown list)<br>- Booking_Type_ID (which is retrieved from the **Booking_Type** table using the Booking_Type_Name; selected from the dropdown list)<br>- Lesson_ID (which is retrieved from the **Lesson** table using the Lesson_Name; selected from the dropdown list)<br>- 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)<br>- Employee_ID (which is retrieved from the **Employee** table using the Employee_Name; selected from the dropdown list) |
| | | 9. The system notifies the administrator that the changes have been successfully made. |
| Alternate Courses: | **[ALT] Step 5**: The required fields are not entered or the required fields are not in the correct format. The system displays an error notification informing the administrator. Return to step 4.<br>**[ALT] Step 7**: The administrator denies the updates to be made. The use case terminates | |
| Conclusion: | The use case concludes when the system notifies the administrator that the schedule changes have been saved successfully to the system. | |

| | |
|---|---|
| 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*

| | | |
|---|---|---|
| Author(s): Shannon Noel<br>Version: 1.0 | | Creation Date:20/06/2022<br>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: | This use case describes the event where the administrator would like to delete a specific schedule record. The use case begins with the administrator requesting to delete the schedule record. The use case concludes when the schedule record has been deleted from the system. | |
| Pre-condition: | • The administrator must be logged into the system.<br>• The schedule record must exist on the system. | |
| Trigger: | The administrator requests to delete a schedule record on the system. | |
| Typical Course | Actor Action: | system Response: |
| | 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:<br>- Venue_Name<br>- Booking_Type_Name<br>- Lesson_Name<br>- Date_Session<br>- Employee_Name<br>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:<br>- Venue_ID<br>- Booking_Type_ID<br>- Lesson_ID<br>- Date_Session_ID<br>- Employee_ID |
| | | 7. The system notifies the administrator that the |

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

*Table 2.1.7.5 View Schedule*

| | | |
|---|---|---|
| Author(s): Shannon Noel<br>Version: 1.0 | | Creation Date:20/06/2022<br>Last Review Date: |
| Use Case Name: | View Schedule | Use case type |
| Use 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 into the system. | |
| Trigger: | The administrator requests to view the schedule page. | |
| Typical Course | Actor Action: | system Response: |
| | 1. The administrator requests to view the schedule page. | 2. The system will display the populated calendar on the screen with the:<br>- Date_Session<br>from the retrieved **Schedule** records |
| | 3. The administrator requests to search a schedule record. | 4. The system invokes Use Case 7.2 “Search Schedule”. |
| Alternate Courses: | **[ALT] Step 3a:** The administrator requests to create a schedule record. The system invokes Use Case 7.1 “Create Schedule”.<br><br>**[ALT] Step 3b:** The administrator requests to update schedule records. The system invokes Use Case 7.3 “Update Schedule”.<br><br>**[ALT] Step 3c:** The administrator requests to delete schedule records. The system invokes Use Case 7.4 “Delete Schedule”. | |
| Conclusion: | The system displays a calendar with all the schedule records. | |
| 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 schedule page (an administrator) | |
| Implementation Constraints and Specifications | None | |
| Assumptions | None | |
| Open Issues | None | |

*Table 2.1.7.6 Create Booking Type*

| | | |
|---|---|---|
| Author(s): Luke Partridge<br>Version: 1.0 | | Creation Date:20/06/2022<br>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: | This use case describes the event where the administrator creates a new booking type on the system. The use case begins when the administrator wants to create a new booking type 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 booking type record has successfully been created on the system. | |
| Pre-condition: | • The administrator must be logged into the system. | |
| Trigger: | The administrator requests to create 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<br>- Booking_Type_Name<br>- Booking_Type_Description |
| | 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:<br>- Booking_Type_Name<br>- Booking_Type_Description<br>Matches any existing booking type record. [ALT] |

| | | |
|---|---|---|
| Alternate Courses: | | 16. The system will add the new booking type record to the **Booking_Type** table with the following attributes:<br>- Booking_Type_ID (the value of which is the previous Booking_Type_ID incremented by 1)<br>- Booking_Type_Name<br>- Booking_Type_Description |
| | | 17. The System displays a success notification for the administrator that the booking type has been created |
| | **[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 associated with the attributes that are empty or invalid. Return to step 3.<br>**[ALT] Step 6**: The administrator denies the information entered is correct. Return to step 3<br>**[ALT] Step 7**: The booking type 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 booking type details are added to the database and the system indicates with a message that the details were added successfully. | |
| Post-condition: | The new booking type record has been added to the system in the **Booking_Type** table | |
| Business Rules: | - Only administrator users are able to create new booking types | |
| Implementation Constraints and Specifications | None | |
| Assumptions | None | |
| Open Issues | None | |

*Table 2.1.7.7 Search Booking Type*

| | | |
|---|---|---|
| Author(s): Luke Partridge<br>Version: 1.0 | | Creation Date:20/06/2022<br>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 | |
| Description: | This use case describes the event where the administrator searches for a booking type record on the system. The use case begins when the administrator requests to search for a booking type record. The use case concludes when the information is displayed for that specific booking type record. | |
| Pre-condition: | • The administrator must be logged into the system. | |
| Trigger: | The administrator requests to search a booking type record on the system. | |
| Typical Course | Actor Action: | system Response: |
| | 1. The administrator requests to search a booking type record on the system | 2. The system prompts the administrator to enter the following details concerning the booking type record from the **Booking_Type** table:<br>- Booking_Type_Name<br>- Booking_Type_Description |
| | 3. 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:<br>- Booking_Type_Name<br>- Booking_Type_Description<br>[ALT] |
| | 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:<br>- Booking_Type_Name<br>- Booking_Type_Description |
| Alternate | **[ALT] Step 4:** No results were found. 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).<br>- 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*

| | | |
|---|---|---|
| Author(s): Luke Partridge<br>Version: 1.0 | | Creation Date:20/06/2022<br>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: | This use case describes the event where the administrator would like to update a specific booking type. The use case begins with the administrator requesting to update the booking type record. The use case concludes when the booking type has been updated and the new details are saved on the system. | |
| Pre-condition: | ● The administrator must be logged into the system.<br>● The booking type must exist on the system. | |
| Trigger: | The administrator requests to update a booking type record on the system. | |
| Typical Course | Actor Action: | system Response: |
| | 1. The administrator requests to update a specific booking type record on the system. | 2. The system reads and displays the following information regarding the booking type record from the **Booking_Type** table:<br>- Booking_Type_Name<br>- Booking_Type_Description |
| | | 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:<br>- Booking_Type_Name<br>- Booking_Type_Description<br>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.<br>- Booking_Type_ID (The Booking_Type_ID remain the same)<br>- Booking_Type_Name<br>- Booking_Type_Description |
| | | 9. The system displays a success notification for the successfully updated booking type 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.<br>**[ALT] Step 7**: The administrator denies the updates to be made. The use case terminates<br>**[ALT] Step 8**: The booking type record already exists on the system, the system displays an error message informing the administrator. Return to step 6. | |
| Conclusion: | The use case concludes when the system notifies the administrator that the booking type changes have been saved successfully to the system. | |
| Post-condition: | The booking type details are successfully updated in the Booking_Type table in the system's database. | |
| Business Rules: | - Only authorised users of the system will be able to update a booking type (an administrator).<br>- A booking type cannot be updated if it does not already exist on the system.<br>- The administrator must be registered on the system and logged into the system to be able to update a booking type record. | |
| Implementation Constraints and Specifications | None | |
| Assumptions | None | |
| Open Issues | None | |

*Table 2.1.7.9 Delete Booking Type*

| | | |
|---|---|---|
| Author(s): Luke Partridge<br>Version: 1.0 | | Creation Date:20/06/2022<br>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: | This use case describes the event where the administrator would like to delete a specific booking type record. The use case begins with the administrator requesting to delete the booking type record. The use case concludes when the booking type record has been deleted from the system. | |
| Pre-condition: | • The administrator must be logged into the system.<br>• The booking type record must exist on the system. | |
| Trigger: | The administrator requests to delete a booking type record on the system. | |
| Typical Course | Actor Action: | system Response: |
| | 1. The administrator requests to delete a booking type record on the system. | 1. 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:<br>- Booking_Type_Name<br>- Booking_Type_Description |
| | | 3. The system prompts the administrator to confirm the deletion of the booking type record |
| | 3. The administrator confirms the deletion of the booking type record. [ALT] | 3. The system deletes the booking type record from the **Booking_Type** table with the following attributes |

| | | |
|---|---|---|
| Alternate Courses: | | - Booking_Type_Name<br>- Booking_Type_Description |
| | | 3. The system displays a success notification for the successfully deleted booking type record. |
| | **[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.<br>**[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).<br>- A booking type cannot be deleted if it does not already exist on the system.<br>- 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 | |
| Open Issues | None | |

*Table 2.1.7.10 View Booking Type*

| Author(s): Luke Partridge<br>Version: 1.0 | | Creation Date:20/06/2022<br>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: | This use case describes the event where the administrator wants to access the booking type page. The system will display the page; allowing the administrator to access the create, search, edit and delete for booking type. . | |
| Pre-condition: | The administrator must be logged into the system. | |
| Trigger: | The administrator requests to view the booking type page. | |
| Typical Course | Actor Action: | system Response: |
| | 1. The administrator requests to view the booking type page. | 2. The system will display the populated list on the screen with the:<br>- Booking_Type_Name<br>- Booking_Type_Description<br>from the retrieved **Booking_Type** records. |
| | 3. The administrator requests to search a booking type record. [ALT] | 4. The system invokes Use Case 7.7 “Search Booking Type”. |
| Alternate Courses: | **[ALT] Step 3a:** The administrator requests to create a booking type record.<br>The system invokes Use Case 7.6 “Create Booking Type”.<br>**[ALT] Step 3b:** The administrator requests to update a booking type record.<br>The system invokes Use Case 7.8 “Update Booking Type”.<br>**[ALT] Step 3c:** The administrator requests to delete a booking type record.<br>The system invokes Use Case 7.9 “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 search on the booking type page. | |
| Business Rules: | - Only authorised users of the system can view the booking type page (an administrator) | |
| Implementation Constraints and Specifications | None | |
| Assumptions | None | |
| Open Issues | None | |

### 2.1.8 Lesson Plan Subsystem

*Table 2.1.8.6 Create Exercise*

| Author(s): Lyne' Keet<br>Version: 1.0 | | Creation Date:20/06/2022<br>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 create a new exercise on the system | |
| Typical Course | Actor Action: | system Response: |
| | 1. The administrator requests to create a exercise record on the system | 2. The system prompts the trainer to enter the following details concerning the exercise from the **Exercise** table<br>- Exercise_Name<br>- Exercise_Focus<br>- Exercise_URL |
| | | 3. The system will retrieve the list of exercise categories with their:<br>- Exercise_Category_Name<br>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.<br>The system validates the following fields to ensure that valid inputs are entered<br>- The Exercise Name cannot be left empty<br>- The Exercise Url cannot be left empty |

| | | |
|---|---|---|
| | | - The Exercise Focus cannot be left empty<br>- Validate url is a valid youtube link<br>- The Exercise Category is selected [ALT] |
| | | 6. The system displayed the entered information and requests the trainer to read and confirm the entered information |
| | 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:<br>- Exercise_Name<br>Matches any existing Exercise Record. [ALT] |
| | | 9. The system will add the new exercise record to the **Exercise**table with the following attributes.<br>- Exercise_ID (the value of which is the previous Exercise_ID incremented by 1)<br>- Exercise_Name<br>- Exercise_Focus<br>- Exercise_URL<br>- 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<br>**[ALT] Step 7**: The administrator denies the information entered is correct. Return to step 4<br>**[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 | None | |
| Assumptions | None | |
| Open Issues | None | |

*Table 2.1.8.7 Search Exercise*

| | | |
|---|---|---|
| Author(s): Lyne' Keet<br>Version: 1.0 | | Creation Date:20/06/2022<br>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 where the trainer searches for an exercise record on the system. The use case begins when the trainer requests to search for an exercise record. The use case concludes when the information is displayed for that specific exercise. | |
| Pre-condition: | • The trainer must be logged into the system | |
| Trigger: | The administrator requests to search an exercise record on the system | |
| Typical Course | Actor Action: | system Response: |
| | 1. The trainer requests to search an exercise record on the system | 2. The system prompts the trainer to enter a search criteria related to any of the attributed founded in the **Exercise** table |
| | 3. 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**<br><br>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<br>from the **Exercise** table:<br>- Exercise_Name<br>- Exercise_Focus<br>- Exercise_URL<br>from the **Exercise_Category** table:<br>- Exercise_Category_Name |

| | |
|---|---|
| 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.<br>- 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*

| Author(s): Lyne' Keet<br>Version: 1.0 | | Creation Date:20/06/2022<br>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: | This use case describes the event where the administrator would like to update a specific employee type. The use case begins with the administrator requesting to update the employee type record. The use case concludes when the employee type has been updated and the new details are saved on the system. | |
| Pre-condition: | • The trainer must be logged into the system.<br>• The exercise must exist on the system. | |
| Trigger: | The trainer requests to update an exercise record on the system. | |
| Typical Course | Actor Action: | system Response: |
| | 1. 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** table<br>- Exercise_Name<br>- Exercise_Focus<br>- Exercise_URL<br>from the **Exercise_Category** table:<br>- Exercise_Category_ Name<br>- |
| | | 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**.** |
| | 4. The trainer will update the relevant information of the exercise record and select to save the changes made. | 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:<br>- Exercise_Name<br>Matches any existing Exercise Record. [ALT] |
| | | 9. The system will update the exercise record to the **Exercise** table with the following attributes.<br>- Exercise_ID (remains the same)<br>- Exercise_Name<br>- Exercise_Focus<br>- Exercise_URL<br>- Exercise_Category_ID |
| | | 10. The system displays a success notification for the successfully updated exercise |
| 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.<br>**[ALT] Step 7:** The trainer denies updating the employee type. Return to step 4<br>**[ALT] Step 8**: The exercise record already exists on the system, the system displays an error message informing the trainer. Return to step 6. | |
| Conclusion: | The use case concludes when the update has been successfully completed and a successful alert box is displayed. | |
| Post-condition: | The Exercise record has been updated in the **Exercise** table has been updated | |
| Business Rules: | - Only administrators and trainers can update exercises | |
| Implementation Constraints and Specifications | None | |
| Assumptions | - The administrator/trainer updating the exercise is the rightful owner of that account.<br>- 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<br>Version: 1.0 | | Creation Date:20/06/2022<br>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: | This use case describes the event where the trainer would like to delete a specific exercise record. The use case begins with the trainer requesting to delete the exercise record. The use case concludes when the exercise record has been successfully deleted from the system. | |
| Pre-condition: | • The administrator must be logged into the system.<br>• The exercise record must exist on the system. | |
| Trigger: | The administrator requests to delete an exercise record from the system | |
| Typical Course | Actor Action: | system Response: |
| | 1. 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** entity[ALT] |
| | | 3. The system reads and displays the following attributes for selected exercise<br>from the **Exercise** table:<br>- Exercise_Name<br>- Exercise_Focus<br>- Exercise_URL<br>from the<br>**Exercise_Category** table:<br>- Exercise_Category_ Name |
| | | 4. The system prompts the trainer to confirm the deletion of the exercise record |

| | | |
|---|---|---|
| Alternate Courses: | 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.<br>- Exercise_ID<br>- Exercise_Name<br>- Exercise_Focus<br>- Exercise_URL<br>- Exercise_Category_ID |
| | | 7. The system displays a success notification for the successfully deleted exercise. |
| | **[ALT] Step 2**: The system determines there are lesson record/s 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<br>**[ALT] Step 5**: The trainer denies the deletion of the exercise record. The use case terminates | |
| Conclusion: | The use case concludes when the system notifies the trainer that the exercise record has been successfully deleted from the system. | |
| Post-condition: | The exercise record is deleted from the **Exercise** table in the system's database. | |
| Business Rules: | - Only authorised users of the system will be able to delete an exercise record (an administrator, trainer or super user).<br>- The administrator/trainer must be registered on the system and logged into the system to be 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<br>Version: 1.0 | | Creation Date:20/06/2022<br>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 where the administrator wants to access the exercise page. The system will display the page; allowing the trainer to access the create, search, edit and delete for exercises. | |
| Pre-condition: | • The trainer must be logged into the system | |
| Trigger: | The trainer requests to view the exercise page | |
| Typical Course | Actor Action: | system Response: |
| | 1. The trainer requests to view the exercise page. | 2. The system will display the populated list on the screen from the **Exercise** table:<br>- Exercise_Name<br>- Exercise_Focus<br>- Exercise_URL<br>from the **Exercise_Category** table:<br>- Exercise_Category_ Name |
| | 3. The trainer requests to search an exercise record [ALT] | 4. The system invokes Use Case 8.6 "Search Exercise". |
| Alternate Courses: | **[ALT] Step 3a:** The trainer requests to create an exercise record. The system invokes Use Case 8.5 "Create Exercise ".<br><br>**[ALT] Step 3b:** The trainer requests to update an exercise record. The system invokes Use Case 8.7 "Update Exercise".<br><br>**[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<br>Version: 1.0 | | Creation Date:20/06/2022<br>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 logged into the system. | |
| Trigger: | The administrator requests to create a new exercise category on the system | |
| Typical Course | Actor Action: | system Response: |
| | 1. The administrator requests to create a new exercise category record on the system | 2. The system prompts the administrator to enter the following details concerning the exercise category from the **Exercise_Category** table<br>- Exercise_Category_ Name<br>- Exercise_Category_ Description |
| | 3. The administrator enters the required information into the specified fields | 4. 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 |
| | 6. 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:<br>- Exercise_Category_ Name |

| | | |
|---|---|---|
| | | - Exercise_Category_Description<br>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:<br>- Exercise_Category_ID (the value of which is the previous Exercise_Category_ID incremented by 1)<br>- Exercise_Category_Name<br>- Exercise_Category_Description |
| | | 9. The System displays a success notification for the administrator that the exercise category 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 associated with the attributes that are empty or invalid. Return to step 3.<br><br>**[ALT] Step 6**: The administrator denies the information entered is correct. Return to step 3<br><br>**[ALT] Step 7**: The exercise category 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 exercise category details are added to the database and the system indicates with a message that the details were added successfully. | |
| 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 categories | |
| Implementation Constraints and Specifications | None | |
| Assumptions | None | |
| Open Issues | None | |

*Table 2.1.8.12 Search Exercise Category*

| | | |
|---|---|---|
| Author(s): George Liatos<br>Version: 1.0 | | Creation Date:20/06/2022<br>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: | This use case describes the event where the administrator searches for an exercise category record on the system. The use case begins when the administrator requests to search for an exercise category record. The use case concludes when the information is displayed for that specific exercise category record. | |
| Pre-condition: | • The administrator must be logged into the system. | |
| Trigger: | The administrator requests to search an exercise category record on the system. | |
| Typical Course | Actor Action: | system Response: |
| | 1. The administrator requests to search a exercise category record on the system | 2. The system prompts the administrator to enter the following details concerning the exercise category record from the **Exercise_Category** table:<br>- Exercise_Category_Name<br>- Exercise_Category_Description |
| | 3. 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:<br>- Exercise_Category_Name<br>- Exercise_Category_Description<br>[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:<br>- Exercise_Category_Name<br>- 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).<br>- 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 |

*Table 2.1.8.13 Update Exercise Category*

| Author(s): George Liatos<br>Version: 1.0 | | Creation Date:20/06/2022<br>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: | 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. | |
| Pre-condition: | • The administrator must be logged into the system.<br>• The exercise category must exist on the system. | |
| Trigger: | The administrator requests to update an exercise category record on the system. | |
| Typical Course | Actor Action: | system Response: |
| | 1. The administrator requests to update a specific exercise category record on the system. | 2. The system reads and displays the following information regarding the exercise category record from the **Exercise_Category** table:<br>- Exercise_Category_Name<br>- Exercise_Category_Description |
| | | 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. | 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 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:<br>- Exercise_Category_Name<br>- Exercise_Category_Description<br>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.<br>- Exercise_Category_ID (The Exercise_Category_ID remain the same)<br>- Exercise_Category_Name<br>- 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.<br><br>**[ALT] Step 7**: The administrator denies the updates to be made. The use case terminates<br><br>**[ALT] Step 8**: The exercise category record already exists on the system, the system displays an error message informing the 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).<br>- 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*

| | | |
|---|---|---|
| Author(s): George Liatos<br>Version: 1.0 | | Creation Date:20/06/2022<br>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: | This use case describes the event where the administrator would like to delete a specific exercise category record. The use case begins with the administrator requesting to delete the exercise category record. The use case concludes when the exercise category record has been deleted from the system. | |
| Pre-condition: | • The administrator must be logged into the system.<br>• The exercise category record must exist on the system. | |
| Trigger: | The administrator requests to delete an exercise category record on the system. | |
| Typical Course | Actor Action: | system Response: |
| | 1. 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:<br>- Exercise_Category_ Name<br>- Exercise_Category_ Description |
| | | 4. The system prompts the administrator to confirm the deletion of the exercise category record |
| | 5. The administrator confirms the deletion of | 6. The system deletes the exercise category record from the |

| | | |
|---|---|---|
| Alternate Courses: | the exercise category record. [ALT] | **Exercise_Category** table with the following attributes<br>- Exercise_Category_ Name<br>- Exercise_Category_ Description |
| | | 7. The system displays a success notification for the successfully deleted exercise category record. |
| | **[ALT] Step 2**: The system determines there are exercise record/s associated with the selected exercise category record. The system will notify the administrator, and ask them to ensure no exercise records are associated. The use case terminates.<br><br>**[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 exercise category record has been successfully deleted from the system. | |
| Post-condition: | The exercise category record is deleted 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).<br>- An exercise category cannot be deleted if it does not already exist on the system.<br>- 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 Specifications | None | |
| Assumptions | None | |
| Open Issues | None | |

*Table 2.1.8.15 View Exercise Category*

| | | |
|---|---|---|
| Author(s): George Liatos<br>Version: 1.0 | | Creation Date:20/06/2022<br>Last Review Date: |
| 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: | This use case describes the event where the administrator wants to access the exercise category page. The system will display the page; allowing the administrator to access the create, search, edit and delete for exercise category. . | |
| Pre-condition: | The administrator must be logged into the system. | |
| Trigger: | The administrator requests to view the exercise category page. | |
| Typical Course | Actor Action: | system Response: |
| | 1. The administrator requests to view the exercise category page. | 2. The system will display the populated list on the screen with the:<br>- Exercise_Category_Name<br>- Exercise_Category_Description<br>from the retrieved **Exercise_Category** records. |
| | 3. The administrator requests to search an exercise category record. [ALT] | 4. The system invokes Use Case 8.8 “Search exercise category”. |
| Alternate Courses: | **[ALT] Step 3a:** The administrator requests to create an exercise category record.<br>The system invokes Use Case 8.7 “Create exercise category”.<br>**[ALT] Step 3b:** The administrator requests to update an exercise category record.<br>The system invokes Use Case 8.9 “Update exercise category”.<br>**[ALT] Step 3c:** The administrator requests to delete an exercise category record.<br>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 | |

### 2.1.9 Inventory Subsystem

*Table 2.1.9.4 Create Write Off Reason*

| Author(s): Joshua Bester<br>Version: 1.0 | | Creation Date:20/06/2022<br>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 | |
| Description: | This use case describes the event where the administrator creates a new write-off reason on the system. The use case begins when the administrator wants to create a new write-off 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 write-off reason record has successfully been created on the system. | |
| Pre-condition: | • The administrator must be logged into the system | |
| Trigger: | The administrator requests to create a new write-off reason on the system | |
| Typical Course | Actor Action: | system Response: |
| | 1. The administrator requests to create a new write-off reason record on the system | 2. The system prompts the administrator to enter the following details concerning the write-off reason from the **Write_Off_Reason** table<br>- Write_Off_Reason_ Description |
| | 3. The administrator enters the required information in the field provided | 4. The system validates that the Description input field is not left empty.<br>[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<br>[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<br>- Write_Off_Reason_ Description<br>Does not match any existing write-off reason record<br>[ALT] |

| | | |
|---|---|---|
| | | 8. The system will add the new write-off reason record to the **Write_Off_Reason** table with the following attributes<br>- Write_Off_Reason_ID(the value of which is the previous Write_Off_Reason_ID incremented by 1)<br>- 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.<br>**[ALT] Step 6**: The administrator denies the information entered is correct. Return to step 3<br>**[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*

| | | |
|---|---|---|
| Author(s): Joshua Bester<br>Version: 1.0 | | Creation Date:20/06/2022<br>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: | This use case describes the event where the administrator searches for a write-off reason on the system. The use case begins when the administrator selects the option to search for a write-off reason. The use case concludes when the information is displayed for that specific write-off reason. | |
| Pre-condition: | • The administrator must be logged into the system.<br>• The write-off reason must exist on the system. | |
| Trigger: | The administrator requests to search a write-off reason on the system | |
| Typical Course | Actor Action: | system Response: |
| | 1. The administrator requests to search for a write-off reason on the system. | 2. The system prompts the administrator to enter the following details concerning the write-off reason from the **Write_Off_Reason** table:<br>- Write_Off_Reason_ Description. |
| | 3. 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_Description from the **Write_Off_Reason** table that matches the search criteria entered. [ALT] |
| | 5. 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:<br>- Write_Off_Reason_ Description |
| Alternate Courses: | **[ALT] Step 4:** No results were found. 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 system will be able to search a write-off reason (an administrator). | |

| | |
|---|---|
| Implementation Constraints and Specifications | None |
| Assumptions | None |
| Open Issues | None |

*Table 2.1.9.6 Update Write Off Reason*

| Author(s): Joshua Bester<br>Version: 1.0 | | Creation Date:20/06/2022<br>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 logged into the system.<br>• The write-off reason must exist on the system. | |
| Trigger: | The administrator requests to update a write-off reason on the system. | |
| Typical Course | Actor Action: | system Response: |
| | 1. The administrator requests to update a write-off reason on the system. | 2. The system retrieves and displays the following information regarding the write-off reason record from the **Write_Off_Reason** table:<br>- 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. |
| | 4. The administrator will update the relevant information of the write-off reason and select the option to save the changes made. | 5. 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 write-off reason record. [ALT] | 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:<br>- Refind_Reason_Description<br>Do not match any existing write-off reason record.<br>[ALT] |
| | | 9. The system saves the relevant changes to the Write_Off_Reason_Description 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.<br><br>**[ALT] Step 7**: The administrator denies the updates to be made. The use case terminates<br><br>**[ALT] Step 8**:The entered attributes match existing write-off reason records.<br>The system displays a notification informing the administrator. Return to step | |
| 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<br>Version: 1.0 | | Creation Date:20/06/2022<br>Last Review Date: |
| Use Case Name: | Delete Write-Off Reason | Use case type |
| Use Case ID: | 9.7 | 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 | |
| Description: | 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 logged into the system.<br>• The write-off reason record must exist on the system. | |
| Trigger: | The administrator requests to delete a write-off reason record on the system. | |
| Typical Course | Actor Action: | system Response: |
| | 1. The administrator requests to delete a write-off reason record on the system. | 2. 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] |
| | | 3. The system will retrieve and display the:<br>- Write_Off_Reason_ Description<br>from the **Write_Off_Reason** table |
| | | 4. 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_Description. |
| | | 7. The system notifies the administrator that the write-off reason record has been successfully deleted. |
| Alternate Courses: | **[ALT] Step 2**: The system determines there are write-off line record/s associated with the selected write-off reason record. The system will | |

| | |
|---|---|
| | notify the administrator, and ask them to ensure no write-off line records are associated before deletion. The use case terminates.<br><br>**[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 |

*Table 2.1.9.8 ViewWrite Off Reason*

| | | |
|---|---|---|
| Author(s): Joshua Bester<br>Version: 1.0 | | Creation Date:20/06/2022<br>Last Review Date: |
| Use Case Name: | View Write-Off Reason | Use case type |
| Use Case ID: | 9.8 | 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: | This use case describes the event where the administrator wants to access the write-off reason page. The system will display the page; allowing the administrator to access 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: |
| | 1. The administrator requests to view the write-off reason page. | 2. The system will display the populated table on the screen with the:<br>- Write_Off_Reason_ Description<br>from the retrieved **Write_Off_Reason** records |
| | 3. The administrator requests to search a write-off reason record. | 4. The system invokes Use Case 9.5 “Search Write-Off Reason”. |
| Alternate Courses: | **[ALT] Step 3a:** The administrator requests to create a write-off reason record.<br>The system invokes Use Case 9.4 “Create Write-Off Reason”.<br><br>**[ALT] Step 3b:** The administrator requests to update write-off reason records.<br>The system invokes Use Case 9.6 “Update Write-Off Reason”.<br><br>**[ALT] Step 3c:** The administrator requests to delete write-off reason records.<br>The system invokes Use Case 9.7 “Delete Write-Off Reason”. | |
| 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 | |

*Table 2.1.9.14 Receive Stock*

| | | |
|---|---|---|
| Author(s): Joshua Bester<br>Version: 1.0 | | Creation Date:20/06/2022<br>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.