** `**

Del 04: Technical Specifications

The purpose of this deliverable as the technical specification is to describe in detail how the intended capabilities of our system and interaction with its users will be performed. It builds on the functional specification of the preceding deliverables, and expands it to a more detailed and technical level.

Table of Contents

# Document Introduction

# Deliverable Four of our INF 370 project is the technical specification of our system. The technical specification document specifies how the functions, as specified in Deliverable 2, will be performed. In this deliverable, we will discuss the use cases, the technical use case narratives, the process models of the proposed system (functional decomposition to primitive level), system database design (entity relationship model and size estimation of proposed database with 5-year projection), system interface designs including input designs, output designs and other interfaces. The test specifications of the test plan, test scenarios and procedures, as well as test data are included. Hardware and software requirements, network/web Layout specifications, and validation of the technical specification against the requirements. The document is concluded with a client and team sign-off, and the relevant attached appendices.

# Use Cases

* 1. Introduction

##### Introduction

This section contains the use case diagrams for every requirement in the system, and their relevant subsystems. It also includes the technical use case narratives for each.

1.2 Use Case Diagrams

1.2.1 Admin Subsystem



##### 1.2.2 Employee Subsystem



##### 1.2.3 Client Subsystem



##### 1.2.4 User Subsystem



##### 1.2.5 Report Subsystem



##### 1.2.6 Exercise Subsystem Part A (6.1 -6.10)



##### 1.2.6 Exercise Subsystem Part B (6.11 -6.24)



1.3 Use Case Narratives

##### 1.3.1 System narratives

1.3.1.1 Admin Subsystem

1.3.1.1.1 Add new Subscription Details

1.3.1.1.2 Maintain Subscription Details

1.3.1.1.3 Search Subscription Details

1.3.1.1.4 Add new challenge

1.3.1.1.5 Maintain Challenge

1.3.1.1.6 Search challenge

1.3.1.1.7 Add new booking slots

1.3.1.1.8 Maintain booking slots

1.3.1.1.9 Search booking slots

1.3.1.1.10 Maintain company

1.3.1.1.11 Search booking

1.3.1.1.12 Search Subscription

1.3.1.1.13 Confirm payment

1.3.1.1.14 Search Sale

1.3.1.1.15 Conduct Consultation

1.3.1.1.16 Search Weight

1.3.1.1.17 Search Goal

1.3.1.2 Employee Subsystem

1.3.1.2.1 Add new Employee

|  |  |  |  |
| --- | --- | --- | --- |
| **JESSTER FIT** | | | |
| Author (s): Joshua Eales | Version 2.0 | | Date: 2019/07/23 |
| **USE CASE NAME:** | Add new employee | | **Use Case Type:** |
| **USE CASE ID:** | 2.1 | | Business Requirements: |
| **PRIORITY:** | High | | System Analysis: |
| **SOURCE:** | JessterFIT | | System Design:  x |
| **PRIMARY BUSINESS ACTOR** | Owner | | |
| **PRIMARY SYSTEM ACTOR** | None | | |
| **OTHER PARTICIPATING ACTORS:** | None | | |
| **OTHER INTERESTED STAKEHOLDERS:** | None | | |
| **DESCRIPTION:** | This use case describes the event where the owner wants to add new employee to the system. The owner must be logged into the system and select the option to a add a new employee. The owner will enter all details relating to the new employee. The use case ends when the new employee is added to the system and a success message is displayed to the owner. | | |
| **PRE-CONDITION:** | The owner must be logged into the system and have the right access levels and functionality to be able to add an employee to the system. | | |
| **TRIGGER:** | The owner wants to add a new employee to the system. | | |
| **TYPICAL COURSE** | Actor Action | System Response | |
| **OF EVENTS:** | Manual Action | Automated Action |
|  | **Step 1:** The owner requests to add a new employee to the system by clicking the Employee button on the navigation bar. |  | **Step 2**: The system displays the employee page with the following controls:  **Search Bar:**  **Labels:**   * Employee ID * Employee Name * Employee Surname * Employee Email * Employee Address * Employee Contact Number * Employee ID Number   **Heading:**   * Employee   **Text inputs:**   * Employee ID * Employee Name * Employee Surname * Employee Email * Employee Address * Employee Contact Number * Employee ID Number   **Buttons:**   * Add Employee * Edit * Delete   **Table**:   * Table Headings * Employee ID (auto generated) * Employee Name * Employee Surname * Employee Email * Employee Address * Employee Contact Number * Employee ID Number |
|  | **Step 3:** Owner enters new employee details into   * Employee ID (auto generated) * Employee Name * Employee Surname * Employee Email * Employee Address * Employee Contact Number * Employee ID Number   input fields. |  |  |
|  | **Step 4:** Owner clicks add employee button. |  |  |
|  |  |  | **Step 5**: The system validates entered employee details. |
|  |  |  | **Step 6**: The system checks that the employee does not already exist on the system by checking the Employee table for same Employee ID Number. |
|  |  |  | **Step 7:** The system saves the details in the Employee table. |
|  |  |  | **Step 8:** The system displays a message stating that a new employee has been successfully added onto the system |
| **ALTERNATE COURSES:** | **ALT Step 5**: The entered details are invalid or incomplete, display error message with controls:  **Text:**   * Employee details entered are either incorrect or invalid.   **Button:**   * Close   Return to Step 3. | | |
|  | **ALT Step 6**: The entered employee details already exist on the system, display error message with controls:  **Text:**   * Details already exist on the system.   **Button:**   * Close   Terminate Use case | | |
| **CONCLUSION:** | The use case ends when the new employee has been successfully added to the system and a success message is displayed for the owner. | | |
| **POST-CONDITION:** | A new employee has been successfully added to the system. | | |
| **BUSINESS RULES** | None | | |
| **IMPLEMENTATION CONTRAINTS AND SPECIFICATIONS** | None | | |
| **ASSUMPTIONS:** | None | | |
| **OPEN ISSUES:** | None | | |

1.3.1.2.2 Maintain Employee

|  |  |  |  |
| --- | --- | --- | --- |
| **JESSTER FIT** | | | |
| Author (s): Joshua Eales | Version 2.0 | | Date: 2019/07/23 |
| **USE CASE NAME:** | Maintain Employee | | **Use Case Type:** |
| **USE CASE ID:** | 2.2 | | Business Requirements: |
| **PRIORITY:** | High | | System Analysis: |
| **SOURCE:** | JessterFIT | | System Design:  x |
| **PRIMARY BUSINESS ACTOR** | Owner | | |
| **PRIMARY SYSTEM ACTOR** | None | | |
| **OTHER PARTICIPATING ACTORS:** | None | | |
| **OTHER INTERESTED STAKEHOLDERS:** | None | | |
| **DESCRIPTION:** | The Use Case describes the event that an owner would like to maintain an employee on the system. The owner will request to maintain an employee and the system will display the maintain employee page. The system will then prompt the user to search for an existing employee on the employee page and invoke use case 4.3 **Search Employee** and display the output on the page. The owner will then select the employee they would like to maintain by selecting either the edit or delete option. All changes will be saved on the system and a success message will be displayed. | | |
| **PRE-CONDITION:** | The owner must be logged into the system and have the right access levels and functionality to be able to maintain an employee. | | |
| **TRIGGER:** | The owner selects to edit an employee from the edit option. | | |
| **TYPICAL COURSE** | Actor Action | System Response | |
| **OF EVENTS:** | Manual Action | Automated Action |
|  | **Step 1**: The owner requests to maintain a employees’ details by clicking the employee button on the navigation bar. |  | **Step 2**: The system displays the employee page with the following controls:  **Search Bar:**  **Labels:**   * Employee ID * Employee Name * Employee Surname * Employee Email * Employee Address * Employee Contact Number * Employee ID Number   **Heading:**   * Employee   **Text inputs:**   * Employee ID * Employee Name * Employee Surname * Employee Email * Employee Address * Employee Contact Number * Employee ID Number   **Buttons:**   * Add Employee * Edit * Delete   **Table**:   * Table Headings * Employee ID (auto generated) * Employee Name * Employee Surname * Employee Email * Employee Address * Employee Contact Number * Employee ID Number |
|  |  |  | **Step 3:** The system will invoke use case 4.3 Search Employee. |
|  | **Step 4:** The owner will select the edit employee option for the employee that they would like to edit by clicking the edit button. |  |  |
|  |  |  | **Step 5:** The system populates the text fields with the selected employee details.  **Employee ID (non- changeable)**  And   * Employee Name * Employee Surname * Employee Email * Employee Address * Employee Contact Number * Employee ID Number   **(All changeable)** |
|  | **Step 6:** The owner will update the Employee Name, Surname, Email Address, Contact Number and ID Number by typing into the text fields provided. |  |  |
|  | **Step 7:** Owner clicks the submit button to save changes. |  |  |
|  |  |  | **Step 8:** The system will validate the employee to ensure that the details are valid employee details. |
|  |  |  | **Step 9:** The system will then update the employee for the selected **Employee\_ID.** |
|  |  |  | **Step 10:** The system will display a success message to the owner alerting them of successful changes. |
| **ALTERNATE COURSES:** | **Alt Step 4a:** The owner selects the delete option for the employee that they would like to remove by clicking the delete button.  **Alt Step 4b:** The system prompts the user to confirm if they would like to delete the selected employee by displaying a page with the following controls:  **Text:**   * Are you sure you wish to delete this record?   **Buttons:**   * Submit * Cancel   **Alt Step 4c:** The user confirms that they would like to delete the selected employee by clicking the confirm button.  **Alt Step 4cc:** The User revokes the decision to remove the selected employee by clicking the cancel button.  Terminate Use Case.  **Alt Step 4d:** The system removes the selected employee and its relevant details from the Employee **table.** Display success message with controls:  **Text:**   * Selected employee record has been removed successfully   **Buttons:**   * Close | | |
|  | **Alt Step 8:** The updated employee details are invalid, display error message with controls:  **Text:**   * Details entered are invalid. Please revise them to ensure they are correct.   **Button:**   * Close   Return to step 6. | | |
| **CONCLUSION:** | The selected employee has been successfully maintained and all changes have been saved on the system and a success message has been displayed. | | |
| **POST-CONDITION:** | Employee details have been successfully maintained. | | |
| **BUSINESS RULES** | Only a manager or an owner can maintain an employee. | | |
| **IMPLEMENTATION CONTRAINTS AND SPECIFICATIONS** | None | | |
| **ASSUMPTIONS:** | None | | |
| **OPEN ISSUES:** | None | | |

1.3.1.2.3 Search Employee

|  |  |  |  |
| --- | --- | --- | --- |
| **JESSTER FIT** | | | |
| Author (s): Joshua Eales | Version 2.0 | | Date: 2019/07/23 |
| **USE CASE NAME:** | Search Employee | | **Use Case Type:** |
| **USE CASE ID:** | 2.3 | | Business Requirements: |
| **PRIORITY:** | High | | System Analysis: |
| **SOURCE:** | JessterFIT | | System Design:  x |
| **PRIMARY BUSINESS ACTOR** | Owner | | |
| **PRIMARY SYSTEM ACTOR** | None | | |
| **OTHER PARTICIPATING ACTORS:** | None | | |
| **OTHER INTERESTED STAKEHOLDERS:** | None | | |
| **DESCRIPTION:** | This use case narrative describes an event in which an owner wants to search for an employee on the system. The system will request the owner to enter any keyword or phrase matching the existing employee they want to return on the system. The use case will end when the employee details searched are returned to the owner. | | |
| **PRE-CONDITION:** | The owner must be logged into the system and have the right access levels and functionality to be able to search an employee on the system. | | |
| **TRIGGER:** | The owner needs to search for an employee on the system. | | |
| **TYPICAL COURSE** | **Actor Action** | System Response | |
| **OF EVENTS:** | Manual Action | Automated Action |
|  | **Step 1**: Owner requests to search for an existing employees’ details on the system by clicking the Employee button. |  | **Step 2**: The system displays the employee page with the following controls:  **Search Bar:**  **Labels:**   * Employee ID * Employee Name * Employee Surname * Employee Email * Employee Address * Employee Contact Number * Employee ID Number   **Heading:**   * Employee   **Text inputs:**   * Employee ID * Employee Name * Employee Surname * Employee Email * Employee Address * Employee Contact Number * Employee ID Number   **Buttons:**   * Add Employee * Edit * Delete   **Table**:   * Table Headings * Employee ID (auto generated) * Employee Name * Employee Surname * Employee Email * Employee Address * Employee Contact Number * Employee ID Number |
|  | **Step 3:** User enters search criteria of the employee into the search bar:  **Employee ID**  **Or**  **Employee ID Number** |  |  |
|  | **Step 4**: Owner clicks the search button. |  |  |
|  |  |  | **Step 5**: The system displays a list of employees matching entered criteria. |
| **ALTERNATE COURSES:** | **ALT Step 5**: No records matching search criteria, display error message with controls:  **Text:**   * No employee matching search criteria.   **Button:**   * Close   Return to Step 3. | | |
| **CONCLUSION:** | The system has successfully searched for the required employee based off the given search criteria and displayed the results to the owner. | | |
| **POST-CONDITION:** | The system successfully finds the employee(s) on the system. | | |
| **BUSINESS RULES** | Only a manager or an owner can search an employee. | | |
| **IMPLEMENTATION CONTRAINTS AND SPECIFICATIONS** | None | | |
| **ASSUMPTIONS:** | None | | |
| **OPEN ISSUES:** | None | | |

1.3.1.2.4 Add new Employee Type

|  |  |  |  |
| --- | --- | --- | --- |
| **JESSTER FIT** | | | |
| Author (s): Joshua Eales | Version 2.0 | | Date: 2019/07/23 |
| **USE CASE NAME:** | Add new employee type | | **Use Case Type:** |
| **USE CASE ID:** | 2.4 | | Business Requirements: |
| **PRIORITY:** | High | | System Analysis: |
| **SOURCE:** | JessterFIT | | System Design:  x |
| **PRIMARY BUSINESS ACTOR** | Employee | | |
| **PRIMARY SYSTEM ACTOR** | None | | |
| **OTHER PARTICIPATING ACTORS:** | None | | |
| **OTHER INTERESTED STAKEHOLDERS:** | Owner | | |
| **DESCRIPTION:** | This use case describes the event where an employee wants to add new employee type to the system. The employee must be logged into the system and select the option to a add a new employee type. The employee will enter all details relating to the new employee type. The use case ends when the new employee type is added to the system and a success message is displayed to the employee. | | |
| **PRE-CONDITION:** | The employee must be logged into the system and have the right access levels and functionality to be able to add a new employee type to the system. | | |
| **TRIGGER:** | The employee wants to add a new employee type to the system. | | |
| **TYPICAL COURSE** | Actor Action | System Response | |
| **OF EVENTS:** | Manual Action | Automated Action |
|  | **Step 1:** The employee requests to add a new employee type to the system by clicking the Employee Type button on the navigation bar. |  | **Step 2**: The system displays the employee type page with the following controls:  **Search Bar:**  **Labels:**   * Employee Type ID * Employee Type Description   **Heading:**   * Employee Type   **Text inputs:**   * Employee Type ID * Employee Type Description   **Buttons:**   * Add Employee Type * Edit * Delete   **Table**:   * Table Headings * Employee Type ID (auto generated) * Employee Type Description |
|  | **Step 3:** Employee enters new employee type details into   * Employee Type ID(auto generated) * Employee Type Description   input fields. |  |  |
|  | **Step 4:** Employee clicks add employee type button. |  |  |
|  |  |  | **Step 5**: The system validates entered employee type details. |
|  |  |  | **Step 6**: The system checks that the employee type does not already exist on the system by checking the Employee Type table for same Employee Type Description. |
|  |  |  | **Step 7:** The system saves the details in the Employee Type table. |
|  |  |  | **Step 8:** The system displays a message stating that a new employee type has been successfully added onto the system |
| **ALTERNATE COURSES:** | **ALT Step 5**: The entered details are invalid or incomplete, display error message with controls:  **Text:**   * Employee type details entered are either incorrect or invalid.   **Button:**   * Close   Return to Step 3. | | |
|  | **ALT Step 6**: The entered employee type details already exist on the system, display error message with controls:  **Text:**   * Details already exist on the system.   **Button:**   * Close   Terminate Use case | | |
| **CONCLUSION:** | The use case ends when a new employee type has been successfully added to the system and a success message is displayed for the employee. | | |
| **POST-CONDITION:** | A new employee type has been successfully added to the system. | | |
| **BUSINESS RULES** | None | | |
| **IMPLEMENTATION CONTRAINTS AND SPECIFICATIONS** | None | | |
| **ASSUMPTIONS:** | None | | |
| **OPEN ISSUES:** | None | | |

1.3.1.2.5 Maintain Employee Type

|  |  |  |  |
| --- | --- | --- | --- |
| **JESSTER FIT** | | | |
| Author (s): Joshua Eales | Version 2.0 | | Date: 2019/07/23 |
| **USE CASE NAME:** | Maintain Employee Type | | **Use Case Type:** |
| **USE CASE ID:** | 2.5 | | Business Requirements: |
| **PRIORITY:** | High | | System Analysis: |
| **SOURCE:** | JessterFIT | | System Design:  x |
| **PRIMARY BUSINESS ACTOR** | Employee | | |
| **PRIMARY SYSTEM ACTOR** | None | | |
| **OTHER PARTICIPATING ACTORS:** | None | | |
| **OTHER INTERESTED STAKEHOLDERS:** | Owner | | |
| **DESCRIPTION:** | The Use Case describes the event that an employee would like to maintain an employee type on the system. The employee will request to maintain an employee type and the system will display the maintain employee type page. The system will then prompt the user to search for an existing employee type on the employee type page and invoke use case **4.6 Search Employee Type** and display the output on the page. The employee will then select the type they would like to maintain by selecting either the edit or delete option. All changes will be saved on the system and a success message will be displayed. | | |
| **PRE-CONDITION:** | The employee must be logged into the system and have the right access levels and functionality to be able to maintain an employee type. | | |
| **TRIGGER:** | The employee selects to edit an employee type from the edit employee option. | | |
| **TYPICAL COURSE** | Actor Action | System Response | |
| **OF EVENTS:** | Manual Action | Automated Action |
|  | **Step 1**: The employee requests to maintain an employee types details by clicking the employee type button on the navigation bar. |  | **Step 2**: The system displays the employee type page with the following controls:  **Search Bar:**  **Labels:**   * Employee Type ID * Employee Type Description   **Heading:**   * Employee Type   **Text inputs:**   * Employee Type ID * Employee Type Description   **Buttons:**   * Add Employee Type * Edit * Delete   **Table**:   * Table Headings * Employee Type ID (auto generated) * Employee Type Description |
|  |  |  | **Step 3:** The system will invoke use case 4.6 Search Employee Type. |
|  | **Step 4:** The employee will select the edit employee type option for the employee type that they would like to edit by clicking the edit button. |  |  |
|  |  |  | **Step 5:** The system populates the text fields with the selected employee type details.  **Employee Type ID (non- changeable)**  And  **Employee Type Description (changeable)** |
|  | **Step 6:** The employee will update the Employee Type Description by typing into the text field provided. |  |  |
|  | **Step 7:** Employee clicks the submit button to save changes. |  |  |
|  |  |  | **Step 8:** The system will validate the employee type to ensure that the details are valid employee type details. |
|  |  |  | **Step 9:** The system will then update the employee type for the selected **Employee\_Type\_ID.** |
|  |  |  | **Step 10:** The system will display a success message to the employee alerting them of successful changes. |
| **ALTERNATE COURSES:** | **Alt Step 4a:** The employee selects the delete option for the employee type that they would like to remove by clicking the delete button.  **Alt Step 4b:** The system prompts the user to confirm if they would like to delete the selected employee type by displaying a page with the following controls:  **Text:**   * Are you sure you wish to delete this record?   **Buttons:**   * Submit * Cancel   **Alt Step 4c:** The user confirms that they would like to delete the selected employee type by clicking the confirm button.  **Alt Step 4cc:** The User revokes the decision to remove the selected employee type by clicking the cancel button.  Terminate Use Case.  **Alt Step 4d:** The system removes the selected employee type and its relevant details from the Employee Type table**.** Display success message with controls:  **Text:**   * Selected employee type record has been removed successfully   **Buttons:**   * Close | | |
|  | **Alt Step 8:** The updated employee type details are invalid, display error message with controls:  **Text:**   * Details entered are invalid. Please revise them to ensure they are correct.   **Button:**   * Close   Return to step 6. | | |
| **CONCLUSION:** | The selected employee type has been successfully maintained and all changes have been saved on the system and a success message has been displayed. | | |
| **POST-CONDITION:** | Employee type details have been successfully maintained. | | |
| **BUSINESS RULES** | None. | | |
| **IMPLEMENTATION CONTRAINTS AND SPECIFICATIONS** | None | | |
| **ASSUMPTIONS:** | None | | |
| **OPEN ISSUES:** | None | | |

1.3.1.2.6 Search Employee Type

|  |  |  |  |
| --- | --- | --- | --- |
| **JESSTER FIT** | | | |
| Author (s): Joshua Eales | Version 2.0 | | Date: 2019/07/23 |
| **USE CASE NAME:** | Search Employee Type | | **Use Case Type:** |
| **USE CASE ID:** | 2.6 | | Business Requirements: |
| **PRIORITY:** | High | | System Analysis: |
| **SOURCE:** | JessterFIT | | System Design:  x |
| **PRIMARY BUSINESS ACTOR** | Employee | | |
| **PRIMARY SYSTEM ACTOR** | None | | |
| **OTHER PARTICIPATING ACTORS:** | None | | |
| **OTHER INTERESTED STAKEHOLDERS:** | Owner | | |
| **DESCRIPTION:** | This use case narrative describes an event in which an employee wants to search for an employee types’ details on the system. The system will request the employee to enter any keyword or phrase matching the existing employee type details they want to return on the system. The use case will end when the employee types’ details searched are returned to the employee. | | |
| **PRE-CONDITION:** | The employee must be logged into the system and have the right access levels and functionality to be able to search an employee type on the system. | | |
| **TRIGGER:** | The employee needs to search for an employee type on the system. | | |
| **TYPICAL COURSE** | **Actor Action** | System Response | |
| **OF EVENTS:** | Manual Action | Automated Action |
|  | **Step 1**: Employee requests to search for an existing employee types details on the system by clicking the Employee Type button. |  | **Step 2**: The system displays the employee type page with the following controls:  **Search Bar:**  **Labels:**   * Employee Type ID * Employee Type Description   **Heading:**   * Employee Type   **Text inputs:**   * Employee Type ID * Employee Type Description   **Buttons:**   * Add Employee Type * Edit * Delete   **Table**:   * Table Headings * Employee Type ID (auto generated) * Employee Type Description |
|  | **Step 3:** Employee enters search criteria of the employee type into the search bar:  **Employee Type ID**  **Or**  **Employee Type Description** |  |  |
|  | **Step 4**: Employee clicks the search button. |  |  |
|  |  |  | **Step 5**: The system displays a list of employee types matching entered criteria. |
| **ALTERNATE COURSES:** | **ALT Step 5**: No records matching search criteria, display error message with controls:  **Text:**   * No employee type matching search criteria.   **Button:**   * Close   Return to Step 3. | | |
| **CONCLUSION:** | The system has successfully searched for the required employee type based off the given search criteria and displayed the results to the owner. | | |
| **POST-CONDITION:** | The system successfully finds the employee type(s) on the system. | | |
| **BUSINESS RULES** | None | | |
| **IMPLEMENTATION CONTRAINTS AND SPECIFICATIONS** | None | | |
| **ASSUMPTIONS:** | None | | |
| **OPEN ISSUES:** | None | | |

1.3.1.3 Client Subsystem

1.3.1.3.1 Register new client

|  |  |  |  |
| --- | --- | --- | --- |
| **JESSTER FIT** | | | |
| Author (s): Joshua Eales | Version 2.0 | | Date: 2019/07/23 |
| **USE CASE NAME:** | Register New Client | | **Use Case Type:** |
| **USE CASE ID:** | 3.1 | | Business Requirements: |
| **PRIORITY:** | High | | System Analysis: |
| **SOURCE:** | JessterFIT | | System Design:  x |
| **PRIMARY BUSINESS ACTOR** | Owner | | |
| **PRIMARY SYSTEM ACTOR** | None | | |
| **OTHER PARTICIPATING ACTORS:** | None | | |
| **OTHER INTERESTED STAKEHOLDERS:** | None | | |
| **DESCRIPTION:** | This use case describes the event where the client wants to add register a client profile on the system the client will select the register option on the home page. The system will display the register client screen and client will provide all the necessary details and click register. The use case ends when the new client is added to the system and a success message is displayed to the user. | | |
| **PRE-CONDITION:** | The client not must be logged into the system and not have an existing profile. | | |
| **TRIGGER:** | The client wants to register a new profile on the system. | | |
| **TYPICAL COURSE** | Actor Action | System Response | |
| **OF EVENTS:** | Manual Action | Automated Action |
|  | **Step 1:** The client selects the ‘Register’ button on the ‘**home’** page **navigation bar**. |  | **Step 2**: The system displays the employee page with the following controls:  **Labels:**   * Client Name * Client Surname * Client Email * Contact Number   **Heading:**   * Client   **Text inputs:**   * Client Name (text) * Client Surname (text) * Client Email (text) * Contact Number (text) * Password (text)   **Buttons:**   * Register Client * Back   **Table**:  **Table Headings:**   * Client ID (auto generated) * Client Name * Client Surname * Client Email * Client Contact Number * Password |
|  | **Step 3:** Client enters new details into:   * Client Name * Client Surname * Client Email * Client Contact Number   input fields. |  |  |
|  | **Step 4:** Client clicks ‘**Register Client’**  **button.** |  | **Step 5**: The system validates entered client details. |
|  |  |  | **Step 6**: The system checks that the client does not already exist on the system by checking the Client table for same attributes provided. |
|  |  |  | **Step 7:** The system saves the details in the Client table. |
|  |  |  | **Step 8:** The system displays a message stating that a new client has been successfully added onto the system. |
| **ALTERNATE COURSES:** | **ALT Step 5**: The entered details are invalid or incomplete, display error message with controls:  **Text:**   * Client details entered are either incorrect or invalid.   **Button:**   * **Close** * Return to Step 3. | | |
|  | **ALT Step 6**: The entered client details already exist on the system, display error message with controls:  **Text:**   * Details already exist on the system.   **Button:**   * Close   Terminate Use case | | |
| **CONCLUSION:** | The use case ends when the new client has been successfully added to the system and a success message is displayed for the user. | | |
| **POST-CONDITION:** | A new client has been successfully added to the system. | | |
| **BUSINESS RULES** | None | | |
| **IMPLEMENTATION CONTRAINTS AND SPECIFICATIONS** | None | | |
| **ASSUMPTIONS:** | None | | |
| **OPEN ISSUES:** | None | | |

1.3.1.3.2 Maintain Client Profile

|  |  |  |  |
| --- | --- | --- | --- |
| **JESSTER FIT** | | | |
| Author (s): Joshua Eales | Version 2.0 | | Date: 2019/07/23 |
| **USE CASE NAME:** | Maintain Client | | **Use Case Type:** |
| **USE CASE ID:** | 3.2 | | Business Requirements: |
| **PRIORITY:** | High | | System Analysis: |
| **SOURCE:** | JessterFIT | | System Design:  x |
| **PRIMARY BUSINESS ACTOR** | Owner | | |
| **PRIMARY SYSTEM ACTOR** | None | | |
| **OTHER PARTICIPATING ACTORS:** | None | | |
| **OTHER INTERESTED STAKEHOLDERS:** | None | | |
| **DESCRIPTION:** | The Use Case describes the event that a client would like to maintain their details on the system. The client will request to maintain an client and the system will display the maintain client page. The system will then prompt the user to search for an existing client on the client page and invoke use case 3.3 **Search Client** and display the output on the page. The client will edit the details that they are allowed to edit. | | |
| **PRE-CONDITION:** | The client must be logged into the system and have the right access levels and functionality to be able to maintain a client. | | |
| **TRIGGER:** | The client requests to edit their details on the Client Profile Page. | | |
| **TYPICAL COURSE** | Actor Action | System Response | |
| **OF EVENTS:** | Manual Action | Automated Action |
|  | **Step 1**: The client requests to view their profile. |  | **Step 2:** The system invokes **3.17 View Profile.** |
|  | **Step 3**: The client selects the ‘**edit’** profile option. |  | **Step 4**: The system displays the client page with the following controls:  **Labels:**   * Client ID * Client Name * Client Surname * Client Email * Client Contact Number   **Text inputs:**   * Client ID (Disabled) * Client Name (Disabled) * Client Surname (Disabled) * Client Email (Disabled) * Client Contact Number (Disabled)   **Buttons:**   * Edit * Back |
|  | **Step 5:** The client will select the **‘edit’ button.** |  | **Step 6:** The system will enable the following **textboxes**:   * **Client Email (text)** * **Client Contact Number (text)** |
|  | . |  | **Step 7:** The system will display the following controls  Labels:   * Please edit your contact number and email address. * Due to security reasons you cannot edit Name and Surname.   **Buttons:**   * **Update Details** * **Back** |
|  | **Step 8:** The client will edit the necessary details |  |  |
|  | **Step 9:** The client selects the ‘**Update Details’** button. |  | **Step 10:** The system will validate the client details provided to ensure that the details are valid client details. |
|  |  |  | **Step 11:** The system will then update the details in the **Client** table |
|  |  |  | **Step 12:** The system will display a success message to the user alerting them of successful changes. |
| **ALTERNATE COURSES:** | **Alt Step 5:** The client selects the **‘Back’ button** and the system returns the user to the ‘**home’** page. | | |
|  | **Alt Step 9:** The client selects the **‘Back’ button** and the system returns the user to the ‘**View Profile’** page. | | |
|  | **Alt Step 10:** The updated employee details are invalid, display error message with controls:  **Text:**   * Details entered are invalid. Please revise them to ensure they are correct.   **Button:**   * Close   Return to step 8. | | |
| **CONCLUSION:** | The selected client has been successfully maintained and all changes have been saved on the system and a success message has been displayed. | | |
| **POST-CONDITION:** | Client details have been updated in the **Client** table. | | |
| **BUSINESS RULES** | Only a client can edit their own details. | | |
| **IMPLEMENTATION CONTRAINTS AND SPECIFICATIONS** | None | | |
| **ASSUMPTIONS:** | None | | |
| **OPEN ISSUES:** | None | | |

1.3.1.3.3 Search Client

|  |  |  |  |
| --- | --- | --- | --- |
| **JESSTER FIT** | | | |
| Author (s): Joshua Eales | Version 2.0 | | Date: 2019/07/23 |
| **USE CASE NAME:** | Search Client | | **Use Case Type:** |
| **USE CASE ID:** | 3.3 | | Business Requirements: |
| **PRIORITY:** | High | | System Analysis: |
| **SOURCE:** | JessterFIT | | System Design:  x |
| **PRIMARY BUSINESS ACTOR** | Client | | |
| **PRIMARY SYSTEM ACTOR** | None | | |
| **OTHER PARTICIPATING ACTORS:** | None | | |
| **OTHER INTERESTED STAKEHOLDERS:** | None | | |
| **DESCRIPTION:** | This use case narrative describes an event in which a client needs to search for their client profile on the system. The system will request the client to enter any keyword or phrase matching the existing client details they want to return on the system. The use case will end when the employee details searched are returned to the client. | | |
| **PRE-CONDITION:** | The client must be logged into the system and have the right access levels and functionality to be able to search a client on the system. | | |
| **TRIGGER:** | The client needs to search for an employee on the system. | | |
| **TYPICAL COURSE** | **Actor Action** | System Response | |
| **OF EVENTS:** | Manual Action | Automated Action |
|  | **Step 1**: client requests to search for an existing client details on the system by clicking the client button. |  | **Step 2**: The system displays the client page with the following controls:  **Search Bar:**  **Labels:**   * Client ID * Client Name * Client Surname * Cleint Email * Client Contact Number   **Text inputs:**   * Client ID (Disabled) * Client Name (Disabled) * Client Surname (Disabled) * Client Email (Disabled) * Client Contact Number (Disabled) |
|  | **Step 3:** User enters search criteria of the employee into the search bar:  **Client ID**  **Or**  **Client Name** |  |  |
|  | **Step 4**: Client clicks the search button. |  |  |
|  |  |  | **Step 5**: The system displays a list of client matching entered criteria. |
| **ALTERNATE COURSES:** | **ALT Step 5**: No records matching search criteria, display error message with controls:  **Text:**   * No client matching search criteria.   **Button:**   * Close   Return to Step 3. | | |
| **CONCLUSION:** | The system has successfully searched for the required client based off the given search criteria and displayed the results to the client. | | |
| **POST-CONDITION:** | The system successfully finds the client(s) on the system. | | |
| **BUSINESS RULES** | Only a client or an owner can search an client. | | |
| **IMPLEMENTATION CONTRAINTS AND SPECIFICATIONS** | None | | |
| **ASSUMPTIONS:** | None | | |
| **OPEN ISSUES:** | None | | |

1.3.1.3.4 Sign Up for subscription

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **JESSTER FIT** | | | | | | |
| Author (s): | | Version 2.0 | | | Date: 2019/07/23 | |
| **USE CASE NAME:** | **Sign Up for Subscription** | | | | | **Use Case Type:** |
| **USE CASE ID:** | 3.4 | | | | | Business Requirements: 🞎 |
| **PRIORITY:** | High | | | | | System Analysis: 🞎 |
| **SOURCE:** | JessterFIT | | | | | System Design: x |
| **PRIMARY BUSINESS ACTOR** | Client | | | | | |
| **PRIMARY SYSTEM ACTOR** | None. | | | | | |
| **OTHER PARTICIPATING ACTORS:** | None. | | | | | |
| **OTHER INTERESTED STAKEHOLDERS:** | None. | | | | | |
| **DESCRIPTION:** | The Use Case describes the event that a client would like to sign up for a subscription. The client will select to sign up and either select a 6-months, 1-year, 2-year or 3-year subscription. Once the client has selected the length of their subscription, they will add it to the basket and be informed that they have to pay for the basket before they logout otherwise it will be removed. Once confirmed by the owner the subscription will begin. | | | | | |
| **PRE-CONDITION:** | The client must be logged into the system and must not yet be registered for a subscription to be able to sign up for a subscription. | | | | | |
| **TRIGGER:** | The client wants to sign up for a subscription. | | | | | |
| **TYPICAL COURSE** | Actor Action | | System Response | | | |
| **OF EVENTS:** | Manual Action | Automated Action | | |
|  | **Step 1**: The client selects to sign up for a subscriptionon the home page. | |  | **Step 2:** The system will retrieve the following attributes from the **Subscription\_Detail** table:   * **Subscription\_Detail\_ID** * **Price\_Per\_Month**   Using **LINQ queries** and **Entity Framework.** | | |
|  |  | |  | **Step 3:** The system will then display the **‘Subscription Sign Up’** pop up with the following controls:  **Labels**:   * **Subscribe to JessterFIT** * **Why Subscribe?** * **A subscription will get you the best out of the JessterFIT experience, upon payment confirmation you will be able to book a private personal training session with one of our Personal Trainers who bn t** * **Stannard Subscription price per month**   **Table:**  **Column Headings:**   * **Subscription Type** * **Price** * **Price Per Month** * **Discount**   **First Column:**   * **6 Month Subscription Price** * **1 Year Subscription Price** * **2 Year Subscription Price** * **3 Year Subscription Price**   **Second Column:**   * The system will take the **Price\_Per\_Month** attribute and multiply it by 6 to calculate the **six-month price** of a subscription. * The system will take the **Price\_Per\_Month** attribute and multiply it by 12 and then multiply it again by 0.9 to calculate the **one-year price** of a subscription and to give the 10% discount for one year. * The system will take the **Price\_Per\_Month** attribute and multiply it by 24 and then multiply it again by 0.8 to calculate the **two-year price** of a subscription and to give the 20% discount for two years. * The system will take the **Price\_Per\_Month** attribute and multiply it by 36 and then multiply it again by 0.7 to calculate the price for a three-year subscription and give the 30% discount for three years.   **Third Column:**   * **Price\_Per\_Month** * The system divides the **one-year price** by 12 to get the **price per month** for a year subscription. * The system divides the **two-year price** by 24 to get the **price per month** for a two-year subscription. * The system divides the **three-year price** by 36 to get the **price per month** for a three-year subscription.   **Fourth Column:**   * **No Discount** * **10%** * **20%** * **30%**   **Buttons**:   * **Subscribe Now** * **Cancel** | | |
|  | **Step 4:** The client selects the ‘**Subscribe Now**’ **button**. | |  | **Step 5:** The system will hide the two buttons mentioned in step 4 and display the following controls on the **‘Subscription Sign Up’** pop up:  **Labels:**   * **Select the 6 Month subscription for our standard subscription.** * **Do you want to get a discount?** * **Select one of our yearly subscriptions to get a discount.** * **One year for a 10% discount, two years for a 20% discount and 3 years for a 30% discount.**   **Radio Groups:**   * **6 Months** * **1 Year** * **2 Years** * **3 Years**   **Buttons:**   * **Subscribe!** * **Cancel** | | |
|  | **Step 6:** The client selects one of the available **radio buttons** for the length of time they would like to subscribe for. | |  | **Step 7:** The system will retrieve the current date**.** | | |
|  | **Step 8:** The Client selects the ‘**Subscribe!’ button.** | |  | **Step 9:** The system will display the **‘Confirm Subscription’ message box** with the following controls:  **Labels:**   * **Please confirm the chosen length of your subscription and the price.** * **Length of subscription:** * **Length of the subscription (the selected radio button text)** * **Price to Pay:** * **Total subscription price calculated in step 3** * **Price per month calculated in step 3**   **Buttons:**   * **Confirm** * **Cancel** | | |
|  | **Step 10:** The Client selects the ‘**Confirm’ button.** | |  | **Step 11:** The system closes the **‘Confirm Subscription Length’ message box** and will display the following controls to the user on **‘Subscription Sign Up’** pop up:  **Labels:**   * **Please be aware that you will have to pay a once off payment.** * **Once you have made the payment the owner will confirm the payment and once confirmed your subscription will begin.** * **Please click to subscribe to confirm your subscription and make your payment to begin the true JessterFIT experience.** * **Please be aware that once a subscription has been confirmed it cannot be canceled.**   **Buttons:**   * **Add to Basket** * **Cancel**   The system will prompt the user to select the **‘Pay Now’ button**. | | |
|  | **Step 12:** The client selects the ‘**Add to Basket’**  **button.** | |  | **Step 13:** The system retrieves the **Client\_ID** of the logged in client from the **Client** table using **LINQ queries** and **Entity Framework.** | | |
|  |  | |  | **Step 14:** The system will use the **Client\_ID** andinvoke **use case** **3.15 Create New Client Bioform** to gather the necessary details so the owner can create a custom exercise plan. | | |
|  |  | |  | **Step 15:** The system will use the **radio button** selected by the client to calculate the **length in months** of the selected subscription. | | |
|  |  | |  | **Step 16:** The system will search the **Subsription**\_**Length** table for the **Length attribute** that matches the calculated **length in months** and return the **Length\_ID** attributeof the **Length** attribute that matches **length in months.**  This will be done by the system using **LINQ queries** and **Entity Framework.** | | |
|  |  | |  | **Step 17:** The system will create a new entry in the **Subscription** table by getting the most recent **Subscription\_ID** and adding 1 to the **Subscription\_ID** and then adding it to **Subscription** the table. The system will then add the following details to the row:   * The current date as **Subscription\_Start\_Date** * **Calculated price (calculated in step 3)** as **Subscription\_Amount.** * The system will also get the **Subscription\_Status\_ID** from the **Subscription\_Status** table where the **Sub\_Status\_Description** is **‘Unconfirmed’** and save the retrieved ID in the **Subscription** table as a foreign key linking it to the **Subscription\_Status** table. * The retrieved **Length\_ID** as a **Length\_ID** foreign keylinking it to the **Subscription\_Length** table.   This will all be done using **LINQ queries** and **Entity Framework.** | | |
|  |  | |  | **Step 18:** The system will link the subscription to the client by adding the created **Subscription\_ID** to the **Client** table. This will all be done using **LINQ queries** and **Entity Framework.** | | |
|  |  | |  | **Step 19:** The system will display the ‘**Successful Subscription Sign Up’** **message** **box** with the following controls:  **Labels:**   * **Subscription sign up success!** * Once you have made your payment and it has been confirmed your personalized gym plan will be created and emailed to you. * You will also be able to make bookings with any one of our personal trainers at a slot and venue that suits you. * After every monthly payment you will be given the option to create a new custom exercise plan that will suit you and your body changes, please feel free to make use of our services.   **Button:**   * **OK** | | |
|  | **Step 20:** The client selects the ‘**OK’ button.** | |  | **Step 21:** The system will return the client to the **Home** page. | | |
| **ALTERNATE COURSES:** | **Alt Step 4:**  The client selects the ‘**Cancel’ button.** The system closes the **‘Subscription Sign Up’** pop up and returns the client to the ‘**Home’** page. | | | | | |
|  | **Alt Step 8:** The client selects the ‘**Cancel’ button.** The system closes the **‘Subscription Sign Up’** pop up and returns the client to the ‘**Home’** page. | | | | | |
|  | **Alt Step 10:** The client selects the ‘**Cancel’ button.** The system closes the **‘Confirm Subscription Length’ message box** and returns the client to the **‘Subscription Sign Up’** pop up. | | | | | |
|  | **Alt Step 12:** The client selects the ‘**Cancel’ button.** The system closes the **‘Subscription Sign Up’** pop up and returns the client to the ‘**Home’** page. | | | | | |
| **CONCLUSION:** | The use case ends when the client has successfully signed up for a subscription and all the subscription details have been successfully saved to the **Subscription** table. | | | | | |
| **POST-CONDITION:** | The subscription status is ‘**Unconfirmed’**. | | | | | |
| **BUSINESS RULES** | * A client can only have one subscription at a time. * A client cannot cancel a subscription once it has been confirmed. | | | | | |
| **IMPLEMENTATION CONTRAINTS AND SPECIFICATIONS** | None. | | | | | |
| **ASSUMPTIONS:** | The client exists on the system and has not yet registered for the subscription | | | | | |
| **OPEN ISSUES:** | None. | | | | | |

1.3.1.3.5 Register for a challenge

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **JESSTER FIT** | | | | | | |
| Author (s): Iain Steyn | | Version 2.0 | | | Date: 2019/07/23 | |
| **USE CASE NAME:** | **Register for a Challenge** | | | | | **Use Case Type:** |
| **USE CASE ID:** | 3.5 | | | | | Business Requirements: 🞎 |
| **PRIORITY:** | High | | | | | System Analysis: 🞎 |
| **SOURCE:** | JessterFIT | | | | | System Design: x |
| **PRIMARY BUSINESS ACTOR** | Client | | | | | |
| **PRIMARY SYSTEM ACTOR** | None. | | | | | |
| **OTHER PARTICIPATING ACTORS:** | None. | | | | | |
| **OTHER INTERESTED STAKEHOLDERS:** | None. | | | | | |
| **DESCRIPTION:** | The Use Case describes the event that a client would like to register for a challenge on the system. The client will select an upcoming challenge to register for, confirm the registration and then will be prompted to pay for the challenge in their basket. | | | | | |
| **PRE-CONDITION:** | The client must be logged into the system and must not yet be registered for the current challenge. | | | | | |
| **TRIGGER:** | The client wants to register for a current challenge. | | | | | |
| **TYPICAL COURSE** | Actor Action | | System Response | | | |
| **OF EVENTS:** | Manual Action | Automated Action | | |
|  | **Step 1**: The client selects the ‘**Challenge’** option on the ‘**Home’** page **navbar**. | |  | **Step 2:** The system will retrieve the following attributes from the **Challenge** table:   * **Challenge\_ID** * **Challenge\_Detail\_ID** * **Prize\_ID** * **Challenge\_Start\_Date** * **Challenge\_End\_Date**   The following details are retrieved from the **Challenge\_Detail** table using the **Challenge\_Detail\_ID:**   * **Challenge\_Description** * **Challenge\_Price** * **Challenge\_Date\_Edited**   The following details are retrieved from the **Challenge\_Prize** table using the **Prize\_ID:**   * **Prize\_Description**   Using **LINQ queries** and **Entity Framework.** The system will also retrieve the current date. | | |
|  |  | |  | **Step 3:** The system will check if the current date is before the **Challenge\_Start\_Date.** To check if registration is still open for the current challenge. | | |
|  |  | |  | **Step 4:** The system will display the ‘**Challenge’** pop up with the following controls  **Labels:**   * **ChallengeDesc –** populated by the **Challenge\_Description** attribute**.** * **ChallengePrice –** populated by the **Challenge\_Price** attribute. * **ChallengeStart** – populated by the **Challenge\_Start\_Date** attribute. * **ChallengeEnd** – populated by the **Challenge\_End\_Date** attribute. * **ChallengePrize** – populated by the **Prize\_Description** attribute. * **DaysRemaining** – this label is equal the **Challenge\_Start\_Date** day subtracted by the **TmpCurentDate** day.   **Button:**   * **Register Now!** * **Cancel** | | |
|  | **Step 5:** The client selects the ‘**Register Now!’ button.** | |  | **Step 6:**  The system retrieves the **Client\_ID** of the logged in client from the **Client** table using **LINQ queries** and **Entity Framework.** | | |
|  |  | |  | **Step 7:** The system will retrieve the **Challenge\_Client\_Status\_ID** where the **Challenge\_Client\_Stat\_Desc** is equal to ‘**Unconfirmed’** in the **Challenge\_Client\_Status** table. | | |
|  |  | |  | **Step 8:** The system will use the **Challenge\_ID** retrieved in step 2 and the **Client\_ID** retrieved in step 7 as well as the **Challenge\_Client\_Status\_ID** retrieved in step 8to create a new entry in the **Client\_Challenge** table which will link the client to the challenge that he/she has registered for. | | |
|  |  | |  | **Step 9:** The system will display a success message box to the user with the following controls:  **Labels:**   * **Challenge registration successful!** * **Your exercise plan will be emailed to you when your payment has been confirmed.** * **Remember the Challenge starts on the:** * **ChallengeStart –** Populated by the **Challenge\_Start\_Date** attribute retrieved in step 2. * **Once confirmed you will need to book a consultation once every month in order to qualify for the prize at the end of the challenge.** * **Please make the payment for the challenge in your basket, if you log out before making the payment the item will be removed.**   **Button:**   * **Ok** | | |
|  | **Step 10:** The Client selects to the ‘**Ok’ button.** | |  | **Step 11:** The system closes the message box and returns the user to the top of the ‘**Home’** **page.** | | |
| **ALTERNATE COURSES:** | **Alt Step 4a:** The current dateis after the **Challenge\_Start\_Date** and therefore the challenge is running and users can no longer register. The system will display the following controls to the user on the pop up:  **Labels:**   * **The challenge is currently in progress.** * **If you would like to see any results soon and would like to see all JessterFIT Exercise plans click here.** * **If you would like to have personal communication with some of our qualified personal trainers and make personal training booking sessions and get a personalized exercise plan every month click here**.   **Buttons:**   * **Exercise Plans** * **Subscribe Now** * **Home** | | | | | |
|  | **Alt Step 4b:** The user selects the **‘Exercise Plans’ button** and the system takes the user to the ‘**Exercise Plans’** ID div. | | | | | |
|  | **Alt Step 4c:** The user selects the **‘Subscribe Now’ button** and the system will then display the Subscription Sign Up pop up. | | | | | |
|  | **Alt Step 4d:** The user selects the **‘Home’ button** and the system will then display the Subscription Sign Up pop up. | | | | | |
|  | **Alt Step 5:** The client selects the ‘Cancel’ button and the system closes the ‘Challenge’ pop up. | | | | | |
| **CONCLUSION:** | The use case ends when the client has successfully registered for a challenge. | | | | | |
| **POST-CONDITION:** | The **Challenge\_ID** is linked to the **Client\_ID** in the **Client\_Challenge** table. | | | | | |
| **BUSINESS RULES** | * There can only be one challenge running or available for sign up at a time. * Once a challenge has been confirmed a client can not unregister for a challenge. | | | | | |
| **IMPLEMENTATION CONTRAINTS AND SPECIFICATIONS** | None. | | | | | |
| **ASSUMPTIONS:** | The client exists on the system and has not yet registered for the challenge | | | | | |
| **OPEN ISSUES:** | None. | | | | | |

1.3.1.3.6 Unregister for a challenge

1.3.1.3.7 Purchase exercise plan

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **JESSTER FIT** | | | | | | |
| Author (s): Iain Steyn | | Version 2.0 | | | Date: 2019/07/23 | |
| **USE CASE NAME:** | **Purchase Exercise Plan** | | | | | **Use Case Type:** |
| **USE CASE ID:** | 3.7 | | | | | Business Requirements: 🞎 |
| **PRIORITY:** | High | | | | | System Analysis: 🞎 |
| **SOURCE:** | JessterFIT | | | | | System Design: x |
| **PRIMARY BUSINESS ACTOR** | Client | | | | | |
| **PRIMARY SYSTEM ACTOR** | None. | | | | | |
| **OTHER PARTICIPATING ACTORS:** | None. | | | | | |
| **OTHER INTERESTED STAKEHOLDERS:** | None. | | | | | |
| **DESCRIPTION:** | The Use Case describes the event that a client would like to purchase a JessterFIT exercise plan. The client will be given the option of purchasing generic plans or to purchase a custom plan which the owner will then have to create and email to the client.  The client will select which exercise plan they want and the use case will invoke **use case 3.8 Male Payment.** Once the payment has been confirmed the client will be emailed a copy of the exercise plan that they have purchased. | | | | | |
| **PRE-CONDITION:** | The client must be logged into the system. | | | | | |
| **TRIGGER:** | The client requests to view exercise plans on the home screen navbar. | | | | | |
| **TYPICAL COURSE** | Actor Action | | System Response | | | |
| **OF EVENTS:** | Manual Action | Automated Action | | |
|  | **Step 1**: The client selects the **‘Exercise Plan’** option on the home page **navbar.** | |  | **Step 2:** The system will retrieve the following attributes from the **Exercise\_Plan** table:   * **Exercise\_Plan \_ID** * **Exercise\_Plan\_Type\_ID** * **Exercise\_Plan\_Price\_ID** * **Plan\_Gender\_ID** * **Exercise\_Plan\_Name**   The following details are retrieved from the **Exercise\_Plan\_Price** table using the **Exercise\_Plan\_Price\_ID:**   * **Plan\_Price\_Type** * **Plan\_Price**   The following detail is retrieved from the **Exercise\_Plan\_Type** table using the **Exercise\_Plan\_Type\_ID:**   * **Plan\_Type\_Description**   The following detail is retrieved from the **Plan\_Gender** table using the **Plan\_Gender\_ID:**   * **Exercise\_Plan\_Gender**   The system will only retrieve the above details for exercise plans where the **Plan\_Price\_Type** attribute is equal to ‘**Generic’** excepting for the **Plan\_Price** attribute. Thesystem will also retrieve the **Plan\_Price** and **Exercise\_Plan\_Price\_ID**  where the **Plan\_Price\_Type** is equal to **‘Custom’.** All of this will be done using **LINQ queries** and **Entity Framework.** | | |
|  |  | |  | **Step 3:** The system will take the user to the **‘Exercise Plans’** div on the ‘**Home’** page. | | |
|  |  | |  | **Step 4:** The system will display the following controls on the **Exercise Plans’** div:  **Labels:**   * **JessterFIT Exercise Plans** * **The best of JessterFIT:** * **Don’t want the generic or have Injuries that stop you from doing exercises?** * **Get the JessterFIT custom plan deal where we will personally tailor an exercise plan to suit your body and your injuries if any!** * **Custom Price:** * **Plan\_Price**   **Table:**  **Column Headings:**   * **Name** * **Gender** * **Type** * **Price**   **First Column:**   * **Exercise\_Plan\_Name**   **Second Column:**   * **Exercise\_Plan\_Gender**   **Third Column:**   * **Plan\_Type\_Description**   **Forth Column:**   * **Plan\_Price (**for generic exercise plan price types)   **Fifth Column:**   * For every exercise plan displayed there will be a ‘**Purchase’** **button** displayed.   **Button:**   * **Custom Plan** | | |
|  | **Step 5:** The client selects the **‘Custom Plan’** **button.** | |  | **Step 6:** The system will check to see if a client has an existing bioform by using the **Client\_ID** of the logged in client to search for an existing entry in the **Bioform** table. | | |
|  |  | |  | **Step 7:** The client does not have an existing bioform and the system invokes **use case 3.14 Create Client Bioform.** | | |
|  |  | |  | **Step 8:** The system will retrieve the following details from the **Bioform** table using the **Client\_ID:**   * **Gender\_ID** * **Injury\_ID** (if it exists) * **Client\_Weight** * **Client\_Height**   The system will do this using **LINQ queries** and **Entity Framework.** | | |
|  |  | |  | **Step 9:** The system will retrieve all of the existing **Exercise\_Plan\_Type** attributes from the **Exercise\_Plan\_Type** table i.e.:   * **Exercise\_Plan\_Type\_ID** * **Plan\_Type\_Description**   Using **LINQ queries** and **Entity Framework.** | | |
|  |  | |  | **Step 10:** The system will display the **‘Select Exercise Plan Type’ prompt box** with the following controls:    **Labels:**   * **Please select the type of exercise plan that you would like:** * **The weight loss type will be for those of you who are here to get a healthy body and prioritize losing weight as fast as possible and as healthy as possible.** * **The Muscle gain is for those of you who don’t need to lose weight, but are focusing on getting your body to that unrealistic ‘Thor’ standard – JessterFIT is where unrealistic comes to life.** * **The lifestyle type is for those of you who are here to not do intensive training, but just want to live a ‘healthy body, healthy mind’ lifestyle – we recommend this type if your work hours are high and** **neither weight loss nor muscle gain is your main priority, but rather** **you want to get a lean healthy body while crushing the stock market.**   **Dropdown List:**   * **Plan\_Type\_Description**   **Buttons:**   * **Add to Basket** * **Cancel** | | |
|  | **Step 11:** The client selects the **‘Plan Type’ dropdown list** and selects the type they would for the plan to be tailored to**.** | |  |  | | |
|  | **Step 12:** The client selects the **‘Add to Basket’**  **button.** | |  | **Step 13:** The system will use the **Client\_ID** to retrieve the following attributes from the **Client** table:   * **Client\_Name** * **Client\_Surname** | | |
|  |  | |  | **Step 14:** The system will create a new entry in the **Exercise\_Plan** table by getting the most recent **Exercise\_Plan\_ID** and adding 1 to the **Exercise\_Plan\_ID** and then adding it to **Exercise\_Plan** table. The system will then add the following details to the row:   * The selected **Exercise\_Plan\_Type\_ID** * The system will join the retrieved client name and surname (step 13) and add the word ‘Custom’ to create a unique name for the custom plan e.g. ‘IainSteynCustom’as **Exercise\_Plan\_Name.** * Retrieved **Exercise\_Plan\_Price\_ID** (step 2**)** as **Exercise\_Plan\_Price\_ID** * Retrieved **Gender\_ID** (step 8**) as Gender\_ID**   This will all be done using **LINQ queries** and **Entity Framework.** | | |
|  |  | |  | **Step 15:** The system will link the exercise plan to the client by adding the created **Exercise\_Plan\_ID** to the **Client** table. This will all be done using **LINQ queries** and **Entity Framework.** | | |
|  |  | |  | **Step 16:** The system will display the ‘**Successful Exercise Plan Added to Basket’** **message** **box** with the following controls:  **Labels:**   * **Exercise Plan Added to Basket** * Once you have made your payment and it has been confirmed your personalized gym plan will be created and emailed to you. * Please be sure to checkout the basket before you logout otherwise it will be deleted.   **Button:**   * **OK** | | |
|  | **Step 17:** The client selects the **‘OK’ button**. | |  | **Step 18:** The system closes the **alert** **box** and returns the user to the top of the ‘**Home’** **page.** | | |
| **ALTERNATE COURSES:** | **Alt Step 5:** The user selects the ‘**Purchase’** **button** for one of the generic exercise plans.   * The system will continue at step 15. | | | | | |
|  | **Alt Step 12:**  The client selects the ‘**cancel’** **button** and the system closes the **‘Select Exercise Plan Type’ prompt box** andreturns the user to the ‘**Home’** **page.** | | | | | |
| **CONCLUSION:** | The use case ends when the client has successfully added the exercise plan that they have selected to their current basket. | | | | | |
| **POST-CONDITION:** | The **Exercise\_Plan\_ID** is linked to the clientin the **Client** table. | | | | | |
| **BUSINESS RULES** | None. | | | | | |
| **IMPLEMENTATION CONTRAINTS AND SPECIFICATIONS** | None. | | | | | |
| **ASSUMPTIONS:** | The client exists on the system. | | | | | |
| **OPEN ISSUES:** | None. | | | | | |

1.3.1.3.8 Make payment

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **JESSTER FIT** | | | | | | |
| Author (s): | | Version 2.0 | | | Date: 2019/07/23 | |
| **USE CASE NAME:** | **Make Payment** | | | | | **Use Case Type:** |
| **USE CASE ID:** | 3.8 | | | | | Business Requirements: 🞎 |
| **PRIORITY:** | High | | | | | System Analysis: 🞎 |
| **SOURCE:** | JessterFIT | | | | | System Design: x |
| **PRIMARY BUSINESS ACTOR** | Client | | | | | |
| **PRIMARY SYSTEM ACTOR** | None. | | | | | |
| **OTHER PARTICIPATING ACTORS:** | None. | | | | | |
| **OTHER INTERESTED STAKEHOLDERS:** | None. | | | | | |
| **DESCRIPTION:** | The Use Case describes the event that a client needs to make a payment for the current basket, the client will open their basket and confirm all the items in it | | | | | |
| **PRE-CONDITION:** | The client must be logged into the system to be able to make a payment. | | | | | |
| **TRIGGER:** | The client invokes the Make Payment Use Case. | | | | | |
| **TYPICAL COURSE** | Actor Action | | System Response | | | |
| **OF EVENTS:** | Manual Action | Automated Action | | |
|  | **Step 1**: The client selects the basket icon on the ‘**Home’** page. | |  | **Step 2:** The system displays the **‘Basket’** pop up with the following controls:  **Labels:**   * **Basket** * **Current Items in Basket:** * **Total**   **Table:**  **Column Headings:**   * **Basket Item** * **Price**   **First Column:**   * The system will display the following attributes for each basket item that exists:   + If the item is an exercise plan thesystem will display the **Exercise\_Plan\_Name** attribute from the **Exercise\_Plan** table.   + If the item is a Subscription the system will displaythe **Subscription\_Length** attribute fromthe **Subscription\_Length** table with the **‘Subscription’** text afterwards eg **‘6-month Subscription’.**   + If the item is a challenge the system will display the **Challenge\_Description** attribute fromthe **Challenge\_Detail** table.   **Second Column:**   * The system will display the following attributes for each basket item that exists:   + If the item is an exercise plan thesystem will display the **Plan\_Price** attribute from the **Exercise\_Plan\_Price** table.   + If the item is a Subscription the system will displaythe **Subscription\_Amount** attribute fromthe **Subscription** table**.**   + If the item is a challenge the system will display the **Challenge\_Price** attribute fromthe **Challenge\_Detail** table.   **Third Column:**   * For every item displayed there will be a **‘Remove’** **button** displayed.   **Text Box:**   * **Basket Total** – calculated by the system by adding all the retrieved prices together (disabled).   **Buttons:**   * **Pay Now** * **Close**   The system will retrieve all necessary attributes using **LINQ queries** and **Entity Framework.** | | |
|  | **Step 3**: The client selects the **‘Pay Now’** **button.** | |  | **Step 4:**  The system will retrieve the following details necessary for the payment from the **Company** table:   * **Account\_Name** * **Account\_Number** * **Account\_Type** * **Account\_Branch** * **Account\_Branch\_Code** * **Owner\_Email**   The system will do this using **LINQ queries** and **Entity Framework.** | | |
|  |  | |  | **Step 5:** The system will close the **‘Basket’** pop up and the display the ‘**Make Payment’** page with the following controls:  **Labels:**   * **Please pay the following amount into the following account:** * **Payment Amount – retrieved from the use case that invoked Make Payment** * **Account\_Name** * **Account\_Number** * **Account\_Type** * **Account\_Branch** * **Account\_Branch\_Code** * **Once you have made the payment click the confirm.**   **Buttons:**   * **Confirm** * **Close** | | |
|  | **Step 6:** The client makes the EFT payment into the JessterFIT account using the provided details. | |  |  | | |
|  | **Step 7:** The client selects the ‘**Confirm’ button.** | |  | **Step 8:**  The system will retrieve the following details from the **Client** table using the **Client\_ID** of the logged in client:   * **Client\_Name** * **Cient\_Surname** * **User\_ID**   The system will retrieve the following details from the **User** table using the **User\_ID** of the logged in client:   * **Username (email address of the Client)**   The system will do this using **LINQ queries** and **Entity Framework.** | | |
|  |  | |  | **Step 9:** The system generates an email with the following attributes:   * To * Subject * Body   Each attribute is populated using the following details:   * **Owner\_Email – retrieved in step 4.** * **‘Confirm Payment’** * The body attribute will consist of the following details retrieved:   + **Client\_Name**   + **Client\_Surname**   + **Username**   + **Total amount calculated by the system.** | | |
|  |  | |  | **Step 10:** The system will retrieve the current date and retrieve the **Payment\_Status\_ID** for the **Payment\_Status\_Desc** that is equal to ‘Unconfirmed’ from the **Payment\_Status** table. | | |
|  |  | |  | **Step 11:** The system generates a unique **Payment\_ID** and by retrieving the last **Payment\_ID** from the **Payment** table and setting it to the next consecutive number. | | |
|  |  | |  | **Step 12:** The system saves the new payment using **Entity Framework** and **LINQ** queries to the **Payment** table with the following attributes:   * **Payment\_ID – Calculated in step 14.** * **Payment\_Amount – Calculated in step 2.** * **Payment\_Date – Retrieved in step 13.** * **Payment\_Status\_ID –Retrieved in step 13.** | | |
|  |  | |  | **Step 13:** The system will then display the **‘Payment Successful’** pop up with the following controls:  **Labels:**   * **An email has been sent to the owner for the owner to confirm the payment.** * **Please note that your Subscription/Challenge/Exercise Plan will only be available after the owner has confirmed the payment.**   **Buttons:**   * **OK** | | |
|  | **Step 14:** The client selects the ‘**OK’ button.** | |  | **Step 15:** The system will return the client to the **Home** page. | | |
| **ALTERNATE COURSES:** | **Alt Step 3 A:** The client selects the ‘**Remove’** **button** of one of the items in the basket. The system will delete the ID of the item removed from the **Client** table and if it is a challenge it will delete the record from the **Cient\_Challenge** table where the **Client\_ID** is linked to the **Challenge\_ID.** | | | | | |
|  | **Alt Step 3 B:**  The client selects the ‘**Close button.** The system will close the ‘**Basket’** page and return the user to the ‘**Home’** page. | | | | | |
|  | **Alt Step 7:**  The client selects the ‘**Close’ button.** The system will close the ‘**Make Payment’** page and return the user to the ‘**Home’** page. | | | | | |
| **CONCLUSION:** | The use case ends when the payment has been saved to the payment table and the owner has been emailed a notification of the payment. | | | | | |
| **POST-CONDITION:** | The payment status is set to ‘unconfirmed’ in the payment table. | | | | | |
| **BUSINESS RULES** | A client has to make an EFT to pay for any items in the basket. | | | | | |
| **IMPLEMENTATION CONTRAINTS AND SPECIFICATIONS** | None. | | | | | |
| **ASSUMPTIONS:** | The client can make an EFT. | | | | | |
| **OPEN ISSUES:** | None. | | | | | |

1.3.1.3.9 Make new personal training booking

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **JESSTER FIT** | | | | | | |
| Author (s): | | Version 2.0 | | | Date: 2019/07/23 | |
| **USE CASE NAME:** | **Make New Personal Training Booking** | | | | | **Use Case Type:** |
| **USE CASE ID:** | 3.9 | | | | | Business Requirements: 🞎 |
| **PRIORITY:** | High | | | | | System Analysis: 🞎 |
| **SOURCE:** | JessterFIT | | | | | System Design: x |
| **PRIMARY BUSINESS ACTOR** | Client | | | | | |
| **PRIMARY SYSTEM ACTOR** | None | | | | | |
| **OTHER PARTICIPATING ACTORS:** | None | | | | | |
| **OTHER INTERESTED STAKEHOLDERS:** | Personal Trainer | | | | | |
| **DESCRIPTION:** | The Use Case describes the process of a client making a new personal training booking. The system will retrieve all available booking slots within the next two weeks from the **Booking** table. The client will then select a slot that suits them and then the system will notify the personal trainer and update the **Booking** table to reflect the booking. | | | | | |
| **PRE-CONDITION:** | The client must be logged into the system and have a running subscription. | | | | | |
| **TRIGGER:** | A client wants to book a personal training session. | | | | | |
| **TYPICAL COURSE** | Actor Action | | System Response | | | |
| **OF EVENTS:** | Manual Action | Automated Action | | |
|  | **Step 1**: The Client selects their Client profile icon. | |  | **Step 2:** The system invokes **use case 3.17 View Profile.** | | |
|  |  | |  | **Step 3:** The system will retrieve the current date. | | |
|  |  | |  | **Step 4:** The system will retrieve the following details for upcoming bookings from the **Booking** table:   * **Booking\_ID** * **Booking\_Slot\_ID** * **Booking\_Type\_ID** * **Booking\_Status\_ID** * **Employee\_ID**   The following details are retrieved from the **Booking\_Slot** table using the **Booking\_Slot\_ID:**   * **Booking\_Slot\_Date** * **Booking\_Slot\_Start** * **Booking\_Slot\_End** * **Booking\_Slot\_Venue**   The following details are retrieved from the **Booking\_Type** table using the **Booking\_Type\_ID:**   * **Booking\_Type\_Desc**   The following details are retrieved from the **Booking\_Status** table using the **Booking\_Status\_ID:**   * **Booking\_Status\_Desc**   The following details are retrieved from the **Employee** table using the **Employee\_ID**:   * **Employee\_Name** * **Employee\_Surname**   The system will only retrieve the mentioned details above for the bookings that are linked to a **Booking\_Type\_Desc** that says ‘Personal Training’ and that are linked to a **Booking\_Slot\_Date** within two weeks of the current date retrieved in step 3 and that are linked to a  **Booking\_Status\_Desc** that says ‘available’. The system will do this using **LINQ** **queries** and **Entity** **Framework**. | | |
|  |  | |  | **Step 5:** The system will ‘**Make Personal Training Booking’** pop up with the following controls:  **Table:**  **Column Headings:**   * **Date** * **Time** * **Venue** * **Trainer**   **First Column:**   * **Booking\_Slot\_Date**   **Second Column:**   * **Booking\_Slot\_Start** and **Booking\_Slot\_End**   **Third Column:**   * **Booking\_Slot\_Venue**   **Fourth Column:**   * **Employee\_Name** and **Employee\_Surname**   **Fifth Column:**   * For every booking displayed there will be a ‘**Book’** **button** displayed. | | |
|  | **Step 6:** The client selects the ‘**Book Now’** **button** for the personal training session they would like to book. | |  | **Step 7:** The system displays the ‘**Confirm Personal Training Booking’** pop up with the selected booking details as followings:  **Table:**  **Column Headings:**   * **Date** * **Time** * **Venue** * **Trainer**   **First Column:**   * **Booking\_Slot\_Date**   **Second Column:**   * **Booking\_Slot\_Start** and **Booking\_Slot\_End**   **Third Column:**   * **Booking\_Slot\_Venue**   **Fourth Column:**   * **Employee\_Name** and **Employee\_Surname**   Below the **table** the pop up will display the following controls:  **Buttons:**   * **Confirm Booking** * **Cancel Booking** | | |
|  | **Step 8:** The client selects the ‘**Confirm** **Booking’** **button**. | |  | **Step 9:** The system will retrieve the following attribute from the **Employee** table using the **Employee\_ID** retrieved in step 4:   * **User\_ID**   The system will use the retrieved **User\_ID** to retrieve the following attribute in the **User** table:   * **Username** (The email addresses of the personal trainer)   The system will also retrieve the following attributes from the **Client** table using the logged in client’s **Client\_ID**:   * **Client\_Name** * **Client\_Surname**   The using **LINQ** **queries** and **Entity Framework.** | | |
|  |  | |  | **Step 10:** The system generates an email with the following attributes:   * To * Subject * Body   Each attribute is populated using the following details:   * **Username** (email of the booked personal trainer) * **‘Booking Confirmed’** * The body attribute will consist of the following details retrieved in step 4 & 9:   + **Client\_Name**   + **Client\_Surname**   + **Booking\_Slot\_Date**   + **Booking\_Slot\_Start**   + **Booking\_Slot\_End**   + **Booking\_Slot\_Venue** | | |
|  |  | |  | **Step 11:** The system will retrieve the **Booking\_Status\_ID** from the **Booking\_Status\_Booking** table where the **Booking\_Status\_Desc** is equal to ‘Booked’ using **LINQ** **queries** and **Entity** **Framework.** | | |
|  |  | |  | **Step 12:** The system will update the **Booking\_Status\_ID** in the **Booking** table to the ID retrieved in step 11. | | |
|  |  | |  | **Step 13:** The system displays the ‘**Booking Success’** **message box** with the following controls:  **Label:**   * Booking Successful!   **Button:**   * **OK** | | |
|  | **Step 14:** The client selects the ‘OK’ button. | |  | **Step 15:** The system returns the client to the **Home** page. | | |
| **ALTERNATE COURSES:** | **Alt Step 3:** There are no **Booking\_Slot\_Date’s** -linked to the **Booking** table via the **Booking\_Slot\_ID-** in the **Booking\_Slot** table with a **Booking\_Status** - linked to the **Booking** table via the **Booking\_Status\_ID-** that is ‘available’ in the **Booking\_Status table** within the two-week criteria.   * The system will display an error message to the user using the following controls:   **Label:**   * ‘**No Available Bookings’**   **Button:**   * **OK** * The client selects the **‘OK’ button.** * The system will retrieve the owner’s **Username** from the **User** table using **LINQ queries** and **Entity Framework.** * The system generates an email with the following attributes: * To * Subject * Body   Each attribute is populated correspondingly using the following details:   * **Username** (Owner’s email) * ‘No available personal training slots.’ * ‘A client is trying to make a personal training booking, but there are no available slots within the next two weeks.’ * Return the user to the home page. | | | | | |
|  | **Alt Step 8:** The client selects the **‘Cancel Booking’ button.**   * Return to make new personal training booking page. * Return to step 4. | | | | | |
| **CONCLUSION:** | The use case ends when the client has successfully made a personal training session booking and both the client and the employee have been emailed a notification of the booking. | | | | | |
| **POST-CONDITION:** | The **Booking\_Status** has been changed to ‘booked’ in the **Booking\_Staus** table. | | | | | |
| **BUSINESS RULES** | Only a client with an active subscription can make a personal training booking. | | | | | |
| **IMPLEMENTATION CONTRAINTS AND SPECIFICATIONS** | None. | | | | | |
| **ASSUMPTIONS:** | There is at least one available booking every two weeks. | | | | | |
| **OPEN ISSUES:** | None. | | | | | |

1.3.1.3.10 Cancel personal training booking

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **JESSTER FIT** | | | | | | |
| Author (s): | | Version 2.0 | | | Date: 2019/07/23 | |
| **USE CASE NAME:** | **Cancel Personal Training Booking** | | | | | **Use Case Type:** |
| **USE CASE ID:** | 3.10 | | | | | Business Requirements: 🞎 |
| **PRIORITY:** | High | | | | | System Analysis: 🞎 |
| **SOURCE:** | JessterFIT | | | | | System Design: x |
| **PRIMARY BUSINESS ACTOR** | Client | | | | | |
| **PRIMARY SYSTEM ACTOR** | None | | | | | |
| **OTHER PARTICIPATING ACTORS:** | None | | | | | |
| **OTHER INTERESTED STAKEHOLDERS:** | Personal Trainer | | | | | |
| **DESCRIPTION:** | The Use Case describes the event that a client would like to cancel a personal training booking that they have previously booked and can no longer make. The client will select the cancel booking option on their client profile and the system will display all booked bookings to the client and the client will select the booking they would like to cancel. The system will make the booking available and display a success message to the client and email the employee a notification of the cancelation. | | | | | |
| **PRE-CONDITION:** | The client must be logged into the system and have a booked personal training session and a running subscription. | | | | | |
| **TRIGGER:** | A client wants to cancel a personal training session. | | | | | |
| **TYPICAL COURSE** | Actor Action | | System Response | | | |
| **OF EVENTS:** | Manual Action | Automated Action | | |
|  | **Step 1**: The Client selects their Client profile icon. | |  | **Step 2:** The system invokes **use case 3.17 View Profile.** | | |
|  | **Step 3**: The client selects the ‘**Cancel Training Session’** button. | |  | **Step 4:** The system will retrieve the following details for upcoming bookings from the **Booking** table:   * **Booking\_ID** * **Booking\_Slot\_ID** * **Booking\_Type\_ID** * **Booking\_Status\_ID** * **Employee\_ID**   The following details are retrieved from the **Booking\_Slot** table using the **Booking\_Slot\_ID:**   * **Booking\_Slot\_Date** * **Booking\_Slot\_Start** * **Booking\_Slot\_End** * **Booking\_Slot\_Venue**   The following details are retrieved from the **Booking\_Type** table using the **Booking\_Type\_ID:**   * **Booking\_Type\_Desc**   The following details are retrieved from the **Booking\_Status** table using the **Booking\_Status\_ID:**   * **Booking\_Status\_Desc**   The system will only retrieve the mentioned details above for a booking that is linked to client in the client table a **Booking\_ID** in the **Client** table and where the **Booking\_Type\_Desc** is equal to ‘training’. The system will do this using **LINQ** **queries** and **Entity** **Framework**. | | |
|  |  | |  | **Step 5:** The system will display the **‘Cancel Booking’** pop up with the following controls:  **Table:**  **Column Headings:**   * **Date** * **Time** * **Venue**   **First Column:**   * **Booking\_Slot\_Date**   **Second Column:**   * **Booking\_Slot\_Start** and **Booking\_Slot\_End**   **Third Column:**   * **Booking\_Slot\_Venue**   **Forth Column:**   * For every booking displayed there will be a ‘**Cancel’** **button** displayed. | | |
|  | **Step 6:** The client selects the ‘C**ancel’ button** for the personal training session they would like to cancel. | |  | **Step 7:** The system displays the ‘**Confirm Personal Training Booking Cancelation’** pop up with the following controls:  **Column Name:**   * **Date** * **Time** * **Venue**   **First Column:**   * **Booking\_Slot\_Date**   **Second Column:**   * **Booking\_Slot\_Start** and **Booking\_Slot\_End**   **Third Column:**   * **Booking\_Slot\_Venue**   Below the **table** the pop up will display the following controls:  **Buttons:**   * **Cancel Booking** * **Close** | | |
|  | **Step 8:** The client selects the ‘**Cancel** **Booking’** **button**. | |  | **Step 9:** The system uses the **Employee\_ID** retrieved in step 2 to retrieve the following attribute from the **Employee** table:   * **User\_ID**   The system will use the retrieved **User\_ID** to retrieve the following attribute in the **User** table:   * **Username** (The email address of the personal trainer)   The system will also retrieve the following attributes from the **Client** table using the logged in client’s **Client\_ID**:   * **Client\_Name** * **Client\_Surname**   The using **LINQ** **queries** and **Entity Framework.** | | |
|  |  | |  | **Step 10:** The system generates an email with the following attributes:   * To * Subject * Body   Each attribute is populated using the following details:   * **Username** (email of the booked personal trainer) * **‘Booking Cancelled’** * The body attribute will consist of the following details retrieved in step 4 & 9:   + **Client\_Name**   + **Client\_Surname**   + **Booking\_Slot\_Date**   + **Booking\_Slot\_Start**   + **Booking\_Slot\_End**   + **Booking\_Slot\_Venue** | | |
|  |  | |  | **Step 11:** The system will retrieve the **Booking\_Status\_ID** from the **Booking\_Status\_Booking** table where the **Booking\_Status\_Desc** is equal to ‘available’ using **LINQ** **queries** and **Entity** **Framework.** | | |
|  |  | |  | **Step 12:** The system will update the **Booking\_Status\_ID** in the **Booking** table to the ID retrieved in step 11. | | |
|  |  | |  | **Step 13:** The system displays the ‘**Booking Cancelation Success’** **message box** with the following controls:  **Label:**   * Booking Cancelled!   **Button:**   * **OK** | | |
|  | **Step 13:** The client selects the ‘OK’ button. | |  | **Step 14:** The system returns the client to the **Home** page. | | |
| **ALTERNATE COURSES:** | **Alt Step 3:** There are no **Booking\_Slot\_Date’s** -linked to the **Booking** table via the **Booking\_Slot\_ID-** in the **Booking\_Slot** table with a **Booking\_Status** - linked to the **Booking** table via the **Booking\_Status\_ID-** that is ‘booked’ in the **Booking\_Status table**   * The system will display an error message to the user using the following controls:   **Label:**   * ‘**No Bookings to Cancel’**   **Button:**   * **OK** * The client selects the **‘OK’ button.** * Return the user to the home page. | | | | | |
|  | **Alt Step 8:** The client selects the **‘Close’ button.**   * Return to ‘Client Profile’ page. | | | | | |
| **CONCLUSION:** | The use case ends when the client has successfully cancelled a personal training session and the employee has been emailed a notification of the booking. | | | | | |
| **POST-CONDITION:** | The **Booking\_Status** has been changed to ‘available’ in the **Booking\_Staus** table. | | | | | |
| **BUSINESS RULES** | Only a client with an active subscription can cancel a personal training booking. | | | | | |
| **IMPLEMENTATION CONTRAINTS AND SPECIFICATIONS** | None. | | | | | |
| **ASSUMPTIONS:** | The client has a booked booking. | | | | | |
| **OPEN ISSUES:** | None. | | | | | |

1.3.1.3.11 Make new consultation booking

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **JESSTER FIT** | | | | | | |
| Author (s): | | Version 2.0 | | | Date: 2019/07/23 | |
| **USE CASE NAME:** | **Make New Consultation Booking** | | | | | **Use Case Type:** |
| **USE CASE ID:** | 3.11 | | | | | Business Requirements: 🞎 |
| **PRIORITY:** | High | | | | | System Analysis: 🞎 |
| **SOURCE:** | Requirements details and Description | | | | | System Design: x |
| **PRIMARY BUSINESS ACTOR** | Client | | | | | |
| **PRIMARY SYSTEM ACTOR** | None | | | | | |
| **OTHER PARTICIPATING ACTORS:** | None | | | | | |
| **OTHER INTERESTED STAKEHOLDERS:** | Personal Trainer | | | | | |
| **DESCRIPTION:** | The Use Case describes the process of a client making a new consultation booking. The system will retrieve all available booking slots within the next four weeks from the **Booking** table. The client will then select a slot that suits them and then the system will notify the employee who will be conducting the consultation and update the **Booking** table to reflect the booking. | | | | | |
| **PRE-CONDITION:** | The client must be logged into the system and be registered for an existing challenge. | | | | | |
| **TRIGGER:** | A client wants to book a consultation session. | | | | | |
| **TYPICAL COURSE** | Actor Action | | System Response | | | |
| **OF EVENTS:** | Manual Action | Automated Action | | |
|  | **Step 1**: The Client selects their Client profile icon. | |  | **Step 2:** The system invokes **use case 3.17 View Profile.** | | |
|  | . | |  | **Step 3:** The system will retrieve the current date. | | |
|  |  | |  | **Step 4:** The system will retrieve the following details for upcoming bookings from the **Booking** table:   * **Booking\_ID** * **Booking\_Slot\_ID** * **Booking\_Type\_ID** * **Booking\_Status\_ID** * **Employee\_ID**   The following details are retrieved from the **Booking\_Slot** table using the **Booking\_Slot\_ID:**   * **Booking\_Slot\_Date** * **Booking\_Slot\_Start** * **Booking\_Slot\_End** * **Booking\_Slot\_Venue**   The following details are retrieved from the **Booking\_Type** table using the **Booking\_Type\_ID:**   * **Booking\_Type\_Desc**   The following details are retrieved from the **Booking\_Status** table using the **Booking\_Status\_ID:**   * **Booking\_Status\_Desc**   The following details are retrieved from the **Employee** table using the **Employee\_ID**:   * **Employee\_Name** * **Employee\_Surname**   The system will only retrieve the mentioned details above for the bookings that are linked to a **Booking\_Type\_Desc** that says ‘Challenge Consultation’ and that are linked to a **Booking\_Slot\_Date** within four weeks of the current date retrieved in step 3 and that are linked to a  **Booking\_Status\_Desc** that says ‘available’. The system will do this using **LINQ** **queries** and **Entity** **Framework**. | | |
|  |  | |  | **Step 5:** The system will ‘**Make Consultation Booking’** pop up with the following controls:  **Table:**  **Column Headings:**   * **Date** * **Time** * **Venue** * **Trainer**   **First Column:**   * **Booking\_Slot\_Date**   **Second Column:**   * **Booking\_Slot\_Start** and **Booking\_Slot\_End**   **Third Column:**   * **Booking\_Slot\_Venue**   **Fourth Column:**   * **Employee\_Name** and **Employee\_Surname**   **Fifth Column:**   * For every booking displayed there will be a ‘**Book’** **button** displayed. | | |
|  | **Step 6:** The client selects the ‘**Book** **Now’** button for the consultation they would like to book. | |  | **Step 7:** The system displays the ‘**Confirm Consultation Booking’** pop up with the selected booking details as followings:  **Table:**  **Column Headings:**   * **Date** * **Time** * **Venue** * **Trainer**   **First Column:**   * **Booking\_Slot\_Date**   **Second Column:**   * **Booking\_Slot\_Start** and **Booking\_Slot\_End**   **Third Column:**   * **Booking\_Slot\_Venue**   **Fourth Column:**   * **Employee\_Name** and **Employee\_Surname**   Below the **table** the pop up will display the following controls:  **Buttons:**   * **Confirm Booking** * **Cancel Booking** | | |
|  | **Step 8:** The client selects the ‘**Confirm** **Booking’** **button**. | |  | **Step 9:** The system will retrieve the following attribute from the **Employee** table using the **Employee\_ID** retrieved in step 4:   * **User\_ID**   The system will also retrieve the following attributes from the **Client** table using the logged in client’s **Client\_ID**:   * **Client\_Name** * **Client\_Surname**   The system will use the retrieved **User\_ID** to retrieve the following attribute in the **User** table:   * **Username** (The email addresses of the employee)   The using **LINQ** **queries** and **Entity Framework.** | | |
|  |  | |  | **Step 10:** The system generates an email with the following attributes:   * To * Subject * Body   Each attribute is populated using the following details:   * **Employee\_Email** * **‘Booking Confirmed’** * The body attribute will consist of the following details retrieved in step 4 & 9:   + **Client\_Name**   + **Client\_Surname**   + **Booking\_Slot\_Date**   + **Booking\_Slot\_Start**   + **Booking\_Slot\_End**   + **Booking\_Slot\_Venue** | | |
|  |  | |  | **Step 11:**  The system will retrieve the **Booking\_Status\_ID** from the **Booking\_Status\_Booking** table where the **Booking\_Status\_Desc** is equal to ‘Booked’ using **LINQ** **queries** and **Entity** **Framework.** | | |
|  |  | |  | **Step 12:**  The system will update the **Booking\_Status\_ID** in the **Booking** table to the ID retrieved in step 11. | | |
|  |  | |  | **Step 13:** The system displays the ‘**Booking Success’** **message box** with the following controls:  **Label:**   * Booking Successful!   **Button:**  **OK** | | |
|  | **Step 14:** The client selects the ‘OK’ button. | |  | **Step 15:** The system returns the client to the **Home** page. | | |
| **ALTERNATE COURSES:** | **Alt Step 3:** There are no **Booking\_Slot\_Date’s** -linked to the **Booking** table via the **Booking\_Slot\_ID-** in the **Booking\_Slot** table with a **Booking\_Status** - linked to the **Booking** table via the **Booking\_Status\_ID-** that is ‘available’ in the **Booking\_Status table** within the four-week criteria.   * The system will display an error message to the user using the following controls:   **Label:**   * ‘**No Available Bookings’**   **Button:**   * **OK** * The client selects the **‘OK’ button.** * The system will retrieve the owner’s **Username** from the **User** table using **LINQ queries** and **Entity Framework.** * The system generates an email with the following attributes: * To * Subject * Body   Each attribute is populated correspondingly using the following details:   * **Username** (Owner’s email) * ‘No available consultation slots.’ * ‘A client is trying to make a consultation booking, but there are no available slots within the next four weeks.’ * Return the user to the home page. | | | | | |
|  | **Alt Step8:** The client selects the **‘Cancel Booking’ button.**   * Return to make new consultation booking page. * Return to step 4. | | | | | |
| **CONCLUSION:** | The use case ends when the client has successfully made a consultation session booking and both the client and the employee have been emailed a notification of the booking. | | | | | |
| **POST-CONDITION:** | The **Booking\_Status** has been changed to ‘booked’ in the **Booking\_Staus** table. | | | | | |
| **BUSINESS RULES** | Only a client who is registered for an existing challenge can make a consultation booking. | | | | | |
| **IMPLEMENTATION CONTRAINTS AND SPECIFICATIONS** | None. | | | | | |
| **ASSUMPTIONS:** | There is at least one available booking every four weeks. | | | | | |
| **OPEN ISSUES:** | None. | | | | | |

1.3.1.3.12 Cancel consultation booking

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **JESSTER FIT** | | | | | | |
| Author (s): | | Version 2.0 | | | Date: 2019/07/23 | |
| **USE CASE NAME:** | **Cancel Consultation Booking** | | | | | **Use Case Type:** |
| **USE CASE ID:** | 3.12 | | | | | Business Requirements: 🞎 |
| **PRIORITY:** | High | | | | | System Analysis: 🞎 |
| **SOURCE:** | JessterFIT | | | | | System Design: x |
| **PRIMARY BUSINESS ACTOR** | Client | | | | | |
| **PRIMARY SYSTEM ACTOR** | None | | | | | |
| **OTHER PARTICIPATING ACTORS:** | None | | | | | |
| **OTHER INTERESTED STAKEHOLDERS:** | Personal Trainer | | | | | |
| **DESCRIPTION:** | The Use Case describes the event that a client would like to cancel a consultation booking that they have previously booked and can no longer make. The client will select the cancel booking option on their client profile and the system will display all booked bookings to the client and the client will select the booking they would like to cancel. The system will make the booking available and display a success message to the client and email the employee a notification of the cancelation. | | | | | |
| **PRE-CONDITION:** | The client must be logged into the system and have a booked consultation and be registered for a current challenge. | | | | | |
| **TRIGGER:** | A client wants to cancel a consultation. | | | | | |
| **TYPICAL COURSE** | Actor Action | | System Response | | | |
| **OF EVENTS:** | Manual Action | Automated Action | | |
|  | **Step 1**: The Client selects their Client profile icon. | |  | **Step 2:** The system invokes **use case 3.17 View Profile.** | | |
|  | **Step 3**: The client selects the **‘Cancel Consultation’** **button**. | |  | **Step 4:** The system will retrieve the following details for upcoming bookings from the **Booking** table:   * **Booking\_ID** * **Booking\_Slot\_ID** * **Booking\_Type\_ID** * **Booking\_Status\_ID** * **Employee\_ID**   The following details are retrieved from the **Booking\_Slot** table using the **Booking\_Slot\_ID:**   * **Booking\_Slot\_Date** * **Booking\_Slot\_Start** * **Booking\_Slot\_End** * **Booking\_Slot\_Venue**   The following details are retrieved from the **Booking\_Type** table using the **Booking\_Type\_ID:**   * **Booking\_Type\_Desc**   The following details are retrieved from the **Booking\_Status** table using the **Booking\_Status\_ID:**   * **Booking\_Status\_Desc**   The system will only retrieve the mentioned details above for a booking that is linked to client in the client table a **Booking\_ID** in the **Client** table and where the **Booking\_Type\_Desc** is equal to ‘consultation’. The system will do this using **LINQ** **queries** and **Entity** **Framework**. | | |
|  |  | |  | **Step 5:** The system will display the **‘Cancel Booking’** pop up with the following controls:  **Table:**  **Column Headings:**   * **Date** * **Time** * **Venue**   **First Column:**   * **Booking\_Slot\_Date**   **Second Column:**   * **Booking\_Slot\_Start** and **Booking\_Slot\_End**   **Third Column:**   * **Booking\_Slot\_Venue**   **Forth Column:**   * For every booking displayed there will be a ‘**Cancel’** **button** displayed. | | |
|  | **Step 6:** The client selects the ‘C**ancel’ button** for the consultation session they would like to cancel. | |  | **Step 7:** The system displays the ‘**Confirm Consultation Booking Cancelation’** pop up with the following controls:  **Column Name:**   * **Date** * **Time** * **Venue**   **First Column:**   * **Booking\_Slot\_Date**   **Second Column:**   * **Booking\_Slot\_Start** and **Booking\_Slot\_End**   **Third Column:**   * **Booking\_Slot\_Venue**   Below the **table** the pop up will display the following controls:  **Buttons:**   * **Cancel Booking** * **Close** | | |
|  | **Step 8:** The client selects the ‘**Cancel** **Booking’** **button**. | |  | **Step 9:** The system uses the **Employee\_ID** retrieved in step 2 to retrieve the following attribute from the **Employee** table:   * **User\_ID**   The system will use the retrieved **User\_ID** to retrieve the following attribute in the **User** table:   * **Username** (The email address of the personal trainer)   The system will also retrieve the following attributes from the **Client** table using the logged in client’s **Client\_ID**:   * **Client\_Name** * **Client\_Surname**   The using **LINQ** **queries** and **Entity Framework.** | | |
|  |  | |  | **Step 10:** The system generates an email with the following attributes:   * To * Subject * Body   Each attribute is populated using the following details:   * **Username** (email of the booked consultant) * **‘Booking Cancelled’** * The body attribute will consist of the following details retrieved in step 4 & 9:   + **Client\_Name**   + **Client\_Surname**   + **Booking\_Slot\_Date**   + **Booking\_Slot\_Start**   + **Booking\_Slot\_End**   + **Booking\_Slot\_Venue** | | |
|  |  | |  | **Step 11:** The system will retrieve the **Booking\_Status\_ID** from the **Booking\_Status\_Booking** table where the **Booking\_Status\_Desc** is equal to ‘available’ using **LINQ** **queries** and **Entity** **Framework.** | | |
|  |  | |  | **Step 12:** The system will update the **Booking\_Status\_ID** in the **Booking** table to the ID retrieved in step 11. | | |
|  |  | |  | **Step 13:** The system displays the ‘**Booking Cancelation Success’** **message box** with the following controls:  **Label:**   * Booking Cancelled!   **Button:**   * **OK** | | |
|  | **Step 13:** The client selects the ‘OK’ button. | |  | **Step 14:** The system returns the client to the **Home** page. | | |
| **ALTERNATE COURSES:** | **Alt Step 3:** There are no **Booking\_Slot\_Date’s** -linked to the **Booking** table via the **Booking\_Slot\_ID-** in the **Booking\_Slot** table with a **Booking\_Status** - linked to the **Booking** table via the **Booking\_Status\_ID-** that is ‘booked’ in the **Booking\_Status table**   * The system will display an error message to the user using the following controls:   **Label:**   * ‘**No Bookings to Cancel’**   **Button:**   * **OK** * The client selects the **‘OK’ button.** * Return the user to the home page. | | | | | |
|  | **Alt Step 8:** The client selects the **‘Close’ button.**   * Return to ‘Client Profile’ page. | | | | | |
| **CONCLUSION:** | The use case ends when the client has successfully cancelled a consultation booking the employee has been emailed a notification of the booking. | | | | | |
| **POST-CONDITION:** | The **Booking\_Status** has been changed to ‘available’ in the **Booking\_Staus** table. | | | | | |
| **BUSINESS RULES** | Only a client that is registered for a challenge can cancel a consultation booking. | | | | | |
| **IMPLEMENTATION CONTRAINTS AND SPECIFICATIONS** | None. | | | | | |
| **ASSUMPTIONS:** | The client has a booked booking. | | | | | |
| **OPEN ISSUES:** | None. | | | | | |

1.3.1.3.13 Create new challenge form

1.3.1.3.14 Create client bio form

1.3.1.3.15 Maintain bio form

1.3.1.3.16 Maintain bio form

1.3.1.3.17 View client profile

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **JESSTER FIT** | | | | | | |
| Author (s): | | Version 2.0 | | | Date: 2019/07/23 | |
| **USE CASE NAME:** | **View Client Profile** | | | | | **Use Case Type:** |
| **USE CASE ID:** | 3.17 | | | | | Business Requirements: 🞎 |
| **PRIORITY:** | High | | | | | System Analysis: 🞎 |
| **SOURCE:** | JessterFIT | | | | | System Design: x |
| **PRIMARY BUSINESS ACTOR** | Client | | | | | |
| **PRIMARY SYSTEM ACTOR** | None. | | | | | |
| **OTHER PARTICIPATING ACTORS:** | None. | | | | | |
| **OTHER INTERESTED STAKEHOLDERS:** | None. | | | | | |
| **DESCRIPTION:** | The Use Case describes the event that a client would like to view their personal profile on the system. The client will select their profile icon on the home page and system will display all necessary details to the client and all available options that the client may have on the page. | | | | | |
| **PRE-CONDITION:** | The client must be logged into the system and have a registered profile. | | | | | |
| **TRIGGER:** | The client needs to make a monthly subscription payment. | | | | | |
| **TYPICAL COURSE** | Actor Action | | System Response | | | |
| **OF EVENTS:** | Manual Action | Automated Action | | |
|  | **Step 1**: The client selects their personal profile icon on the **‘Home’** page. | |  | **Step 2:** The system retrieves the following attributes from the **Client** table:   * **Client\_Name** * **Client\_Number** * **Client\_Contact** * **Subscription\_Balance** (only if the balance is greater than zero) * **User\_ID**   The following attribute is retrieved from the **User** table using the **User\_ID:**   * **Username** * **Access\_Level\_ID**   The system will use the **Access\_Level\_ID** retrieved to retrieve all related **Functionality\_ID’s** in the **Access\_Functionality** table.  For every **Functionality\_ID** retrieved the system will retrieve the corresponding **Functionality\_Desc** in the **Functionality** table.  This will be done using **LINQ queries** and **Entity Framework.** | | |
|  |  | |  | **Step 3:** The system will create a **list** called the **‘Client functionality list’** and populate it with all the retrieved **Functionality\_Desc** attributes. | | |
|  |  | |  | **Step 4:** The system will use the retrieved **Client\_ID** and invoke **use case 3.17 Search Bioform** and if a bioform is found return the **Bioform\_ID** from the **Bioform** table.  This will be done using **LINQ queries** and **Entity Framework** | | |
|  |  | |  | **Step 5:** The system returns a retrieved **Bioform\_ID** which will be used by the system to retrieve the following attributes from the **Bioform** table:   * **Client\_Weight** * **Client\_Height** * **Date\_of\_Birth** * **Injuries** * **Gender\_ID**   The **Injuries** attribute can be **null** and the system will only retrieve the **Injuries** attribute if it is **not** **null.**  The following attribute is retrieved from the **Gender** table using the **Gender\_ID:**   * **Gender\_Description**   This will be done using **LINQ queries** and **Entity Framework.** | | |
|  |  | |  | **Step 6:** The system will display the ‘**Client Profile’** page with the following controls:  **Labels:**   * **Name** * **Client\_Name** * **Surname** * **Client\_Surname** * **Email** * **Username** * **Contact** * **Client\_Contact** * **Subscription Balance** (if **Subscription\_Balance** attribute is greater than zero) * **Subscription\_Balance** (if greater than zero)   The following **labels** will only be displayed if the client has an existing bioform:   * **Weight** * **Client\_Weight** * **Height** * **Client\_Height** * **Gender** * **Gender\_Desc** * **Date of Birth** * **Date\_of\_Birth** * **Injuries** * **Injuries (Attribute)**   The injuries attribute and label will only be displayed if the system returns an injuries attribute in step 4.  **Buttons:**   * **Edit Profile** * **Home**   The following **button** will only be displayed if the client has an existing bioform:   * **Edit Bioform** | | |
|  |  | |  | **Step 7:** The system will use the created **‘Client functionality list’** and check if any of the items in the list are equal to **‘Running Subscription’** and if the case is true the system will display the following **buttons** on the ‘**Client Profile’** page:   * **Make Personal Training Booking** * **Cancel Personal Training Booking** * **Make Subscription Payment** | | |
|  |  | |  | **Step 8:** The system will use the created **‘Client functionality list’** and check if any of the items in the list are equal to **‘Running Challenge** and if the case is true the system will display the following **buttons** on the ‘**Client Profile’** page:   * **Make Consultation Booking** * **Cancel Consultation Booking** * **Make Subscription Payment** | | |
|  |  | |  | **Step 9:** If the case in step 8 is true the system will use the **Bioform\_ID** retrieved to retrieve the most recent **Goal\_ID** from the **Bioform\_Challenge** table and use it to retrieve the following attribute from the **Challenge\_Form** table:   * **Goal** | | |
|  |  | |  | **Step 10:** If the case in step 8 is true the system will then display the following **labels** on the ‘**Client Profile’** page:   * **Your next Goal for the challenge:** * **Goal (Attribute)** | | |
| **ALTERNATE COURSES:** | **Alt Step 5:**  The system does not return a B**ioform\_ID** and will only display the following controls on the ‘**Client Profile’** page:  **Labels:**   * **Name** * **Client\_Name** * **Surname** * **Client\_Surname** * **Email** * **Username** * **Contact** * **Client\_Contact**   **Buttons:**   * **Edit Profile** * **Home** | | | | | |
|  | **Alt Step 7:** The client does not have **‘Running Subscription’ Functionality\_Desc** linked to their account.   * Continue at step 8. | | | | | |
|  | **Alt Step 8:** The client does not have **‘Running Challenge Functionality\_Desc** linked to their account the use case ends. | | | | | |
| **CONCLUSION:** | The use case ends when all the client details have been successfully retrieved and displayed on the ‘**Client Profile’** page. | | | | | |
| **POST-CONDITION:** | All the necessary buttons must be displayed to the client on the ‘**Client Profile’** page. | | | | | |
| **BUSINESS RULES** | None. | | | | | |
| **IMPLEMENTATION CONTRAINTS AND SPECIFICATIONS** | None. | | | | | |
| **ASSUMPTIONS:** | The client exists on the system. | | | | | |
| **OPEN ISSUES:** | None. | | | | | |

1.3.1.4 User Subsystem

1.3.1.4.1 Login

|  |  |  |  |
| --- | --- | --- | --- |
| **JESSTER FIT** | | | |
| Author (s): Joshua Eales | Version 2.0 | | Date: 2019/07/23 |
| **USE CASE NAME:** | **Login** | | **Use Case Type:** |
| **USE CASE ID:** | 4.1 | | Business Requirements: |
| **PRIORITY:** | High | | System Analysis: |
| **SOURCE:** | JessterFIT | | System Design:  x |
| **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 log into the system. The user enters login details and proceeds to login. **L**ogin details will then be validated by the system and user will be granted login if details are correct. | | |
| **PRE-CONDITION:** | The user must have login details/ profile saved on the system. | | |
| **TRIGGER:** | The user requests to login. | | |
| **TYPICAL COURSE** | Actor Action | System Response | |
| **OF EVENTS:** | Manual Action | Automated Action |
|  | **Step 1**: The user requests to access their account, by clicking (Login Option) on the navigation bar. |  | **Step 2**: The system displays the login page with the following controls:  **Labels:**   * User Name/Email * Password   **Heading:**   * Login   **Text entry fields for:**   * User Name/Email * Password   **Buttons:**   * Submit |
|  | **Step 3:** The user provides their login details as requested by the system by filling in their:  **Email address**  and  **Current Password** saved on the system into the provided text fields. |  |  |
|  | **Step 4:** User clicks the submit button. |  |  |
|  |  |  | **Step 5:** The system validates the details entered by the user, comparing it to the usernames/email addresses and passwords stored in the user table. |
|  |  |  | **Step 6**: They system determines the access level and functionality of the user based on whether the user is a client or an employee. |
|  |  |  | **Step 7:** System changes login status to actives and sets GUID for current session. |
|  |  |  | **Step 8**: The system logs the user into the system. |
| **ALTERNATE COURSES:** | **ALT Step 5**: The user login details do not exist in the system, or do not match the records in user table, display error message with the following controls:  **Text:**   * Provided username details are not found on the system! Sign Up if you don’t have a profile or please re-enter details.   **Buttons:**   * Close * Sign Up   Return to Step 3. | | |
|  | **ALT Step 5**: The user login details exist on the system; however, entered password is incorrect, display error message with controls:  **Text:**   * Provided username/ email details do not match password entered! Please re-enter password or click forgotten password to reset password.   **Button:**   * Close * Forgotten Password   Return to Step 3. | | |
| **CONCLUSION:** | The use case concludes when user is logged onto the system and has access to their various functionalities of the system. | | |
| **POST-CONDITION:** | User can access information related to their profile. | | |
| **BUSINESS RULES** | None | | |
| **IMPLEMENTATION CONTRAINTS AND SPECIFICATIONS** | None | | |
| **ASSUMPTIONS:** | None | | |
| **OPEN ISSUES:** | None | | |

1.3.1.4.2 Update Password

|  |  |  |  |
| --- | --- | --- | --- |
| **JESSTER FIT** | | | |
| Author (s): Joshua Eales | Version 2.0 | | Date: 2019/07/23 |
| **USE CASE NAME:** | Update Password | | **Use Case Type:** |
| **USE CASE ID:** | 4.2 | | Business Requirements: |
| **PRIORITY:** | High | | System Analysis: |
| **SOURCE:** | JessterFIT | | System Design:  x |
| **PRIMARY BUSINESS ACTOR** | User | | |
| **PRIMARY SYSTEM ACTOR** | None | | |
| **OTHER PARTICIPATING ACTORS:** | None | | |
| **OTHER INTERESTED STAKEHOLDERS:** | None | | |
| **DESCRIPTION:** | This use case describes the process where a user wants to update their password. The user proceeds to update the details on the system by selecting to update their password. The system requests previous password validates if its correct. The user will then be able to update their password by entering a new one. The use case concludes once the user has updated their password successfully. | | |
| **PRE-CONDITION:** | The user must be logged into the system. | | |
| **TRIGGER:** | The user requests to update their password. | | |
| **TYPICAL COURSE** | Actor Action | System Response | |
| **OF EVENTS:** | Manual Action | Automated Action |
|  | **Step 1**: The user requests update their password, by clicking (Update Password Button) on the users’ profile. |  | **Step 2**: The system displays the confirm current password page with the following controls:  **Labels:**   * Please confirm current password   **Heading:**   * Update Password   **Text entry fields for:**   * Current Password   **Buttons:**   * Submit   **Image:**   * Unlock |
|  | **Step 3:** The user provides their password details as requested by the system by filling in their:  **Current Password** saved on the system into the provided text field. |  |  |
|  | **Step 4:** User clicks the submit button. |  |  |
|  |  |  | **Step 5**: The system validates the entered details by the user, by comparing it to the passwords stored in the User table. |
|  |  |  | **Step 6**: The system displays the new password page with the following controls:  **Labels:**   * New password * Confirm new password * Enter new password and repeat password   **Heading:**   * New Password   **Text entry fields for:**   * New Password * Confirm new password   **Buttons:**   * Submit   **Image:**   * Unlock |
|  | **Step 7:** The user provides their new desired password details as requested by the system by filling in their:  **New Password**  And  Repeating new password into the provided text fields. |  |  |
|  | **Step 8:** User clicks the submit button. |  | **Step 9:** The system checks if new password matches old password. |
|  |  |  | **Step 10:** The system updates the relevant details:  Employee\_Password or Client\_Password in user table. |
|  |  |  | **Step 11:** The system displays a confirmation message that the user has successfully updated their password. |
| **ALTERNATE COURSES:** | **ALT Step 5**: The entered password is incorrect and does not match the records in user table, display error message with the following controls:  **Text:**   * Provided password confirmation information is incorrect. Please retype your password.   **Button:**   * Close   Return to Step 3. | | |
|  | **ALT Step 9**: The new password details entered are the same as old password, display error message with controls:  **Text:**   * Provided password matches old password. Please enter a new password to continue with the process.   **Button:**   * Close   Return to Step 7. | | |
| **CONCLUSION:** | The user has successfully updated their password. | | |
| **POST-CONDITION:** | Updated password details are successfully saved on the system. | | |
| **BUSINESS RULES** | Password can only be updated once previous password has been validated. | | |
| **IMPLEMENTATION CONTRAINTS AND SPECIFICATIONS** | None | | |
| **ASSUMPTIONS:** | None | | |
| **OPEN ISSUES:** | None | | |

1.3.1.4.3 Forgotten Password

|  |  |  |  |
| --- | --- | --- | --- |
| **JESSTER FIT** | | | |
| Author (s): Joshua Eales | Version 2.0 | | Date: 2019/07/23 |
| **USE CASE NAME:** | Forgotten Password | | **Use Case Type:** |
| **USE CASE ID:** | 4.3 | | Business Requirements: |
| **PRIORITY:** | High | | System Analysis: |
| **SOURCE:** | JessterFIT | | System Design:  x |
| **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 has forgotten their password. The user will select that they have forgotten the password. The system will ask for their email address. The user will provide their valid email address and the system will send a password recovery email. | | |
| **PRE-CONDITION:** | The user must be registered on the system. | | |
| **TRIGGER:** | The user has forgotten their password login details. | | |
| **TYPICAL COURSE** | Actor Action | System Response | |
| **OF EVENTS:** | Manual Action | Automated Action |
|  | **Step 1:** The user requests for new password since they have forgotten their password by clicking the (**Forgotten Password Button**). |  | **Step 2**: The system displays the forgotten password page with the following controls:  **Labels:**   * Please enter email address for recovery * Email Address   **Heading:**   * Forgotten Password   **Text entry fields for:**   * Email Address   **Buttons:**   * Submit   **Image:**   * Img Lock |
|  | **Step 3:** The user provides their details as requested by the system by filling in their: **Email address** into the provided text field. |  |  |
|  | **Step 4:** User clicks the submit button. |  |  |
|  |  |  | **Step 5:** The system validates the details entered by the user. |
|  |  |  | **Step 6**: The system sends a link to the update password page to the entered email. |
|  |  |  | **Step 7:** System display confirmation message notifying user to check the email address entered for the link. |
| **ALTERNATE COURSES:** | **ALT Step 5**: Email address entered is not a valid email address. Display error message.  **Text:**   * Email address entered is invalid. Please enter a valid email address for the recovery email to be sent to.   **Buttons:**   * Close   Return to Step 3. | | |
|  | **ALT Step 5**: The user login details exist on the system; however, entered password is incorrect, display error message with controls:  **Text:**   * Provided username/ email details do not match password entered! Please re-enter password or click forgotten password to reset password.   **Button:**   * Close * Forgotten Password   Return to Step 3. | | |
| **CONCLUSION:** | The user has updated their forgotten password. | | |
| **POST-CONDITION:** | The user has access to the system using their new password. | | |
| **BUSINESS RULES** | None | | |
| **IMPLEMENTATION CONTRAINTS AND SPECIFICATIONS** | None | | |
| **ASSUMPTIONS:** | None | | |
| **OPEN ISSUES:** | None | | |

1.3.1.4.4 Logout

|  |  |  |  |
| --- | --- | --- | --- |
| **JESSTER FIT** | | | |
| Author (s): Joshua Eales | Version 2.0 | | Date: 2019/07/23 |
| **USE CASE NAME:** | Log Out | | **Use Case Type:** |
| **USE CASE ID:** | 4.4 | | Business Requirements: |
| **PRIORITY:** | High | | System Analysis: |
| **SOURCE:** | JessterFIT | | System Design:  x |
| **PRIMARY BUSINESS ACTOR** | User | | |
| **PRIMARY SYSTEM ACTOR** | None | | |
| **OTHER PARTICIPATING ACTORS:** | None | | |
| **OTHER INTERESTED STAKEHOLDERS:** | None | | |
| **DESCRIPTION:** | This use case describes the process where the user is done using the system for the time being and they request to log out of the system. The user selects to logout and the system logs them out and returns them to the login home screen. | | |
| **PRE-CONDITION:** | The user must be logged in the system | | |
| **TRIGGER:** | The user wants to Log out of the system | | |
| **TYPICAL COURSE** | Actor Action | System Response | |
| **OF EVENTS:** | Manual Action | Automated Action |
|  | **Step 1:** The user requests to logout of the system by clicking the (Log Out **Button**) in the navigation bar. |  | **Step 2**: The system displays the confirm logout page with the following controls:  **Labels:**   * Please confirm decision to logout.   **Heading:**   * Confirm Logout   **Buttons:**   * Confirm Logout * Cancel   **Image:**   * Door Out |
|  | **Step 3:** User clicks the confirm log out button. |  |  |
|  |  |  | **Step 4:**  System changes login status to inactive. |
|  |  |  | **Step 5**: The system display success redirects the user to the login page. |
| **ALTERNATE COURSES:** | **ALT Step 3**: User selects the cancel logout button instead.  Terminate use case. | | |
| **CONCLUSION:** | The user is logged out of the system successfully. | | |
| **POST-CONDITION:** | The user is logged out of the system successfully. | | |
| **BUSINESS RULES** | None | | |
| **IMPLEMENTATION CONTRAINTS AND SPECIFICATIONS** | None | | |
| **ASSUMPTIONS:** | None | | |
| **OPEN ISSUES:** | None | | |

1.3.1.4.5Add access level

|  |  |  |  |
| --- | --- | --- | --- |
| **JESSTER FIT** | | | |
| Author (s): Joshua Eales | Version 2.0 | | Date: 2019/07/23 |
| **USE CASE NAME:** | Add Access Level | | **Use Case Type:** |
| **USE CASE ID:** | 4.5 | | Business Requirements: |
| **PRIORITY:** | High | | System Analysis: |
| **SOURCE:** | JessterFIT | | System Design:  x |
| **PRIMARY BUSINESS ACTOR** | Employee | | |
| **PRIMARY SYSTEM ACTOR** | None | | |
| **OTHER PARTICIPATING ACTORS:** | None | | |
| **OTHER INTERESTED STAKEHOLDERS:** | None | | |
| **DESCRIPTION:** | This use case describes the event where an employee wants to add a new access level to the system. The employee must be logged into the system and select the option to a add a new access level. The employee will enter all details relating to the new access level. The use case ends when the access level is added to the system and a success message is displayed to the employee. | | |
| **PRE-CONDITION:** | The employee must be logged into the system. | | |
| **TRIGGER:** | The employee selects to add a new access level to the system. | | |
| **TYPICAL COURSE** | Actor Action | System Response | |
| **OF EVENTS:** | Manual Action | Automated Action |
|  | **Step 1:** An Employee requests to add a new Access level to the system by clicking the Add new Access Level button. |  | **Step 2**: The system displays the access level page with the following controls:  **Search Bar:**  **Labels:**   * Access Level ID * Access level description   **Heading:**   * Access Level   **Text inputs:**   * Access Level ID * Access level description   **Buttons:**   * Add Access Level * Edit * Delete   **Image:**   * DoorOut   **Table**:   * Table Headings * Access Level ID (autogenerated by system) * Access Level Description |
|  | **Step 3:** Employee enters new access level details into  Access Level Description input field. |  |  |
|  | **Step 4:** Employee clicks add access level button. |  |  |
|  |  |  | **Step 5**: The system validates entered access level details. |
|  |  |  | **Step 6**: The system checks that the access level does not already exist on the system by checking the Access Level table for same access level name. |
|  |  |  | **Step 7:** The system saves the details in the Access Level table. |
|  |  |  | **Step 8:** The system displays a message stating that a new access level has been successfully added onto the system |
| **ALTERNATE COURSES:** | **ALT Step 5**: The entered details are invalid or incomplete, display error message with controls:  **Text:**   * Access levels entered are either incorrect or invalid.   **Button:**   * Close   Return to Step 3. | | |
|  | **ALT Step 6**: The entered access level details already exist on the system, display error message with controls:  **Text:**   * Details already exist on the system.   **Button:**   * Close   Terminate Use case | | |
| **CONCLUSION:** | The use case ends when a new Access Level has been successfully added to the system and a success message is displayed for the employee. | | |
| **POST-CONDITION:** | A new Access Level has been successfully added to the system. | | |
| **BUSINESS RULES** | None | | |
| **IMPLEMENTATION CONTRAINTS AND SPECIFICATIONS** | None | | |
| **ASSUMPTIONS:** | None | | |
| **OPEN ISSUES:** | None | | |

1.3.1.4.6 Maintain access level

|  |  |  |  |
| --- | --- | --- | --- |
| **JESSTER FIT** | | | |
| Author (s): Joshua Eales | Version 2.0 | | Date: 2019/07/23 |
| **USE CASE NAME:** | Maintain Access Level | | **Use Case Type:** |
| **USE CASE ID:** | 4.6 | | Business Requirements: |
| **PRIORITY:** | High | | System Analysis: |
| **SOURCE:** | JessterFIT | | System Design:  x |
| **PRIMARY BUSINESS ACTOR** | Employee | | |
| **PRIMARY SYSTEM ACTOR** | None | | |
| **OTHER PARTICIPATING ACTORS:** | None | | |
| **OTHER INTERESTED STAKEHOLDERS:** | None | | |
| **DESCRIPTION:** | The Use Case describes the event that an employee requests to maintain an access level on the system. The employee will request to maintain an access level and the system will display the maintain access level page. The system will then prompt the user to search for an existing access level on the search access level page and invoke use case 3.7 **Search Access Level** and display the output on the page. The employee will then select the access level they would like to maintain by selecting either the edit or delete option. All changes will be saved on the system and a success message will be displayed. | | |
| **PRE-CONDITION:** | The employee must be logged into the system and have the right access levels and functionality to be able to maintain an access level. | | |
| **TRIGGER:** | The employee selects to edit an access level from the edit option. | | |
| **TYPICAL COURSE** | Actor Action | System Response | |
| **OF EVENTS:** | Manual Action | Automated Action |
|  | **Step 1**: The employee requests to maintain an access levels’ details by clicking the access level button on the navigation bar. |  | **Step 2**: The system displays the access level page with the following controls:  **Search Bar:**  **Labels:**   * Access Level ID * Access level description   **Heading:**   * Access Level   **Text inputs:**   * Access Level ID * Access level description   **Buttons:**   * Add Access Level * Edit * Delete   **Image:**   * DoorOut   **Table**:   * Table Headings * Access Level ID (autogenerated by system) * Access Level Description |
|  |  |  | **Step 3:** The system will invoke use case 3.7 Search Access level. |
|  | **Step 4:** The employee will select the edit access level option for the access level that they would like to edit by clicking the edit button. |  |  |
|  |  |  | **Step 5:** The system populates the text fields with the selected access levels details.  **Access Level ID (non- changeable)**  And  **Access Level Description (Changeable)** |
|  | **Step 6:** The employee will update the access level description by typing it into the text field provided. |  |  |
|  | **Step 7:** Employee clicks the submit button to save changes. |  |  |
|  |  |  | **Step 8:** The system will validate the access level to ensure that the details are valid access level details. |
|  |  |  | **Step 9:** The system will then update the access level for the selected Access\_Level**\_ID and Access\_Level\_Description.** |
|  |  |  | **Step 10:** The system will display a success message to the employee alerting them of successful changes. |
| **ALTERNATE COURSES:** | **Alt Step 4a:** The employee selects the delete access level option for the access level that they would like to remove by clicking the delete button.  **Alt Step 4b:** The system prompts the user to confirm if they would like to delete the selected access level by displaying a page with the following controls:  **Text:**   * Are you sure you wish to delete this record?   **Buttons:**   * Submit * Cancel   **Alt Step 4c:** The user confirms that they would like to delete the selected access level by clicking the confirm button.  **Alt Step 4cc:** The User revokes the decision to remove the selected access level by clicking the cancel button.  **Alt Step 4d:** The system removes the selected access level and its relevant details from the Access Level **table.** Display success message with controls:  **Text:**   * Selected access level record has been removed successfully   **Buttons:**   * Close | | |
|  | **Alt Step 8:** The updated access level details are invalid, display error message with controls:  **Text:**   * Details entered are invalid. Please revise them to ensure they are correct.   **Button:**   * Close   Return to step 6. | | |
| **CONCLUSION:** | The selected access level has been successfully maintained and all changes have been saved on the system and a success message has been displayed. | | |
| **POST-CONDITION:** | Access Level details have been successfully maintained. | | |
| **BUSINESS RULES** | Only a manager or an owner can maintain an Access Level. | | |
| **IMPLEMENTATION CONTRAINTS AND SPECIFICATIONS** | None | | |
| **ASSUMPTIONS:** | None | | |
| **OPEN ISSUES:** | None | | |

1.3.1.4.7 Search access level

|  |  |  |  |
| --- | --- | --- | --- |
| **JESSTER FIT** | | | |
| Author (s): Joshua Eales | Version 2.0 | | Date: 2019/07/23 |
| **USE CASE NAME:** | Search Access Level | | **Use Case Type:** |
| **USE CASE ID:** | 4.7 | | Business Requirements: |
| **PRIORITY:** | High | | System Analysis: |
| **SOURCE:** | JessterFIT | | System Design:  x |
| **PRIMARY BUSINESS ACTOR** | Employee | | |
| **PRIMARY SYSTEM ACTOR** | None | | |
| **OTHER PARTICIPATING ACTORS:** | None | | |
| **OTHER INTERESTED STAKEHOLDERS:** | None | | |
| **DESCRIPTION:** | This use case narrative describes an event in which an employee wants to search for an Access Level on the system. The system will request the employee to enter any keyword or phrase matching the existing access level details they want to return on the system. The use case will end when the access level details searched are returned to the employee. | | |
| **PRE-CONDITION:** | Employee must be logged into the system | | |
| **TRIGGER:** | Employee requests to search for an access levels’ details on the system. | | |
| **TYPICAL COURSE** | Actor Action | System Response | |
| **OF EVENTS:** | Manual Action | Automated Action |
|  | **Step 1**: Employee requests to search for an existing access levels details on the system by clicking the Access Level button. |  | **Step 2**: The system displays the access level page with the following controls:  **Search Bar:**  **Labels:**   * Access Level ID * Access level description   **Heading:**   * Access Level   **Text inputs:**   * Access Level ID * Access level description   **Buttons:**   * Add Access Level * Edit * Delete   **Table**:   * Table Headings * Access Level ID (autogenerated by system) * Access Level Description |
|  | **Step 3:** User enters search criteria of access level:  **Access Level Description**  **Or**  **Access Level ID** |  |  |
|  | **Step 4**: Employee clicks the search button. |  |  |
|  |  |  | **Step 5**: The system displays a list of access levels matching entered criteria. |
| **ALTERNATE COURSES:** | **ALT Step 5**: No records matching search criteria, display error message with controls:  **Text:**   * No access level matching search criteria.   **Button:**   * Close   Return to Step 3. | | |
| **CONCLUSION:** | The use case ends when the access level searched details are returned by the system | | |
| **POST-CONDITION:** | Access level details exist on the system. | | |
| **BUSINESS RULES** | None | | |
| **IMPLEMENTATION CONTRAINTS AND SPECIFICATIONS** | None | | |
| **ASSUMPTIONS:** | None | | |
| **OPEN ISSUES:** | None | | |

1.3.1.4.8Add functionality

|  |  |  |  |
| --- | --- | --- | --- |
| **JESSTER FIT** | | | |
| Author (s): Joshua Eales | Version 2.0 | | Date: 2019/07/23 |
| **USE CASE NAME:** | Add Functionality | | **Use Case Type:** |
| **USE CASE ID:** | 4.8 | | Business Requirements: |
| **PRIORITY:** | High | | System Analysis: |
| **SOURCE:** | JessterFIT | | System Design:  x |
| **PRIMARY BUSINESS ACTOR** | Employee | | |
| **PRIMARY SYSTEM ACTOR** | None | | |
| **OTHER PARTICIPATING ACTORS:** | None | | |
| **OTHER INTERESTED STAKEHOLDERS:** | None | | |
| **DESCRIPTION:** | This use case describes the event where an employee wants to add new functionality to the system. The employee must be logged into the system and select the option to a add a new functionality. The employee will enter all details relating to the new functionality. The use case ends when the new functionality is added to the system and a success message is displayed to the employee. | | |
| **PRE-CONDITION:** | The employee must be logged into the system. | | |
| **TRIGGER:** | The employee requests to add a new functionality to the system. | | |
| **TYPICAL COURSE** | Actor Action | System Response | |
| **OF EVENTS:** | Manual Action | Automated Action |
|  | **Step 1:** An Employee requests to add a new functionality to the system by clicking the Add new Functionality button. |  | **Step 2**: The system displays the functionality page with the following controls:  **Search Bar:**  **Labels:**   * Functionality ID * Functionality description   **Heading:**   * Functionality   **Text inputs:**   * Functionality ID * Functionality description   **Buttons:**   * Add Functionality * Edit * Delete   **Table**:   * Table Headings * Functionality ID (autogenerated by system) * Functionality Description |
|  | **Step 3:** Employee enters new functionality details into  Functionality Description input field. |  |  |
|  | **Step 4:** Employee clicks add functionality button. |  |  |
|  |  |  | **Step 5**: The system validates entered functionality details. |
|  |  |  | **Step 6**: The system checks that the functionality does not already exist on the system by checking the Functionality table for same functionality description. |
|  |  |  | **Step 7:** The system saves the details in the Functionality table. |
|  |  |  | **Step 8:** The system displays a message stating that a new functionality has been successfully added onto the system |
| **ALTERNATE COURSES:** | **ALT Step 5**: The entered details are invalid or incomplete, display error message with controls:  **Text:**   * Functionality details entered are either incorrect or invalid.   **Button:**   * Close   Return to Step 3. | | |
|  | **ALT Step 6**: The entered functionality details already exist on the system, display error message with controls:  **Text:**   * Details already exist on the system.   **Button:**   * Close   Terminate Use case | | |
| **CONCLUSION:** | The use case ends when a new Functionality has been successfully added to the system and a success message is displayed for the employee. | | |
| **POST-CONDITION:** | A new Functionality has been successfully added to the system. | | |
| **BUSINESS RULES** | None | | |
| **IMPLEMENTATION CONTRAINTS AND SPECIFICATIONS** | None | | |
| **ASSUMPTIONS:** | None | | |
| **OPEN ISSUES:** | None | | |

1.3.1.4.9 Maintain functionality

|  |  |  |  |
| --- | --- | --- | --- |
| **JESSTER FIT** | | | |
| Author (s): Joshua Eales | Version 2.0 | | Date: 2019/07/23 |
| **USE CASE NAME:** | Maintain Functionality | | **Use Case Type:** |
| **USE CASE ID:** | 4.9 | | Business Requirements: |
| **PRIORITY:** | High | | System Analysis: |
| **SOURCE:** | JessterFIT | | System Design:  x |
| **PRIMARY BUSINESS ACTOR** | Employee | | |
| **PRIMARY SYSTEM ACTOR** | None | | |
| **OTHER PARTICIPATING ACTORS:** | None | | |
| **OTHER INTERESTED STAKEHOLDERS:** | None | | |
| **DESCRIPTION:** | The Use Case describes the event that an employee requests to maintain a functionality on the system. The employee will request to maintain a functionality and the system will display the maintain functionality page. The system will then prompt the user to search for an existing functionality on the functionality page and invoke use case 3.10 **Search Functionality** and display the output on the page. The employee will then select the functionality they would like to maintain by selecting either the edit or delete option. All changes will be saved on the system and a success message will be displayed. | | |
| **PRE-CONDITION:** | The employee must be logged into the system and have the right access levels and functionality to be able to maintain a functionality. | | |
| **TRIGGER:** | The employee selects to edit a functionality from the edit option. | | |
| **TYPICAL COURSE** | Actor Action | System Response | |
| **OF EVENTS:** | Manual Action | Automated Action |
|  | **Step 1**: The employee requests to maintain a functionality’s’ details by clicking the functionality button on the navigation bar. |  | **Step 2**: The system displays the functionality page with the following controls:  **Search Bar:**  **Labels:**   * Functionality ID * Functionality description   **Heading:**   * Functionality   **Text inputs:**   * Functionality ID * Functionality description   **Buttons:**   * Add Functionality * Edit * Delete   **Table**:   * Table Headings * Functionality ID (autogenerated by system) * Functionality Description |
|  |  |  | **Step 3:** The system will invoke use case 3.10 Search Functionality. |
|  | **Step 4:** The employee will select the edit functionality option for the functionality that they would like to edit by clicking the edit button. |  |  |
|  |  |  | **Step 5:** The system populates the text fields with the selected functionality’s details.  **Functionality ID (non- changeable)**  And  **Functionality Description (Changeable)** |
|  | **Step 6:** The employee will update the functionality description by typing it into the text field provided. |  |  |
|  | **Step 7:** Employee clicks the submit button to save changes. |  |  |
|  |  |  | **Step 8:** The system will validate the functionality to ensure that the details are valid functionality details. |
|  |  |  | **Step 9:** The system will then update the functionality for the selected **Functionality\_ID and Functionality\_Description.** |
|  |  |  | **Step 10:** The system will display a success message to the employee alerting them of successful changes. |
| **ALTERNATE COURSES:** | **Alt Step 4a:** The employee selects the functionality option for the functionality that they would like to remove by clicking the delete button.  **Alt Step 4b:** The system prompts the user to confirm if they would like to delete the selected functionality by displaying a page with the following controls:  **Text:**   * Are you sure you wish to delete this record?   **Buttons:**   * Submit * Cancel   **Alt Step 4c:** The user confirms that they would like to delete the selected functionality by clicking the confirm button.  **Alt Step 4cc:** The User revokes the decision to remove the selected functionality by clicking the cancel button.  Terminate Use Case.  **Alt Step 4d:** The system removes the selected functionality and its relevant details from the **Functionality** **table.** Display success message with controls:  **Text:**   * Selected functionality record has been removed successfully   **Buttons:**   * Close | | |
|  | **Alt Step 8:** The updated functionality details are invalid, display error message with controls:  **Text:**   * Details entered are invalid. Please revise them to ensure they are correct.   **Button:**   * Close   Return to step 6. | | |
| **CONCLUSION:** | The selected functionality has been successfully maintained and all changes have been saved on the system and a success message has been displayed. | | |
| **POST-CONDITION:** | Functionality details have been successfully maintained. | | |
| **BUSINESS RULES** | Only a manager or an owner can maintain a Functionality. | | |
| **IMPLEMENTATION CONTRAINTS AND SPECIFICATIONS** | None | | |
| **ASSUMPTIONS:** | None | | |
| **OPEN ISSUES:** | None | | |

1.3.1.4.10 Search functionality

|  |  |  |  |
| --- | --- | --- | --- |
| **JESSTER FIT** | | | |
| Author (s): Joshua Eales | Version 2.0 | | Date: 2019/07/23 |
| **USE CASE NAME:** | Search Functionality | | **Use Case Type:** |
| **USE CASE ID:** | 4.10 | | Business Requirements: |
| **PRIORITY:** | High | | System Analysis: |
| **SOURCE:** | JessterFIT | | System Design:  x |
| **PRIMARY BUSINESS ACTOR** | Employee | | |
| **PRIMARY SYSTEM ACTOR** | None | | |
| **OTHER PARTICIPATING ACTORS:** | None | | |
| **OTHER INTERESTED STAKEHOLDERS:** | None | | |
| **DESCRIPTION:** | This use case narrative describes an event in which an employee wants to search for a Functionality on the system. The system will request the employee to enter any keyword or phrase matching the existing functionalities details they want to return on the system. The use case will end when the functionalities details searched are returned to the employee. | | |
| **PRE-CONDITION:** | The employee must be logged into the system. | | |
| **TRIGGER:** | Employee request to search for a functionality on the system. | | |
| **TYPICAL COURSE** | **Actor Action** | System Response | |
| **OF EVENTS:** | Manual Action | Automated Action |
|  | **Step 1**: Employee requests to search for an existing functionality’s’ details on the system by clicking the Functionality button. |  | **Step 2**: The system displays the functionality page with the following controls:  **Search Bar:**  **Labels:**   * Functionality ID * Functionality description   **Heading:**   * Functionality   **Text inputs:**   * Functionality ID * Functionality description   **Buttons:**   * Add Functionality * Edit * Delete   **Table**:   * Table Headings * Functionality ID (autogenerated by system) * Functionality Description |
|  | **Step 3:** User enters search criteria of the functionality:  **Functionality Description**  **Or**  **Functionality ID** |  |  |
|  | **Step 4**: Employee clicks the search button. |  |  |
|  |  |  | **Step 5**: The system displays a list of functionalities matching entered criteria. |
| **ALTERNATE COURSES:** | **ALT Step 5**: No records matching search criteria, display error message with controls:  **Text:**   * No functionality matching search criteria.   **Button:**   * Close   Return to Step 3. | | |
| **CONCLUSION:** | The use case ends when the functionality searched details are returned by the system | | |
| **POST-CONDITION:** | Functionality details exist on the system. | | |
| **BUSINESS RULES** | None | | |
| **IMPLEMENTATION CONTRAINTS AND SPECIFICATIONS** | None | | |
| **ASSUMPTIONS:** | None | | |
| **OPEN ISSUES:** | None | | |

1.3.1.5 Report Subsystem

1.3.1.5.1 Generate exercise plan report

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | JessterFIT | **Date: 2019/04/28** | | **Authors(s): Joshua Eales** | **Version 1.3** |   **JESSTER FIT** | | | |
| Author (s): Joshua Eales | Version 2.0 | | Date: 2019/07/23 |
| **USE CASE NAME:** | Generate Exercise Plan report | | **Use Case Type:** |
| **USE CASE ID:** | 5.1 | | Business Requirements: |
| **PRIORITY:** | High | | System Analysis: |
| **SOURCE:** | JessterFIT | | System Design:  x |
| **PRIMARY BUSINESS ACTOR** | Owner | | |
| **PRIMARY SYSTEM ACTOR** | None | | |
| **OTHER PARTICIPATING ACTORS:** | None | | |
| **OTHER INTERESTED STAKEHOLDERS:** | Manager | | |
| **DESCRIPTION:** | The use case describes the event where the owner requests to generate a sales report. System will display the report generating screen and will prompt the user to enter a start date and end date isolating sales over a certain period. Report generated will retrieve information from all sales transactions completed within the time specified by the owner. Once the system has retrieved the required sales information. The system will generate the report creating an onscreen display of Exercise Plans and displays it allowing the owner to see how sales are doing in each individual product offered by the business. Exercise Plan name will be included on the report as well as amounts bought and the total amount for the date. A grand total adding all sales will be at the bottom representing the total of all sales during the time specified. There will also be options for a pdf of the report to be saved or printed. | | |
| **PRE-CONDITION:** | Owner must be logged into the system. | | |
| **TRIGGER:** | The owner requests to generate a sales report. | | |
| **TYPICAL COURSE** | **Actor Action** | System Response | |
| **OF EVENTS:** | Manual Action | Automated Action |
|  | **Step 1**: Owner requests to generate a exercise plan report for a specific period on the system by clicking the Generate Report button. |  | **Step 2**: The system displays the generate report page with the following controls:  **Heading:**  Reports  **Buttons:**   * Generate Exercise plan Report * Generate Subscription Report * Generate Challenge Entry Report |
|  | **Step 3:** Owner clicks the Generate Sales report button. |  |  |
|  |  |  | **Step 4**: The system returns a pop- up page requesting owner to select a challenge to report on with the following controls:  **Label:**   * Specify period to report on. * Start Date: * End Date:   **Input fields:**   * Day: Month: Year (for both from and to dates)   **Buttons:**   * Generate |
|  | **Step 5**: Owner enters a start and end date for the report. |  |  |
|  | **Step 6:** Owner clicks generate report button. |  |  |
|  |  |  | **Step 7:** The system validates the start and end date. |
|  |  |  | **Step 7:** System generates an onscreen report with the following controls:  **Heading:**   * Exercise Plan report + * Start Date and End date entered previously.   **Table:**  **Table Headings:**   * Exercise Plan Name * Price * Date * Sale Amount   **Label:**   * Grand Total + amount of all sales in period specified combined. * Start Date * End Date * Top Seller * Worst Seller   **Graph:**   * Type: Bar Graph * Heading: Amount of each type of exercise plan bought. * Y-axis: Exercise Plan names * X-axis: Amount Sold   **Buttons:**   * Download PDF   **SQL Transactions:**   * Read: Sale Table * Read: Exercise Plan Table |
|  | **Step 8:** Owner chooses to view on screen report. |  |  |
| **ALTERNATE COURSES:** | **Alt Step 6:** Start and end date entered are not valid. Error Message with controls:  **Labels:**   * Entered start date and end date are invalid please ensure you have selected the right dates for the report to be generated.   **Buttons:**   * Close.   Return to step 5. | | |
|  | **ALT Step 8:** Owner chooses to download PDF document representation of the report by clicking the download PDF button:  System downloads the PDF document and can be saved by the manager or printed. | | |
| **CONCLUSION:** | When the sales report is successfully displayed to the owner. | | |
| **POST-CONDITION:** | Details displayed by the system are correct for the months sales details. | | |
| **BUSINESS RULES** | None | | |
| **IMPLEMENTATION CONTRAINTS AND SPECIFICATIONS** | None | | |
| **ASSUMPTIONS:** | Business has completed sales transactions during the specified period. | | |
| **OPEN ISSUES:** | None. | | |

1.3.1.5.2 Generate Subscription report

|  |  |  |  |
| --- | --- | --- | --- |
| **JESSTER FIT** | | | |
| Author (s): Joshua Eales | Version 2.0 | | Date: 2019/07/23 |
| **USE CASE NAME:** | Generate Subscription Report | | **Use Case Type:** |
| **USE CASE ID:** | 5.2 | | Business Requirements: |
| **PRIORITY:** | High | | System Analysis: |
| **SOURCE:** | JessterFIT | | System Design:  x |
| **PRIMARY BUSINESS ACTOR** | Owner | | |
| **PRIMARY SYSTEM ACTOR** | None | | |
| **OTHER PARTICIPATING ACTORS:** | None | | |
| **OTHER INTERESTED STAKEHOLDERS:** | Manager | | |
| **DESCRIPTION:** | The use case describes the event where the owner requests to generate a subscription report. System will display the subscription report generating screen and will prompt the user to enter a start date and end date isolating number of subscriptions over a certain period. Report generated will retrieve information from all subscription registrations transactions completed within the time specified by the owner. Once the system has retrieved the required subscription information. The system will generate the report creating an onscreen display allowing the owner to see how many clients have subscribed to what type of subscription running. A grand total adding all subscriptions amounts will be at the bottom representing the total of all subscriptions during the time specified. There will also be options for a pdf of the report to be saved or printed. | | |
| **PRE-CONDITION:** | Owner must be logged into the system. | | |
| **TRIGGER:** | The owner requests to generate a subscription report. | | |
| **TYPICAL COURSE** | **Actor Action** | System Response | |
| **OF EVENTS:** | Manual Action | Automated Action |
|  | **Step 1**: Owner requests to generate a subscription report for a specific period on the system by clicking the Generate Report button. |  | **Step 2**: The system displays the generate report page with the following controls:  **Heading:**  Reports  **Buttons:**   * Generate Exercise plan Report * Generate Subscription Report * Generate Challenge Entry Report |
|  | **Step 3:** Owner clicks the Generate Subscription report button. |  |  |
|  |  |  | **Step 4**: The system returns a pop- up page requesting owner to enter a start and end date for the report with the following controls:  **Label:**   * Specify period to report on. * Start Date: * End Date:   **Input fields:**   * Day: Month: Year (for both start and end dates)   **Buttons:**   * Generate |
|  | **Step 5**: Owner enters a start and end date for the report. |  |  |
|  | **Step 6:** Owner clicks generate report button. |  |  |
|  |  |  | **Step 7:** System generates an onscreen report with the following controls:  **Heading:**   * Subscription report + * Start Date and End date entered previously.   **Table:**  **Table Headings:**   * Client Name * Client ID * Subscription Description * Subscription Payment Date * Subscription payment amount   **Label:**   * Grand Total + amount of all sales in period specified combined. * Start Date * End Date * Top Seller * Worst Seller   **Graph:**   * Type: Bar Graph * Heading: Amount of each subscriptions per day of period specified. * Y-axis: Subscription Type * X-axis: Amount Bought   **Buttons:**   * Download PDF   **SQL Transactions**:   * Read: Client Table * Read: Subscription Table * Read: Subscription Detail Table |
|  | **Step 8:** Owner chooses to view on screen report. |  |  |
| **ALTERNATE COURSES:** | **Alt Step 6:** Start and end date entered are not valid. Error Message with controls:  **Labels:**   * Entered start date and end date are invalid please ensure you have selected the right dates for the report to be generated.   **Buttons:**   * Close.   Return to step 5. | | |
|  | **ALT Step 8:** Owner chooses to download PDF document representation of the report by clicking the download PDF button:  System downloads the PDF document and can be saved by the manager or printed. | | |
| **CONCLUSION:** | When the subscription report is successfully displayed to the owner. | | |
| **POST-CONDITION:** | Details displayed by the system are correct for the periods subscription details. | | |
| **BUSINESS RULES** | None | | |
| **IMPLEMENTATION CONTRAINTS AND SPECIFICATIONS** | None | | |
| **ASSUMPTIONS:** | Business has completed subscription transactions during the specified period. | | |
| **OPEN ISSUES:** | None. | | |

1.3.1.5.3 Generate Challenge entry report

|  |  |  |  |
| --- | --- | --- | --- |
| **JESSTER FIT** | | | |
| Author (s): Joshua Eales | Version 2.0 | | Date: 2019/07/23 |
| **USE CASE NAME:** | Generate Challenge Entry Report | | **Use Case Type:** |
| **USE CASE ID:** | 5.3 | | Business Requirements: |
| **PRIORITY:** | High | | System Analysis: |
| **SOURCE:** | JessterFIT | | System Design:  x |
| **PRIMARY BUSINESS ACTOR** | Owner | | |
| **PRIMARY SYSTEM ACTOR** | None | | |
| **OTHER PARTICIPATING ACTORS:** | None | | |
| **OTHER INTERESTED STAKEHOLDERS:** | Manager | | |
| **DESCRIPTION:** | The use case describes the event where the owner requests to generate a challenge booking report. System will display the challenge booking report generating screen and will prompt the user to enter a start date and end date isolating number of challenges booked over a certain period. Report generated will retrieve information from all challenge booking registrations transactions completed within the time specified by the owner. Once the system has retrieved the required challenge booking information. The system will generate the report creating an onscreen display allowing the owner to see how many clients have booked for challenges in the period specified. A grand total adding all challenge booking amounts will be at the bottom representing the total of all challenges during the time specified and number of entries for the period specified. There will also be options for a pdf of the report to be saved or printed. | | |
| **PRE-CONDITION:** | Owner must be logged into the system. | | |
| **TRIGGER:** | The owner requests to generate a challenge report. | | |
| **TYPICAL COURSE** | **Actor Action** | System Response | |
| **OF EVENTS:** | Manual Action | Automated Action |
|  | **Step 1**: Owner requests to generate a challenge report for a specific period on the system by clicking the Generate Report button. |  | **Step 2**: The system displays the generate report page with the following controls:  **Heading:**  Reports  **Buttons:**   * Generate Exercise plan Report * Generate Subscription Report * Generate Challenge Entry Report |
|  | **Step 3:** Owner clicks the Generate Challenge report button. |  |  |
|  |  |  | **Step 4**: The system returns a pop- up page requesting owner to select a challenge to report on with the following controls:  **Label:**   * Specify challenge to report on.   **Input fields:**   * Drop down list (running and previous challenge details.   **Buttons:**   * Generate   **SQL Transaction:**   * Read: Challenge Detail Table |
|  | **Step 5**: Owner selects challenge from drop down. |  |  |
|  | **Step 6:** Owner clicks generate report button. |  |  |
|  |  |  | **Step 7:** System generates an onscreen report with the following controls:  **Heading:**   * Challenge report + * Start Date and End date entered previously.   **Table:**  **Table Headings:**   * Client Name * Challenge ID * Challenge Payment Date * Challenge Payment Amount   **Label:**   * Total amount of challenge bookings for period specified + amount of all challenges booked in specified period. * Number of entries: * Start Date * End Date   **Graphs:**  **Graph 1:**   * Type: Bar Graph * Heading: Comparison of grand total to previous 2 challenges * Y-axis: Grand Total * X-axis: Amount   **Graph 2:**   * Type: Bar Graph * Heading: Comparison of number of entries to previous 2 challenges * Y-axis: Number of entries * X-axis: Amount   **Buttons:**   * Download PDF   **SQL Transactions**:   * Read: Client Table * Read: Challenge Table * Read: Challenge Detail Table |
|  | **Step 8:** Owner chooses to view on screen report. |  |  |
| **ALTERNATE COURSES:** | **Alt Step 6:** Start and end date entered are not valid. Error Message with controls:  **Labels:**   * Entered start date and end date are invalid please ensure you have selected the right dates for the report to be generated.   **Buttons:**   * Close.   Return to step 5. | | |
|  | **ALT Step 8:** Owner chooses to download PDF document representation of the report by clicking the download PDF button:  System downloads the PDF document and can be saved by the manager or printed. | | |
| **CONCLUSION:** | When the challenge booking report is successfully displayed to the owner. | | |
| **POST-CONDITION:** | Details displayed by the system are correct for the periods challenge booking details. | | |
| **BUSINESS RULES** | None | | |
| **IMPLEMENTATION CONTRAINTS AND SPECIFICATIONS** | None | | |
| **ASSUMPTIONS:** | Business has completed challenge booking transactions during the specified period. | | |
| **OPEN ISSUES:** | None. | | |

1.3.1.5.4 Generate consultation booking details report

|  |  |  |  |
| --- | --- | --- | --- |
| **JESSTER FIT** | | | |
|  | Version 2.0 | | Date: 2019/07/23 |
| **USE CASE NAME:** | Generate Consultation Booking Details Report | | **Use Case Type:** |
| **USE CASE ID:** | 5.4 | | Business Requirements: |
| **PRIORITY:** | High | | System Analysis: |
| **SOURCE:** | JessterFIT | | System Design:  x |
| **PRIMARY BUSINESS ACTOR** | Client | | |
| **PRIMARY SYSTEM ACTOR** | None | | |
| **OTHER PARTICIPATING ACTORS:** | None | | |
| **OTHER INTERESTED STAKEHOLDERS:** | Employee | | |
| **DESCRIPTION:** | The use case describes the event where a client has completed their consultation booking for the challenge. System will retrieve data from the newly created consultation booking. The report will be displayed on screen with various details relating to the client, booking slot chosen and bio form details. Download as PDF option will also be available through the clicking of the download pdf button. | | |
| **PRE-CONDITION:** | Client must be logged into the system. | | |
| **TRIGGER:** | Client requests to generate their consultation booking details report. | | |
| **TYPICAL COURSE** | **Actor Action** | System Response | |
| **OF EVENTS:** | Manual Action | Automated Action |
|  | **Step 1**: Client will click the generate consultation booking details report. By clicking the generate report button. |  |  |
|  |  |  | **Step 2:** System generates an onscreen report with the following controls:  **Heading:**   * Consultation Booking Details   **Label:**   * Client ID * Client Name * Gender * Date of Birth * Date * Time * Client Height (m) * Client Weight (kg) * Client Details * Booking Slot Details * Bio Form Details * Injuries to pay attention to   **Buttons:**   * Download PDF   **SQL Transaction:**   * Read: Bio Form Table |
|  | **Step 3:** Client chooses to view on screen report. |  |  |
| **ALTERNATE COURSES:** |  | | |
|  | **ALT Step 3:** Client chooses to download PDF document representation of the report by clicking the download PDF button:  System downloads the PDF document and can be saved by the manager or printed. | | |
| **CONCLUSION:** | When the challenge booking report is successfully displayed to the owner. | | |
| **POST-CONDITION:** | Details displayed by the system are correct for consultation booking details. | | |
| **BUSINESS RULES** | None | | |
| **IMPLEMENTATION CONTRAINTS AND SPECIFICATIONS** | None | | |
| **ASSUMPTIONS:** | Client as booked a consultation booking. | | |
| **OPEN ISSUES:** | None. | | |

1.3.1.5.5 Generate personal training booking details report

|  |  |  |  |
| --- | --- | --- | --- |
| **JESSTER FIT** | | | |
|  | Version 2.0 | | Date: 2019/07/23 |
| **USE CASE NAME:** | Generate Personal Training Booking Details Report | | **Use Case Type:** |
| **USE CASE ID:** | 5.5 | | Business Requirements: |
| **PRIORITY:** | High | | System Analysis: |
| **SOURCE:** | JessterFIT | | System Design:  x |
| **PRIMARY BUSINESS ACTOR** | Client | | |
| **PRIMARY SYSTEM ACTOR** | None | | |
| **OTHER PARTICIPATING ACTORS:** | None | | |
| **OTHER INTERESTED STAKEHOLDERS:** | Employee | | |
| **DESCRIPTION:** | The use case describes the event where a client has completed their personal training booking by selecting a booking slot. System will retrieve data from the newly created personal training booking. The report will be displayed on screen with various details relating to the client, booking slot chosen and bio form details. Download as PDF option will also be available through the clicking of the download pdf button. | | |
| **PRE-CONDITION:** | Client must be logged into the system. | | |
| **TRIGGER:** | Client requests to generate their personal training booking details report. | | |
| **TYPICAL COURSE** | **Actor Action** | System Response | |
| **OF EVENTS:** | Manual Action | Automated Action |
|  | **Step 1**: Client will click the generate consultation booking details report. By clicking the generate report button. |  |  |
|  |  |  | **Step 2:** System generates an onscreen report with the following controls:  **Heading:**   * Consultation Booking Details   **Label:**   * Client ID * Client Name * Date of Birth * Date * Time * Client Height (m) * Client Weight (kg) * Client Details * Booking Slot Details * Bio Form Details * Injuries to pay attention to   **Buttons:**   * Download PDF   **SQL Transaction:**   * Read: Bio Form Table |
|  | **Step 8:** Client chooses to view on screen report. |  |  |
| **ALTERNATE COURSES:** | **Alt Step 6:** Start and end date entered are not valid. Error Message with controls:  **Labels:**   * Entered start date and end date are invalid please ensure you have selected the right dates for the report to be generated.   **Buttons:**   * Close.   Return to step 5. | | |
|  | **ALT Step 8:** Client chooses to download PDF document representation of the report by clicking the download PDF button:  System downloads the PDF document and can be saved by the manager or printed. | | |
| **CONCLUSION:** | When the challenge booking report is successfully displayed to the owner. | | |
| **POST-CONDITION:** | Details displayed by the system are correct for consultation booking details. | | |
| **BUSINESS RULES** | None | | |
| **IMPLEMENTATION CONTRAINTS AND SPECIFICATIONS** | None | | |
| **ASSUMPTIONS:** | Client as booked a consultation booking. | | |
| **OPEN ISSUES:** | None. | | |

1.3.1.6 Exercise Subsystem

1.3.1.6.1 Add new exercise plan type

1.3.1.6.2 Maintain exercise plan type

1.3.1.6.3 Search new exercise plan type

1.3.1.6.4 Add new exercise type

1.3.1.6.5 Maintain exercise type

1.3.1.6.6 Search exercise type

1.3.1.6.7 Add new rep

1.3.1.6.8 Maintain rep

1.3.1.6.9 Add new set

1.3.1.6.10 Maintain Set

1.3.1.6.11 Add new exercise body type

1.3.1.6.12 Maintain exercise body type

1.3.1.6.13 Search exercise body type

1.3.1.6.14 Add new exercise

1.3.1.6.15 Maintain exercise

1.3.1.6.16 Search exercise

1.3.1.6.17 Add new workout

1.3.1.6.18 Maintain workout

1.3.1.6.19 Search workout

1.3.1.6.20 Add new exercise plan

1.3.1.6.21 Maintain exercise plan

1.3.1.6.22 Search exercise plan

1.3.1.6.23 Create workout

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **JESSTER FIT** | | | | |
| Author (s): Doreen Maloka | | Version 2.0 | | Date: 2019/07/24 |
| **USE CASE NAME:** | **Create Workout** | | | **Use Case Type:** |
| **USE CASE ID:** | 6.23 | | | Business Requirements: 🞎 |
| **PRIORITY:** | High | | | System Analysis: 🞎 |
| **SOURCE:** | Requirements list, details and descriptions | | | System Design: x |
| **PRIMARY BUSINESS ACTOR** | Employee | | | |
| **PRIMARY SYSTEM ACTOR** | None | | | |
| **OTHER PARTICIPATING ACTORS:** | None | | | |
| **OTHER INTERESTED STAKEHOLDERS:** | None | | | |
| **DESCRIPTION:** | The Use Case describes the process of an employee creating a unique workout to be used in an exercise plan for a client, the employee will select create a new workout on the backend system. The system will display the create new workout page and prompt the user to select all the required details for the workout and then save it on the system. Once it has been successfully added the system will display the **‘Workout has been Successfully Created’ message box.** | | | |
| **PRE-CONDITION:** | The employee must be logged into the system and have the right access levels and functionality to be able to create an workou. | | | |
| **TRIGGER:** | An employee Clicks on the **‘WORKOUTS’ item** on the ‘**Home’** page **navbar** on the backend. | | | |
| **TYPICAL COURSE** | Actor Action | | System Response | |
| **OF EVENTS:** | Manual Action | Automated Action |
|  | **Step 1**: An employee Clicks on the **‘WORKOUTS’ item** on the ‘**Home’** page **navbar** on the backend. | |  | **Step 2:** The system will display the ‘**Workouts’** pagewith the following controls:  **Labels:**   * **Workouts** * **Workout ID** * **Workout Name**   **Display Box:**   * **Workout\_ID**   **Textboxes (Editable):**   * **WorkoutName (text)** * **SearchBar (text)**   **Buttons:**   * **Add Sets** * **Search** |
|  | **Step 3:** The employee enters the name of the workout they would like to create in the **WorkoutName textbox.** | |  |  |
|  | **Step 4:** The employee selects the **‘Add Sets’ Button** | |  | **Step 5:** The system validates that the entered detail is of the correct format, namely:   * **Workout\_Name is max: 32 characters** |
|  |  | |  | **Step 6:** The system will retrieve all of the following attributes:   * All existing **Exercise\_ID’s** and theircorresponding **Exercise\_Name’s** from the **Exercise** table. * All existing **Set\_ID’s** and theircorresponding **Set\_Number’s** from the **Set** table. * All existing **Rep\_ID’s** and theircorresponding **Rep\_Amount’s** from the **Rep** table.   The system will do all of this using **LINQ queries** and **Entity Framework.** |
|  |  | |  | **Step 7:** The system displays the ‘**Create New Workout’** pop up with the following controls:  **Labels:**   * **Workout** * **Workout ID** * **Exercise Name** * **Set Number** * **Amount of Reps**   **Display Box:**   * **Workout\_ID**   **Dropdown List:**   * **Populated with retrieved Exercise\_Name attributes in step 2.**   **Combobox:**   * **Populated with retrieved Set\_Number attributes in step 2.** * **Populated with retrieved Rep\_Amounts attributes in step 2.**   **Buttons:**   * **Add Set** * **Close** |
|  | **Step 8:** The employee selects an exercise from the populated **Exercise\_Name dropdown list.** | |  |  |
|  | **Step 9:** The employee selects an exercise from the populated **Exercise\_Name dropdown list.** | |  |  |
|  | **Step 10:** The employee selects an exercise from the populated **Exercise\_Name dropdown list.** | |  |  |
|  | **Step 11:** The employee clicks on the ‘**Add Set’ button.** | |  | **Step 12:** The system will retrieve the most recent **Workout\_ID** from the **Workout** table. |
|  |  | |  | **Step 13:** The system will add the following attributes to the **Exercise\_Workout** table**:**   * **Workout\_ID** (Created by adding 1 to the retrieved **Workout\_ID** in step 11) * **Exercise\_ID** of selected **Exercise\_Name.** * **Set\_ID’s** of selected **Set\_Number.** * **Rep\_ID’s** of selected **Rep\_Amount.** |
|  |  | |  | **Step 14:** The system will display the following controls on the **‘Create New Workout’** page:  **Table:**  **Column Heading:**   * **Set Number** * **Amount of Reps**   **First Column:**   * **Set\_Number**   **Second Column:**   * **Rep\_Amount**   **Third Column:**   * **Edit Button** * **Delete Button**   **Button:**   * **Submit** |
|  | **Step 15:** The employee clicks the ‘**Submit’** button. | |  | **Step 16:** The system displays the ‘**Are You Sure Workout’ message box** with the following controls:  **Labels:**   * **Are you sure the workout is complete and there are now missing exercises that you would like to add to the workout?**   **Buttons:**   * **Yes** * **No** |
|  | **Step 17:** The employee clicks the ‘**Yes’** **button**. | |  | **Step 18:** The system displays the **‘Workout Created Successfully’ message box** with the following controls:  **Label:**   * **Workout successfully created.**   **Button:**   * **OK** |
|  | **Step 19:** The employee clicks the ‘**OK’** **button.** | |  | **Step 20:** The system closes the ‘**Create New Workout’** pop up and returns the user to the ‘**Workouts’** page. |
| **ALTERNATE COURSES:** | **ALT step 4 A:** The **Workout\_Name** field is blank. The system displays a message box with an error message and an ‘**OK’** **button**. The employee selects the ‘**OK’** **button**   * **Returns to step 4.** | | | |
| **ALT step 4 B:** The **Workout\_Name** field is in an invalid format i.e. contains numbers or invalid characters. The system displays a message box with an error message and an ‘**OK’** **button**. The employee selects the ‘**OK’** **button**   * **Returns to step 4** | | | |
| **Alt Step 11:** The client selects the **‘Close’ button.** The system closes the ‘**Create New Workout’** pop up and returns the client to the ‘**Workouts’** page**.** | | | |
| **Alt Step 15:** The client selects the **‘Close’ button.** The system closes the ‘**Create New Workout’** pop up and returns the client to the ‘**Workouts’** page**.** | | | |
| **Alt Step 17:** The client selects the **‘No’ button.** The system closes the ‘**Are You Sure Workout’ message box** and returns the client to the ‘**Create New Workout’** pop up. | | | |
| **CONCLUSION:** | The use case ends when a new rep has been successfully added to the **Exercise\_Plan table** and a success message is displayed on the screen. | | | |
| **POST-CONDITION:** | A new rep has been successfully added to the **Exercise\_Plan table** | | | |
| **BUSINESS RULES** | Only a manager or owner can add a new rep. | | | |
| **IMPLEMENTATION CONTRAINTS AND SPECIFICATIONS** |  | | | |
| **ASSUMPTIONS:** |  | | | |
| **OPEN ISSUES:** |  | | | |

1.3.1.6.24 Create exercise plan

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **JESSTER FIT** | | | | |
| Author (s): Doreen Maloka | | Version 2.0 | | Date: 2019/07/24 |
| **USE CASE NAME:** | **Create Custom Exercise Plan** | | | **Use Case Type:** |
| **USE CASE ID:** | 6.24 | | | Business Requirements: 🞎 |
| **PRIORITY:** | High | | | System Analysis: 🞎 |
| **SOURCE:** | Requirements list, details and descriptions | | | System Design: x |
| **PRIMARY BUSINESS ACTOR** | Employee | | | |
| **PRIMARY SYSTEM ACTOR** | None | | | |
| **OTHER PARTICIPATING ACTORS:** | None | | | |
| **OTHER INTERESTED STAKEHOLDERS:** | None | | | |
| **DESCRIPTION:** | The Use Case describes the process of an employee creating a unique exercise plan for a client, the employee will select the create a new exercise plan option on the backend system. The system will display the **‘Create New Exercise Plan’** page and prompt the user to select all the required details for the exercise plan and then save it on the system. Once it has been successfully added the system will display the **‘Exercise Plan Created Successfully’ message.** | | | |
| **PRE-CONDITION:** | The employee must be logged into the system and have the right access levels and functionality to be able to create an exercise plan. | | | |
| **TRIGGER:** | An employee Clicks on the **‘EXERCISE PLAN’ item** on the ‘**Home’** page **navbar** on the backend. | | | |
| **TYPICAL COURSE** | Actor Action | | System Response | |
| **OF EVENTS:** | Manual Action | Automated Action |
|  | **Step 1**: An employee Clicks on the **‘EXERCISE PLAN’ item** on the ‘**Home’** page **navbar** on the backend. | |  | **Step 2:** The system will display the ‘**Exercise Plan’** pagewith the following controls:  **Label:**   * **Exercise Plan** * **Please select the type of exercise plan you would like to create:**   **Radio buttons:**   * **Custom** * **Generic**   **Button:**   * **Begin** * **Home** |
|  | **Step 3**: The employee clicks on the **‘Custom’ radio button** on the ‘**Exercise Plan’** page. | |  |  |
|  | **Step 4**: The employee clicks on the **‘Begin’ button** on the ‘**Exercise Plan’** page. | |  | **Step 5:** The system will display the **‘Provide Client Custom Plan Details’** **prompt box** with the following controls:  **Label:**   * **Please provide the ID of the Client you would like to create the custom exercise plan for.** * **Client ID** * **Please provide the ID of the exercise plan type the client requested for their custom plan.** * **Exercise Plan Type ID** * **Both ID’s would’ve been emailed to the owner’s email account.**   **Textboxes:**   * **ClientID (text)** * **ExercisePlanTypeID (text)**   **Button:**   * **OK** * **Cancel** |
|  | **Step 6**: The employee clicks the **‘OK’ button** on the **‘Provide Client Custom Plan Details’** **prompt box.** | |  | **Step 7:** The system validates that the entered details are of the correct format, namely:   * **ClientID must be a number and max 3 characters** * **ExercisePlanTypeID must be a number and max 1 character**   The system will also validate that both ID’s exist on the system. |
|  |  | |  | **Step 8:** The system will retrieve all of the following attributes:   * The **Plan\_Type\_Description** where the **Exercise\_Plan\_Type\_ID** is equal to the provided **ExercisePlanTypeID** in the **Exercise\_Plan\_Price** table. * The **Exercise\_Plan\_Price\_ID** and the **Plan\_Price** where the **Plan\_Price\_Type** is equal to ‘**Custom’** from the **Exercise\_Plan\_Price** table. * The **Gender\_ID** from the **Bioform** table where the **Client\_ID** attribute is equal to the provided **Client\_ID** attribute. * The **Gender\_Description** from the **Gender** table where the **Gender\_ID** attribute is equal to the retrieved **Gender\_ID** attribute.   The system will do all of this using **LINQ queries** and **Entity Framework.** |
|  |  | |  | **Step 9:** The system will hide the **‘Begin’ button** anddisplay the following controls on the ‘**Exercise Plan’** page:  **Labels:**   * **Exercise Plan** * **Exercise Plan ID** * **Exercise Plan Name** * **Exercise Plan Type** * **Price** * **Gender**   **Display Box:**   * **Exercise\_Plan\_ID** * **Plan\_Type\_Description (Retrieved in step 8)** * **Plan\_Price (Retrieved in step 8)** * **Gender\_Description (Retrieved in step 8)**   **Textbox (Editable):**   * **ExercisePlanName (text)**   **Buttons:**   * **Add Exercise Plan** * **Home** |
|  | **Step 10:** The employee enters the name of the exercise plan they would like to create in the **ExercisePlanName textbox.** | |  |  |
|  | **Step 11:** The employee selects the **‘Add Exercise Plan’ button.** | |  | **Step 12:** The system validates that the entered detail is of the correct format, namely:   * **Exercise\_Plan\_Name is max: 32 characters** |
|  |  | |  | **Step 13:** The system will retrieve the most recent **Exercise\_Plan\_ID** from the **Exercise\_Plan** table. |
|  |  | |  | **Step 14:** The system will add the following attributes to the **Exercise\_Plan** table**:**   * **Exercise\_Plan\_ID** (Created by adding 1 to the retrieved **Exercise\_Plan\_ID** in step 13) * **Exercise\_Plan\_Name** provided by the user as the **Exercise\_Plan\_Name** attribute. * **Plan\_Type\_ID (Retrieved in step 8)** * **Exercise\_Plan\_Price\_ID (Retrieved in step 8)** * **Gender\_ID (Retrieved in step 8)** |
|  |  | |  | **Step 15:** The system will retrieve all of the following attributes:   * All existing **Workout\_ID’s** and theircorresponding **Workout\_Name’s** from the **Workout** table.   The system will do all of this using **LINQ queries** and **Entity Framework.** |
|  |  | |  | **Step 16:** The system displays the ‘**Create Exercise Plan’** pop up with the following controls:  **Labels:**   * **Exercise Plan** * **Exercise Plan ID** * **Exercise Plan Name** * **Exercise Plan Type** * **Price** * **Gender**   **Display Box:**   * **Exercise\_Plan\_ID** * **Plan\_Type\_Description (Retrieved in step 8)** * **Plan\_Price (Retrieved in step 8)** * **Gender\_Description (Retrieved in step 8)**   **Dropdown List:**   * Populated with retrieved **Workout\_Name** attributes instep 15.   **Buttons:**   * **Add Workout to Exercise Plan** * **Close** * **Submit** |
|  | **Step 17:** The employee selects a workout from the populated **Workout\_Name dropdown list.** | |  |  |
|  | **Step 18:** The employee clicks on the ‘**Add Workout to Exercise Plan’ button.** | |  | **Step 19:** The system will retrieve the **Workout\_ID** of the selected workoutfrom the **Workout** table. |
|  |  | |  | **Step 20:** The system will add the following attributes to the **Exercise\_Workout** table**:**   * **Exercise\_Plan\_ID** (Created in step 14) * **Workout\_ID** (Retrieved in step 19) |
|  |  | |  | **Step 21:** The system will display the following controls on the **‘Create New Exercise Plan’** page:  **Table:**  **Column Heading:**   * **Exercise Plan** * **Workout**   **First Column:**   * **Exercise\_Plan\_Name**   **Second Column:**   * **Workout\_Name**   **Third Column:**   * **Edit Button** * **Delete Button**   **Button:**   * **Add More Workouts** * **Submit** |
|  | **Step 22:** The employee clicks the ‘**Submit’** button. | |  | **Step 23:** The system displays the ‘**Are You Sure Exercise Plan message box** with the following controls:  **Labels:**   * **Are you sure the exercise plan is complete and there are no missing workouts that you would like to add to the exercise plan?**   **Buttons:**   * **Yes** * **No** |
|  | **Step 24:** The employee clicks the ‘**Yes’** **button**. | |  | **Step 25:** The system will add the following attributes to the **Exercise\_Plan** table**:**   * **Exercise\_Plan\_ID** (Created in step 14) * **Client\_ID** (Provided in step 7) |
|  |  | |  | **Step 26:** The system displays the **‘Exercise Plan Created Successfully’ message box** with the following controls:  **Label:**   * **Exercise plan successfully created.**   **Button:**   * **OK** |
|  | **Step 27:** The employee clicks the ‘**OK’** **button.** | |  | **Step 28:** The system closes the ‘**Create New Workout’** pop up and returns the user to the ‘**Workouts’** page. |
| **ALTERNATE COURSES:** | **ALT step 4 A:** The **Workout\_Name** field is blank. The system displays a message box with an error message and an ‘**OK’** **button**. The employee selects the ‘**OK’** **button**   * **Returns to step 4.** | | | |
| **ALT step 4 B:** The **Workout\_Name** field is in an invalid format i.e. contains numbers or invalid characters. The system displays a message box with an error message and an ‘**OK’** **button**. The employee selects the ‘**OK’** **button**   * **Returns to step 4** | | | |
| **Alt Step 11:** The client selects the **‘Close’ button.** The system closes the ‘**Create New Workout’** pop up and returns the client to the ‘**Workouts’** page**.** | | | |
| **Alt Step 15:** The client selects the **‘Close’ button.** The system closes the ‘**Create New Workout’** pop up and returns the client to the ‘**Workouts’** page**.** | | | |
| **Alt Step 17:** The client selects the **‘No’ button.** The system closes the ‘**Are You Sure Workout’ message box** and returns the client to the ‘**Create New Workout’** pop up. | | | |
| **CONCLUSION:** | The use case ends when a new rep has been successfully added to the **Exercise\_Plan table** and a success message is displayed on the screen. | | | |
| **POST-CONDITION:** | A new rep has been successfully added to the **Exercise\_Plan table** | | | |
| **BUSINESS RULES** | Only a manager or owner can add a new rep. | | | |
| **IMPLEMENTATION CONTRAINTS AND SPECIFICATIONS** |  | | | |
| **ASSUMPTIONS:** |  | | | |
| **OPEN ISSUES:** |  | | | |

1.4 Conclusion

This section contained the use case diagrams for every requirement in the system, and their relevant subsystems. It also included the technical use case narratives for each.

2. Process-orientated design

2.1 Introduction

The second section of deliverable 4 has the process-oriented designs of the proposed system, which holds the context diagram, a complete functional decomposition diagram, a complete set of data flow diagrams for the system, and a complete physical description of each process.

2.2 Context Diagram

IMPORTANT NOTE! The context diagram has been split up into multiple different diagrams. This is since we had some serious trouble getting Systems Architect to allow us to connect arrows the minute a diagram became too big. We apologize for the inconvenience.

















2.3 Functional Decomposition Diagram

2.3.1 Full Decomposition

Please refer to the Appendix B of this document for a larger print of the diagram due to its size.

2.3.2 Subsystem 1 – Admin Subsystem



2.3.3 Subsystem 2 – Employee Subsystem



2.3.4 Subsystem 3 – Client Subsystem



2.3.5 Subsystem 4 – User Subsystem



2.3.6 Subsystem 5 – Reporting Subsystem



2.3.7 Subsystem 6 – Exercise Subsystem Part A



Subsystem 6 – Exercise Subsystem Part B



2.4 Physical Data Flow Diagrams

2.4.1 Subsystem1 – Admin subsystem

2.4.1.1a High Level Data Flow Diagrams



2.4.1.1b



2.4.1.1c



2.4.1.d



2.4.1.2 Middle Level Data Flow Diagrams

2.4.1.2.1 Use Case 1.1 – Add new subscription detail



2.4.1.2.2 Use Case 1.2 – Maintain Subscription Detail



2.4.1.2.3 Use Case 1.3 – Search subscription detail



2.4.1.2.4 Use case 1.4 – Add new challenge



2.4.1.2.5 Use case 1.5 – Maintain Challenge



2.4.1.2.6 Use case 1.6 – Search Challenge



2.4.1.2.7 Use case 1.7 – Add new booking slot



2.4.1.2.8 Use case 1.8 – Maintain Booking slot



2.4.1.2.9 Use case 1.9 – Search booking slot



2.4.1.2.10 Use case 1.10 – Maintain company



2.4.1.2.11 Use case 1.11 – Search booking



2.4.1.2.12 Use case 1.12 – Search Subscription



2.4.1.2.13 Use case 1.13 – Search subscription payment



2.4.1.2.14 Use case 1.14 –

2.4.1.2.15 Use case 1.15 -

2.4.1.2.16 Use case 1.16 -

2.4.1.2.17 Use case 1.17 -

2.4.1.3 Primitive Level Data Flow Diagrams

2.4.1.3.1 Use case 1.1 - Add new subscription detail



2.4.1.3.2 Use Case 1.2 – Maintain Subscription Detail



2.4.1.3.3 Use case 1.3 – Search subscription detail



2.4.1.3.4 Use case 1.4 – Add new challenge



2.4.1.3.5 Use case 1.5 – Maintain Challenge



2.4.1.3.6 Use case 1.6 – Search challenge



2.4.1.3.7 Use case 1.7 – Add new booking slot



2.4.1.3.8 Use case 1.8 Maintain booking slot



2.4.1.3.9 Use case 1.9 – Search booking slot



2.4.1.3.10 Use case 1.10 – Maintain company



2.4.1.3.11 Use case 1.11 – Search booking



2.4.1.3.12 Use case 1.12 – Search subscription



2.4.1.3.13 Use case 1.3 – Search subscription payment



2.4.1.3.14 Use case 1.14 –

2.4.1.3.15 Use case 1.15 -

2.4.1.3.16 Use case 1.16 -

2.4.1.3.17 Use case 1.17 -

2.4.4 Subsystem 2 – Employee Subsystem

2.4.4.1 High Level Data Flow Diagram



2.4.4.2 Middle Level Data flow diagrams

2.4.4.2.1 Use case 2.1 – Add employee



2.4.4.2.2 Use case 2.2 – Maintain employee



2.4.4.2.3 Use case 2.3 – Search employee



2.4.4.2.4 Use case 2.4 – Add employee type



2.4.4.2.5 Use case 2.5 Maintain employee type



2.4.4.2.6 Use case 2.6 – Search employee type



2.4.4.3 Primitive level data flow diagrams

2.4.4.3.1 Use case 2.1 – Add employee



2.4.4.3.2 Use case 2.2 – Maintain employee



2.4.4.3.3 Use case 2.3 – Search employee



2.4.4.3.4 Use case 2.4 – Add employee type



2.4.4.3.5 Use case 2.5 – Maintain employee type



2.4.4.3.6 Search employee type



2.4.5 Subsystem 3 – Client Subsystem

2.4.5.1 High level data flow diagram

2.4.5.2 Middle level data flow diagrams

2.4.5.2.1 Use case 3.1 –Register new client

2.4.5.2.2 Use case 3.2 –Maintain client profile

2.4.5.2.3 Use case 3.3 – Search client

2.4.5.2.4 Use case 3.4 –Signup for subscription

2.4.5.2.5 Use case 3.5 –Register for a challenge

2.4.5.2.6 Use case 3.6 –Unregister for a challenge

2.4.5.2.7 Use case 3.7 –Purchase exercise plan

2.4.5.2.8 Use case 3.8 –Make payment

2.4.5.2.9 Use case 3.9 –Make personal training booking

2.4.5.2.10 Use case 3.10 –Cancel personal training booking

2.4.5.2.11 Use case 3.11 –Make new consultation booking

2.4.5.2.12 Use case 3.12 –Cancel consultation booking

2.4.5.2.13 Use case 3.13 –Create new challenge form

2.4.5.2.14 Use case 3.14 –Create client bio form

2.4.5.2.15 Use case 3.15 –Maintain bio form

2.4.5.2.16 Use case 3.16 –Search bio form

2.4.5.2.17 Use case 3.17 –View client profile

2.4.5.3 Primitive level data flow diagrams

2.4.5.3.1 Use case 3.1 –Register new client

2.4.5.3.2 Use case 3.2 –Maintain client profile

2.4.5.3.3 Use case 3.3 – Search client

2.4.5.3.4 Use case 3.4 –Signup for subscription

2.4.5.3.5 Use case 3.5 –Register for a challenge

2.4.5.3.6 Use case 3.6 –Unregister for a challenge

2.4.5.3.7 Use case 3.7 –Purchase exercise plan

2.4.5.3.8 Use case 3.8 –Make payment

2.4.5.3.9 Use case 3.9 –Make personal training booking

2.4.5.3.10 Use case 3.10 –Cancel personal training booking

2.4.5.3.11 Use case 3.11 –Make new consultation booking

2.4.5.3.12 Use case 3.12 –Cancel consultation booking

2.4.5.3.13 Use case 3.13 –Create new challenge form

2.4.5.3.14 Use case 3.14 –Create client bio form

2.4.5.3.15 Use case 3.15 –Maintain bio form

2.4.5.3.16 Use case 3.16 –Search bio form

2.4.5.3.17 Use case 3.17 –View client profile

2.4.6 Subsystem 4 – User Subsystem

2.4.6.1a High Level Data Flow diagram



2.4.6.1b High Level Data Flow diagram



2.4.6.2 Middle Level Data Flow Diagrams

2.4.6.2.1 Use case 4.1 – Login



2.4.6.2.2 Use case 4.2 – Update Password



2.4.6.2.3 Use case 4.3 –Forgotten Password



2.4.6.2.4 Use case 4.4 – Logout



2.4.6.2.5 Use case 4.5 – Add access level



2.4.6.2.6 Use case 4.6 – Maintain access level



2.4.6.2.7 Use case 4.7 – Search access level



2.4.6.2.8 Use case 4.8 –Add functionality



2.4.6.2.9 Use case 4.9 – Maintain functionality



2.4.6.2.10 Use case 4.10 – Search functionality



2.4.6.3 Primitive level data flow diagrams

2.4.6.3.1 Use case 4.1 – Login



2.4.6.3.2 Use case 4.2 – Update Password



2.4.6.3.3 Use case 4.3 –Forgotten Password



2.4.6.3.4 Use case 4.4 –Logout



2.4.6.3.5 Use case 4.5 –Add access level



2.4.6.3.6 Use case 4.6 –Maintain access level



2.4.6.3.7 Use case 4.7 –Search access level



2.4.6.3.8 Use case 4.8 –Add new functionality



2.4.6.3.9 Use case 4.9 –Maintain functionality



2.4.6.3.10 Use case 4.10 –Search functionality



2.4.7 Subsystem 5 – Reporting Subsystem

2.4.7.1 High Level Data Flow Diagram



2.4.7.2 Middle Level Data Flow Diagram

2.4.7.2.1 Use case 5.1 – Generate exercise plan report

P5

5.1 Generate Exercise Plan

Report

Owner

D

Exercise plan table

Alt 5.1 GUI: Owner clicks to download PDF document

SQL trans: Read

5.1 GUI: System displays on screen report with appropriate controls

5.1 GUI: Owner clicks generate report

5.1 GUI: Owner enters start and end date

5.1 GUI: System returns pop up screen with appropriate controls

5.1 GUI: Owner clicks generate exercise plan report

5.1 GUI: Display generate report screen

5.1 GUI: request to generate exercise plan report by clicking generate report

2.4.7.2.2 Use case 5.2 – Generate subscription report



2.4.7.2.3 Use case 5.3 – Generate challenge entry report



2.4.7.2.4 Use case 5.4 – Generate consultation booking details report



2.4.7.2.5 Use case 5.5 – Generate personal training booking details report



2.4.7.3 Primitive level data flow diagrams

2.4.7.3.1 Use case 5.1 – Generate exercise plan report



2.4.7.3.2 Use case 5.2 – Generate subscription report



2.4.7.3.3 Use case 5.3 – Generate challenge entry report



2.4.7.3.4 Use case 5.4 – Generate consultation booking details report



2.4.7.3.5 Use case 5.5 – Generate personal training booking details report



2.4.8 Subsystem 6 – Exercise Subsystem

2.4.8.1 High level data flow diagram

2.4.8.2 Middle level data flow diagrams

2.4.8.2.1

2.4.8.3 Primitive level data flow diagrams

2.4.8.3.1

2.5 Physical description of each process (pseudo code)

2.5.1 Subsystem 1 – Admin Subsystem

2.5.2 Subsystem 2 – Employee Subsystem

2.5.3 Subsystem 3 – Client Subsystem

2.5.4 Subsystem 4 – User Subsystem

2.5.5 Subsystem 5 – Reporting Subsystem

2.5.6 Subsystem 6 – Exercise Subsystem

8.1 Add new exercise plan

|  |  |
| --- | --- |
| Process 8.1.1 | Display and populate exercise plan screen |
| Pseudo code | On click of the “Add Exercise Plan Type Button” on the Navigation Bar  Read Data from Exercise Plan Type Table  Display table with exercise plan type details |
| Process 8.1.2 | Display Exercise Plan Type Tab |
| Pseudo code | On click of “Add Exercise Plan Type Button” on the Navigation Bar  Read Data from Exercise Plan Type Table  Display Data in the Data grid view  Display Exercise Plan Type Tab |
| Process 8.1.3 | Capture Exercise plan Type details |
| Pseudo code | On click of “Add Exercise Plan Type Button” on Add Exercise Plan Type screen  Capture information from textboxes |
| Process 8.1.4 | Check for blank fields in the textboxes |
| Pseudo Code | On click of “Add Exercise Plan Type Button” on Add Exercise Plan Type screen  Read through textboxes on screen  IF (Textbox data = ‘null’)  Continue to **ALT Step 7**  ELSE  Continue to **Step 8** |
| ALT  On click of “Add Exercise Plan Type Button” on Add Exercise Plan Type screen  Display Message box with error message “Please fill all fields”  and an “OK” button  On click “OK Button” on Message box  Display Exercise Plan Type screen  Highlight empty textbox with red colour |
| Process 8.1.5 | * Validate Exercise Plan Type information |
| Pseudo Code | Validation requirements from the Exercise Plan Type Table  IF (Captured information = VALID)  FILTER the Exercise Plan Type Table by captured information  IF (RECORD = EXIST)  Display Error Message (“record already exists”)  ELSE  Continue to process 8.1.6  END IF  ELSE  Display Error Message (“Invalid Data Entered”)  END IF |
| Process 8.1.6 | Generate a new Exercise Plan Type ID |
| Pseudo Code | Filter the Exercise Plan Type Table  IF (Table is empty)  Exercise\_Plan\_Type\_ID = 000001  Else  Read last saved Exercise\_Plan\_Type\_ID = Exercise\_Plan\_Type\_ID + 1  END IF |
| Process 8.1. | Save Exercise Plan Type details |
|  | Insert Exercise Plan Type ID and Description into Exercise Plan Type Table |
|  | VALIDATION |
| Pseudo Code | Exercise\_Plan\_Type\_Description: only alphabets, Max 250 characters, does not already exist |

8.2 Maintain new exercise plan type

|  |  |
| --- | --- |
| Process 8.2.1 | Display and populate exercise plan type screen |
| Pseudo code | On click of the “Search Exercise Plan Type Button” on the Navigation Bar  Read Data from Exercise Plan Type Table  Display table with exercise plan type details |
| Process 8.2.2 | Search Exercise plan type |
| Pseudo code | On click of “Search Exercise Plan Type Button” on the Exercise Plan Type screen  Invoke Use Case 8.3 “Search Exercise Plan Type” |
| Process 8.2.3 | Enable controls on the screens |
| Pseudo code | On Load of exercise plan type screen  Button Update Enabled = True  Button Delete Enabled = False  Textbox Description = Enabled |
| [ALT] On Click of the “Delete Button”  Continue to process 8.2.6 |
| Process 8.2.4 | Check for blank fields in the textboxes |
| Pseudo Code | On click of “Add Exercise Plan Type Button” on Add Exercise Plan Type screen  Read through textboxes on screen  IF (Textbox data = ‘null’)  Continue to ALT  ELSE  Continue to process 8.2.5 |
| ALT  On click of “Add Exercise Plan Type Button” on Add Exercise Plan Type screen  Display Message box with error message “Please fill all fields”  and an “OK” button  On click “OK Button” on Message box  Display Exercise Plan Type screen  Highlight empty textbox with red colour |
| Process 8.2.5 | Capture Updated Exercise Plan Type details |
|  | On Click of “Update Button” on the Maintain Exercise Plan Type screen  Capture information from the textboxes |
| [ALT]  On Click “Add Exercise Plan Type” on the Maintain Exercise Plan Type screen  Invoke Use Case 8.1 “Add Exercise Plan Type” |
| [ALT]  On Click of “Back Button” on the Maintain Exercise Plan Type Screen  Display Message Box for confirmation  IF Response = YES  Clear all Textboxes  Else Response = NO  Break  END IF |
| Process 8.2.6 | * Validate Exercise Plan Type information |
| Pseudo Code | Validation requirements from the Exercise Plan Type Table  IF (Captured information = VALID)  FILTER the Exercise Plan Type Table by captured information  IF (RECORD = EXIST)  Display Error Message (“record already exists”)  ELSE  Continue to process 8.1.6  END IF  ELSE  Display Error Message (“Invalid Data Entered”)  END IF |
|  | Update Exercise Plan Type Table |
| Pseudo Code | Update Exercise Plan Type Table where criteria are equal to the criteria of the selected Exercise Plan |
| [ALT]  Delete from the Exercise Plan Type Table where criteria are equal to the criteria of the selected Exercise Plan |
| Process 8.2.7 | Notify employee of successful update |
| Pseudo Code | Display Message Box “Successfully updated Exercise Plan Type” |
| [ALT]  Display Message Box “Successfully Deleted Exercise Plan Type” |
| Process | * VALIDATION |
| Pseudo Code | Exercise\_Plan\_Type\_Description: only alphabets, Max 250 characters, does not already exist |

‘

8.3 Search Exercise Plan Types

|  |  |
| --- | --- |
| Process 8.3.1 | Display and populate exercise plan Type screen |
| Pseudo code | On click of the “Search Exercise Plan Type Button” on the Navigation Bar  Read Data from Exercise Plan Type Table  Display table with Exercise Plan Type details |
| Process 8.3.2 | Display Exercise Plan Type Tab |
| Pseudo code | On click of “Search Exercise Plan Type Button” on the Navigation Bar  Read Data from Exercise Plan Type Table  Populate Textboxes with the data from the Table  Display Data in the Data grid view  Enable Exercise Plan Type ID Textbox |
| Process 8.3.3 | Capture Search parameters |
| Pseudo code | On click of “Search Exercise Plan Type Button” on Search Exercise Plan Type screen  Capture information from textboxes |
| Process 8.3.4 | Check for blank fields in the textboxes |
| Pseudo Code | On click of “Search Exercise Plan Type Button” on Search Exercise Plan Type screen  Read through textboxes on screen  IF (Textbox data = ‘null’)  Continue to **ALT Step 5**  ELSE  Continue to **Step 6** |
| ALT  On click of “Search Exercise Plan Type Button” on Search Exercise Plan Type screen  Display Message box with error message “Please fill all fields”  and an “OK” button  On click “OK Button” on Message box  Display Exercise Plan Type screen  Highlight empty textbox with red colour |
| Process 8.3.5 | Search Exercise |
| Pseudo Code | Read from the Exercise Plan Type Table  IF (Record count of Results = 0)  [ALT] Display Error Message (“Yielded no results”)  ELSE  Continue to process 8.3.6 |
| Process 8.3.6 | Filter Data Grid View |
| Pseudo Code | Display Exercise Plan Type details  Filter Data Grid View by Search Results  Display Data Grid View |

‘

8.4 Add new Exercise Types

|  |  |
| --- | --- |
| Process 8.4.1 | Display and populate exercise plan screen |
| Pseudo code | On click of the “Add Exercise Type Button” on the Navigation Bar  Read Data from Exercise Type Table  Display table with Exercise Type details |
| Process 8.4.2 | Display Exercise Type Tab |
| Pseudo code | On click of “Add Exercise Type Button” on the Navigation Bar  Read Data from Exercise Type Table  Display Data in the Data grid view  Display Exercise Type Tab |
| Process 8.4.3 | Capture Exercise Type details |
| Pseudo code | On click of “Add Exercise Type Button” on Add Exercise Type screen  Capture information from textboxes |
| Process 8.4.4 | Check for blank fields in the textboxes |
| Pseudo Code | On click of “Add Exercise Type Button” on Add Exercise Type screen  Read through textboxes on screen  IF (Textbox data = ‘null’)  Continue to **ALT Step 7**  ELSE  Continue to **Step 8** |
| ALT  On click of “Add Exercise Type Button” on Add Exercise Type screen  Display Message box with error message “Please fill all fields”  and an “OK” button  On click “OK Button” on Message box  Display Exercise Type screen  Highlight empty textbox with red colour |
| Process 8.4.5 | * Validate Exercise Type information |
| Pseudo Code | Validation requirements from the Exercise Type Table  IF (Captured information = VALID)  FILTER the Exercise Type Table by captured information  IF (RECORD = EXIST)  Display Error Message (“record already exists”)  ELSE  Continue to process 8.4.6  END IF  ELSE  Display Error Message (“Invalid Data Entered”)  END IF |
| Process 8.4.6 | Generate a new Exercise Type ID |
| Pseudo Code | Filter the Exercise Type Table  IF (Table is empty)  Exercise\_Type\_ID = 000001  Else  Read last saved Exercise\_Type\_ID = Exercise\_Type\_ID + 1  END IF |
| Process 8.4.7 | Save Exercise Type details |
|  | Insert Exercise Type ID and Description into Exercise Type Table |
|  | VALIDATION |
| Pseudo Code | Exercise\_Type\_Description: only alphabets, Max 250 characters, does not already exist |

8.5 Maintain new Exercise Type

|  |  |
| --- | --- |
| Process 8.5.1 | Display and populate Exercise Type screen |
| Pseudo code | On click of the “Search Exercise Type Button” on the Navigation Bar  Read Data from Exercise Type Table  Display table with Exercise Type details |
| Process 8.5.2 | Search Exercise Type |
| Pseudo code | On click of “Search Exercise Type Button” on the Exercise Type screen  Invoke Use Case 8.3 “Search Exercise Type” |
| Process 8.5.3 | Enable controls on the screens |
| Pseudo code | On Load of Exercise Type screen  Button Update Enabled = True  Button Delete Enabled = False  Textbox Description = Enabled |
| [ALT] On Click of the “Delete Button”  Continue to process 8.5.6 |
| Process 8.5.4 | Check for blank fields in the textboxes |
| Pseudo Code | On click of “Add Exercise Type Button” on Add Exercise Type screen  Read through textboxes on screen  IF (Textbox data = ‘null’)  Continue to **ALT Step 7**  ELSE  Continue to **Step 8** |
| ALT  On click of “Add Exercise Type Button” on Add Exercise Type screen  Display Message box with error message “Please fill all fields”  and an “OK” button  On click “OK Button” on Message box  Display Exercise Type screen  Highlight empty textbox with red colour |
| Process 8.5.5 | Capture Updated Exercise Type details |
|  | On Click of “Update Button” on the Maintain Exercise Type screen  Capture information from the textboxes |
| [ALT]  On Click “Add Exercise Type” on the Maintain Exercise Type screen  Invoke Use Case 8.1 “Add Exercise Type” |
| [ALT]  On Click of “Back Button” on the Maintain Exercise Type Screen  Display Message Box for confirmation  IF Response = YES  Clear all Textboxes  Else Response = NO  Break  END IF |
| Process 8.5.6 | Update Exercise Type Table |
| Pseudo Code | Update Exercise Type Table where criteria are equal to the criteria of the selected Exercise Plan |
| [ALT]  Delete from the Exercise Type Table where criteria are equal to the criteria of the selected Exercise Plan |
| Process 8.5.7 | Notify employee of successful update |
| Pseudo Code | Display Message Box “Successfully updated Exercise Type” |
| [ALT]  Display Message Box “Successfully Deleted Exercise Type” |

8.6 Search Exercise Types

|  |  |
| --- | --- |
| Process 8.6.1 | Display and populate Exercise Type screen |
| Pseudo code | On click of the “Search Exercise Type Button” on the Navigation Bar  Read Data from Exercise Type Table  Display table with Exercise Type details |
| Process 8.6.2 | Display Exercise Type Tab |
| Pseudo code | On click of “Search Exercise Type Button” on the Navigation Bar  Read Data from Exercise Type Table  Populate Textboxes with the data from the Table  Display Data in the Data grid view  Enable Exercise Type ID Textbox |
| Process 8.6.3 | Capture Search parameters |
| Pseudo code | On click of “Search Exercise Type Button” on Search Exercise Type screen  Capture information from textboxes |
| Process 8.6.4 | Check for blank fields in the textboxes |
| Pseudo Code | On click of “Search Exercise Type Button” on Search Exercise Type screen  Read through textboxes on screen  IF (Textbox data = ‘null’)  Continue to **ALT Step 5**  ELSE  Continue to **Step 6** |
| ALT  On click of “Search Exercise Type Button” on Search Exercise Type screen  Display Message box with error message “Please fill all fields”  and an “OK” button  On click “OK Button” on Message box  Display Exercise Type screen  Highlight empty textbox with red colour |
| Process 8.6.5 | Search Exercise Type |
| Pseudo Code | Read from the Exercise Type Table  IF (Record count of Results = 0)  [ALT] Display Error Message (“Yielded no results”)  ELSE  Continue to process 8.6.6 |
| Process 8.6.6 | Filter Data Grid View |
| Pseudo Code | Display Exercise Type details  Filter Data Grid View by Search Results  Display Data Grid View |

8. Exercises subsystem

8.7 Add new Rep

|  |  |
| --- | --- |
| Process 8.7.1 | Display and populate Rep screen |
| Pseudo code | On click of the “Add Rep Button” on the Navigation Bar  Read Data from Rep Table  Display table with Rep details |
| Process 8.7.2 | Display Rep Tab |
| Pseudo code | On click of “Add Rep Button” on the Navigation Bar  Read Data from Rep Table  Display Data in the Data grid view  Display Rep Tab |
| Process 8.7.3 | Capture Rep details |
| Pseudo code | On click of “Add Rep Button” on Add Rep screen  Capture information from textboxes |
| Process 8.7.4 | Check for blank fields in the textboxes |
| Pseudo Code | On click of “Add Rep Button” on Add Rep screen  Read through textboxes on screen  IF (Textbox data = ‘null’)  Continue to **ALT Step 7**  ELSE  Continue to **Step 8** |
| ALT  On click of “Add Rep Button” on Add Rep screen  Display Message box with error message “Please fill all fields”  and an “OK” button  On click “OK Button” on Message box  Display Rep screen  Highlight empty textbox with red colour |
| Process 8.7.5 | * Validate Rep information |
| Pseudo Code | Validation requirements from the Rep Table  IF (Captured information = VALID)  FILTER the Rep Table by captured information  IF (RECORD = EXIST)  Display Error Message (“record already exists”)  ELSE  Continue to process 8.7.6  END IF  ELSE  Display Error Message (“Invalid Data Entered”)  END IF |
| Process 8.7.6 | Generate a new Rep ID |
| Pseudo Code | Filter the Rep Table  IF (Table is empty)  Rep\_ID = 000001  Else  Read last saved Rep\_ID = Rep\_ID + 1  END IF |
| Process 8.7. | Save Rep details |
|  | Insert Rep ID and Description into Rep Table |
|  | VALIDATION |
| Pseudo Code | Rep\_Description: only alphabets, Max 250 characters, does not already exist |

8.8 Maintain new Rep

|  |  |
| --- | --- |
| Process 8.8.1 | Display and populate Rep screen |
| Pseudo code | On click of the “Search Rep Button” on the Navigation Bar  Read Data from Rep Table  Display table with Rep details |
| Process 8.8.2 | Search Rep |
| Pseudo code | On click of “Search Rep Button” on the Rep screen  Invoke Use Case 8.3 “Search Rep” |
| Process 8.8.3 | Enable controls on the screens |
| Pseudo code | On Load of Rep screen  Button Update Enabled = True  Button Delete Enabled = False  Textbox Description = Enabled |
| [ALT] On Click of the “Delete Button”  Continue to process 8.8.6 |
| Process 8.8.4 | Check for blank fields in the textboxes |
| Pseudo Code | On click of “Add Rep Button” on Add Rep screen  Read through textboxes on screen  IF (Textbox data = ‘null’)  Continue to **ALT Step 7**  ELSE  Continue to **Step 8** |
| ALT  On click of “Add Rep Button” on Add Rep screen  Display Message box with error message “Please fill all fields”  and an “OK” button  On click “OK Button” on Message box  Display Rep screen  Highlight empty textbox with red colour |
| Process 8.8.5 | Capture Updated Rep details |
|  | On Click of “Update Button” on the Maintain Rep screen  Capture information from the textboxes |
| [ALT]  On Click “Add Rep” on the Maintain Rep screen  Invoke Use Case 8.1 “Add Rep” |
| [ALT]  On Click of “Back Button” on the Maintain Rep Screen  Display Message Box for confirmation  IF Response = YES  Clear all Textboxes  Else Response = NO  Break  END IF |
| Process 8.8.6 | Update Rep Table |
| Pseudo Code | Update Rep Table where criteria are equal to the criteria of the selected Exercise Plan |
| [ALT]  Delete from the Rep Table where criteria are equal to the criteria of the selected Exercise Plan |
| Process 8.8.7 | Notify employee of successful update |
| Pseudo Code | Display Message Box “Successfully updated Rep” |
| [ALT]  Display Message Box “Successfully Deleted Rep” |

8.9 Add new Set

|  |  |
| --- | --- |
| Process 8.9.1 | Display and populate Set screen |
| Pseudo code | On click of the “Add Set Button” on the Navigation Bar  Read Data from Set Table  Display table with Set details |
| Process 8.9.2 | Display Set Tab |
| Pseudo code | On click of “Add Set Button” on the Navigation Bar  Read Data from Set Table  Display Data in the Data grid view  Display Set Tab |
| Process 8.9.3 | Capture Set details |
| Pseudo code | On click of “Add Set Button” on Add Set screen  Capture information from textboxes |
| Process 8.9.4 | Check for blank fields in the textboxes |
| Pseudo Code | On click of “Add Set Button” on Add Set screen  Read through textboxes on screen  IF (Textbox data = ‘null’)  Continue to **ALT Step 7**  ELSE  Continue to **Step 8** |
| ALT  On click of “Add Set Button” on Add Set screen  Display Message box with error message “Please fill all fields”  and an “OK” button  On click “OK Button” on Message box  Display Set screen  Highlight empty textbox with red colour |
| Process 8.9.5 | * Validate Set information |
| Pseudo Code | Validation requirements from the Set Table  IF (Captured information = VALID)  FILTER the Set Table by captured information  IF (RECORD = EXIST)  Display Error Message (“record already exists”)  ELSE  Continue to process 8.9.6  END IF  ELSE  Display Error Message (“Invalid Data Entered”)  END IF |
| Process 8.9.6 | Generate a new Set ID |
| Pseudo Code | Filter the Set Table  IF (Table is empty)  Set\_ID = 000001  Else  Read last saved Set\_ID = Set\_ID + 1  END IF |
| Process 8.9. | Save Set details |
|  | Insert Set ID and Description into Set Table |
|  | VALIDATION |
| Pseudo Code | Set\_Description: only alphabets, Max 250 characters, does not already exist |

8.10 Maintain new Set

|  |  |
| --- | --- |
| Process 8.10.1 | Display and populate Set screen |
| Pseudo code | On click of the “Search Set Button” on the Navigation Bar  Read Data from Set Table  Display table with Set details |
| Process 8.10.2 | Search Set |
| Pseudo code | On click of “Search Set Button” on the Set screen  Invoke Use Case 8.3 “Search Set” |
| Process 8.10.3 | Enable controls on the screens |
| Pseudo code | On Load of Set screen  Button Update Enabled = True  Button Delete Enabled = False  Textbox Description = Enabled |
| [ALT] On Click of the “Delete Button”  Continue to process 8.10.6 |
| Process 8.10.4 | Check for blank fields in the textboxes |
| Pseudo Code | On click of “Add Set Button” on Add Set screen  Read through textboxes on screen  IF (Textbox data = ‘null’)  Continue to **ALT Step 7**  ELSE  Continue to **Step 8** |
| ALT  On click of “Add Set Button” on Add Set screen  Display Message box with error message “Please fill all fields”  and an “OK” button  On click “OK Button” on Message box  Display Set screen  Highlight empty textbox with red colour |
| Process 8.10.5 | Capture Updated Set details |
|  | On Click of “Update Button” on the Maintain Set screen  Capture information from the textboxes |
| [ALT]  On Click “Add Set” on the Maintain Set screen  Invoke Use Case 8.1 “Add Set” |
| [ALT]  On Click of “Back Button” on the Maintain Set Screen  Display Message Box for confirmation  IF Response = YES  Clear all Textboxes  Else Response = NO  Break  END IF |
| Process 8.10.6 | Update Set Table |
| Pseudo Code | Update Set Table where criteria are equal to the criteria of the selected Exercise Plan |
| [ALT]  Delete from the Set Table where criteria are equal to the criteria of the selected Exercise Plan |
| Process 8.10.7 | Notify employee of successful update |
| Pseudo Code | Display Message Box “Successfully updated Set” |
| [ALT]  Display Message Box “Successfully Deleted Set” |

8.12 Maintain new Exercise Body Type

|  |  |
| --- | --- |
| Process 8.12.1 | Display and populate Exercise Body Type screen |
| Pseudo code | On click of the “Search Exercise Body Type Button” on the Navigation Bar  Read Data from Exercise Body Type Table  Display table with Exercise Body Type details |
| Process 8.12.2 | Search Exercise Body Type |
| Pseudo code | On click of “Search Exercise Body Type Button” on the Exercise Body Type screen  Invoke Use Case 8.3 “Search Exercise Body Type” |
| Process 8.12.3 | Enable controls on the screens |
| Pseudo code | On Load of Exercise Body Type screen  Button Update Enabled = True  Button Delete Enabled = False  Textbox Description = Enabled |
| [ALT] On Click of the “Delete Button”  Continue to process 8.12.6 |
| Process 8.12.4 | Check for blank fields in the textboxes |
| Pseudo Code | On click of “Add Exercise Body Type Button” on Add Exercise Body Type screen  Read through textboxes on screen  IF (Textbox data = ‘null’)  Continue to **ALT Step 7**  ELSE  Continue to **Step 8** |
| ALT  On click of “Add Exercise Body Type Button” on Add Exercise Body Type screen  Display Message box with error message “Please fill all fields”  and an “OK” button  On click “OK Button” on Message box  Display Exercise Body Type screen  Highlight empty textbox with red colour |
| Process 8.12.5 | Capture Updated Exercise Body Type details |
|  | On Click of “Update Button” on the Maintain Exercise Body Type screen  Capture information from the textboxes |
| [ALT]  On Click “Add Exercise Body Type” on the Maintain Exercise Body Type screen  Invoke Use Case 8.1 “Add Exercise Body Type” |
| [ALT]  On Click of “Back Button” on the Maintain Exercise Body Type Screen  Display Message Box for confirmation  IF Response = YES  Clear all Textboxes  Else Response = NO  Break  END IF |
| Process 8.12.6 | Update Exercise Body Type Table |
| Pseudo Code | Update Exercise Body Type Table where criteria are equal to the criteria of the selected Exercise Plan |
| [ALT]  Delete from the Exercise Body Type Table where criteria are equal to the criteria of the selected Exercise Plan |
| Process 8.12.7 | Notify employee of successful update |
| Pseudo Code | Display Message Box “Successfully updated Exercise Body Type” |
| [ALT]  Display Message Box “Successfully Deleted Exercise Body Type” |

8.11 Add new Exercise Body Types

|  |  |
| --- | --- |
| Process 8.11.1 | Display and populate exercise plan screen |
| Pseudo code | On click of the “Add Exercise Body Type Button” on the Navigation Bar  Read Data from Exercise Body Type Table  Display table with Exercise Body Type details |
| Process 8.11.2 | Display Exercise Body Type Tab |
| Pseudo code | On click of “Add Exercise Body Type Button” on the Navigation Bar  Read Data from Exercise Body Type Table  Display Data in the Data grid view  Display Exercise Body Type Tab |
| Process 8.11.3 | Capture Exercise Body Type details |
| Pseudo code | On click of “Add Exercise Body Type Button” on Add Exercise Body Type screen  Capture information from textboxes |
| Process 8.11.4 | Check for blank fields in the textboxes |
| Pseudo Code | On click of “Add Exercise Body Type Button” on Add Exercise Body Type screen  Read through textboxes on screen  IF (Textbox data = ‘null’)  Continue to **ALT Step 7**  ELSE  Continue to **Step 8** |
| ALT  On click of “Add Exercise Body Type Button” on Add Exercise Body Type screen  Display Message box with error message “Please fill all fields”  and an “OK” button  On click “OK Button” on Message box  Display Exercise Body Type screen  Highlight empty textbox with red colour |
| Process 8.11.5 | * Validate Exercise Body Type information |
| Pseudo Code | Validation requirements from the Exercise Body Type Table  IF (Captured information = VALID)  FILTER the Exercise Body Type Table by captured information  IF (RECORD = EXIST)  Display Error Message (“record already exists”)  ELSE  Continue to process 8.11.6  END IF  ELSE  Display Error Message (“Invalid Data Entered”)  END IF |
| Process 8.11.6 | Generate a new Exercise Body Type ID |
| Pseudo Code | Filter the Exercise Body Type Table  IF (Table is empty)  Exercise\_Body\_Type\_ID = 000001  Else  Read last saved Exercise\_Body\_Type\_ID = Exercise\_Body\_Type\_ID + 1  END IF |
| Process 8.11.7 | Save Exercise Body Type details |
|  | Insert Exercise Body Type ID and Description into Exercise Body Type Table |
|  | VALIDATION |
| Pseudo Code | Exercise\_Body\_Type\_Description: only alphabets, Max 250 characters, does not already exist |

8.13 Search Exercise Body Types

|  |  |
| --- | --- |
| Process 8.13.1 | Display and populate Exercise Body Type screen |
| Pseudo code | On click of the “Search Exercise Body Type Button” on the Navigation Bar  Read Data from Exercise Body Type Table  Display table with Exercise Body Type details |
| Process 8.13.2 | Display Exercise Body Type Tab |
| Pseudo code | On click of “Search Exercise Body Type Button” on the Navigation Bar  Read Data from Exercise Body Type Table  Populate Textboxes with the data from the Table  Display Data in the Data grid view  Enable Exercise Body Type ID Textbox |
| Process 8.13.3 | Capture Search parameters |
| Pseudo code | On click of “Search Exercise Body Type Button” on Search Exercise Body Type screen  Capture information from textboxes |
| Process 8.13.4 | Check for blank fields in the textboxes |
| Pseudo Code | On click of “Search Exercise Body Type Button” on Search Exercise Body Type screen  Read through textboxes on screen  IF (Textbox data = ‘null’)  Continue to **ALT Step 5**  ELSE  Continue to **Step 6** |
| ALT  On click of “Search Exercise Body Type Button” on Search Exercise Body Type screen  Display Message box with error message “Please fill all fields”  and an “OK” button  On click “OK Button” on Message box  Display Exercise Body Type screen  Highlight empty textbox with red colour |
| Process 8.13.5 | Search Exercise Body Type |
| Pseudo Code | Read from the Exercise Body Type Table  IF (Record count of Results = 0)  [ALT] Display Error Message (“Yielded no results”)  ELSE  Continue to process 8.13.6 |
| Process 8.13.6 | Filter Data Grid View |
| Pseudo Code | Display Exercise Body Type details  Filter Data Grid View by Search Results  Display Data Grid View |

8. Exercises subsystem

8.14 Add new Exercise

|  |  |
| --- | --- |
| Process 8.14.1 | Display and populate Exercise screen |
| Pseudo code | On click of the “Add Exercise Button” on the Navigation Bar  Read Data from Exercise Table  Display table with Exercise details |
| Process 8.14.2 | Display Exercise Tab |
| Pseudo code | On click of “Add Exercise Button” on the Navigation Bar  Read Data from Exercise Table  Display Data in the Data grid view  Display Exercise Tab |
| Process 8.14.3 | Capture Exercise details |
| Pseudo code | On click of “Add Exercise Button” on Add Exercise screen  Capture information from textboxes |
| Process 8.14.4 | Check for blank fields in the textboxes |
| Pseudo Code | On click of “Add Exercise Button” on Add Exercise screen  Read through textboxes on screen  IF (Textbox data = ‘null’)  Continue to **ALT Step 7**  ELSE  Continue to **Step 8** |
| ALT  On click of “Add Exercise Button” on Add Exercise screen  Display Message box with error message “Please fill all fields”  and an “OK” button  On click “OK Button” on Message box  Display Exercise screen  Highlight empty textbox with red colour |
| Process 8.14.5 | * Validate Exercise information |
| Pseudo Code | Validation requirements from the Exercise Table  IF (Captured information = VALID)  FILTER the Exercise Table by captured information  IF (RECORD = EXIST)  Display Error Message (“record already exists”)  ELSE  Continue to process 8.14.6  END IF  ELSE  Display Error Message (“Invalid Data Entered”)  END IF |
| Process 8.14.6 | Generate a new Exercise ID |
| Pseudo Code | Filter the Exercise Table  IF (Table is empty)  Exercise\_ID = 000001  Else  Read last saved Exercise\_ID = Exercise\_ID + 1  END IF |
| Process 8.14. | Save Exercise details |
|  | Insert Exercise ID and Description into Exercise Table |
|  | VALIDATION |
| Pseudo Code | Exercise\_Description: only alphabets, Max 250 characters, does not already exist |

8.15 Maintain new Exercise

|  |  |
| --- | --- |
| Process 8.15.1 | Display and populate Exercise screen |
| Pseudo code | On click of the “Search Exercise Button” on the Navigation Bar  Read Data from Exercise Table  Display table with Exercise details |
| Process 8.15.2 | Search Exercise |
| Pseudo code | On click of “Search Exercise Button” on the Exercise screen  Invoke Use Case 8.3 “Search Exercise” |
| Process 8.15.3 | Enable controls on the screens |
| Pseudo code | On Load of Exercise screen  Button Update Enabled = True  Button Delete Enabled = False  Textbox Description = Enabled |
| [ALT] On Click of the “Delete Button”  Continue to process 8.15.6 |
| Process 8.15.4 | Check for blank fields in the textboxes |
| Pseudo Code | On click of “Add Exercise Button” on Add Exercise screen  Read through textboxes on screen  IF (Textbox data = ‘null’)  Continue to **ALT Step 7**  ELSE  Continue to **Step 8** |
| ALT  On click of “Add Exercise Button” on Add Exercise screen  Display Message box with error message “Please fill all fields”  and an “OK” button  On click “OK Button” on Message box  Display Exercise screen  Highlight empty textbox with red colour |
| Process 8.15.5 | Capture Updated Exercise details |
|  | On Click of “Update Button” on the Maintain Exercise screen  Capture information from the textboxes |
| [ALT]  On Click “Add Exercise” on the Maintain Exercise screen  Invoke Use Case 8.1 “Add Exercise” |
| [ALT]  On Click of “Back Button” on the Maintain Exercise Screen  Display Message Box for confirmation  IF Response = YES  Clear all Textboxes  Else Response = NO  Break  END IF |
| Process 8.15.6 | Update Exercise Table |
| Pseudo Code | Update Exercise Table where criteria are equal to the criteria of the selected Exercise Plan |
| [ALT]  Delete from the Exercise Table where criteria are equal to the criteria of the selected Exercise Plan |
| Process 8.15.7 | Notify employee of successful update |
| Pseudo Code | Display Message Box “Successfully updated Exercise” |
| [ALT]  Display Message Box “Successfully Deleted Exercise” |

8.16 Search Exercises

|  |  |
| --- | --- |
| Process 8.16.1 | Display and populate Exercise screen |
| Pseudo code | On click of the “Search Exercise Button” on the Navigation Bar  Read Data from Exercise Table  Display table with Exercise details |
| Process 8.16.2 | Display Exercise Tab |
| Pseudo code | On click of “Search Exercise Button” on the Navigation Bar  Read Data from Exercise Table  Populate Textboxes with the data from the Table  Display Data in the Data grid view  Enable Exercise ID Textbox |
| Process 8.16.3 | Capture Search parameters |
| Pseudo code | On click of “Search Exercise Button” on Search Exercise screen  Capture information from textboxes |
| Process 8.16.4 | Check for blank fields in the textboxes |
| Pseudo Code | On click of “Search Exercise Button” on Search Exercise screen  Read through textboxes on screen  IF (Textbox data = ‘null’)  Continue to **ALT Step 5**  ELSE  Continue to **Step 6** |
| ALT  On click of “Search Exercise Button” on Search Exercise screen  Display Message box with error message “Please fill all fields”  and an “OK” button  On click “OK Button” on Message box  Display Exercise screen  Highlight empty textbox with red colour |
| Process 8.16.5 | Search Exercise |
| Pseudo Code | Read from the Exercise Table  IF (Record count of Results = 0)  [ALT] Display Error Message (“Yielded no results”)  ELSE  Continue to process 8.16.6 |
| Process 8.16.6 | Filter Data Grid View |
| Pseudo Code | Display Exercise details  Filter Data Grid View by Search Results  Display Data Grid View |

8.17 Add new Workout

|  |  |
| --- | --- |
| Process 8.17.1 | Display and populate Workout screen |
| Pseudo code | On click of the “Add Workout Button” on the Navigation Bar  Read Data from Workout Table  Display table with Workout details |
| Process 8.17.2 | Display Workout Tab |
| Pseudo code | On click of “Add Workout Button” on the Navigation Bar  Read Data from Workout Table  Display Data in the Data grid view  Display Workout Tab |
| Process 8.17.3 | Capture Workout details |
| Pseudo code | On click of “Add Workout Button” on Add Workout screen  Capture information from textboxes |
| Process 8.17.4 | Check for blank fields in the textboxes |
| Pseudo Code | On click of “Add Workout Button” on Add Workout screen  Read through textboxes on screen  IF (Textbox data = ‘null’)  Continue to **ALT Step 7**  ELSE  Continue to **Step 8** |
| ALT  On click of “Add Workout Button” on Add Workout screen  Display Message box with error message “Please fill all fields”  and an “OK” button  On click “OK Button” on Message box  Display Workout screen  Highlight empty textbox with red colour |
| Process 8.17.5 | * Validate Workout information |
| Pseudo Code | Validation requirements from the Workout Table  IF (Captured information = VALID)  FILTER the Workout Table by captured information  IF (RECORD = EXIST)  Display Error Message (“record already exists”)  ELSE  Continue to process 8.17.6  END IF  ELSE  Display Error Message (“Invalid Data Entered”)  END IF |
| Process 8.17.6 | Generate a new Workout ID |
| Pseudo Code | Filter the Workout Table  IF (Table is empty)  Workout\_ID = 000001  Else  Read last saved Workout\_ID = Workout\_ID + 1  END IF |
| Process 8.17. | Save Workout details |
|  | Insert Workout ID and Description into Workout Table |
|  | VALIDATION |
| Pseudo Code | Workout\_Description: only alphabets, Max 250 characters, does not already exist |

8.18 Maintain new Workout

|  |  |
| --- | --- |
| Process 8.18.1 | Display and populate Workout screen |
| Pseudo code | On click of the “Search Workout Button” on the Navigation Bar  Read Data from Workout Table  Display table with Workout details |
| Process 8.18.2 | Search Workout |
| Pseudo code | On click of “Search Workout Button” on the Workout screen  Invoke Use Case 8.3 “Search Exercise” |
| Process 8.18.3 | Enable controls on the screens |
| Pseudo code | On Load of Workout screen  Button Update Enabled = True  Button Delete Enabled = False  Textbox Description = Enabled |
| [ALT] On Click of the “Delete Button”  Continue to process 8.18.6 |
| Process 8.18.4 | Check for blank fields in the textboxes |
| Pseudo Code | On click of “Add Workout Button” on Add Workout screen  Read through textboxes on screen  IF (Textbox data = ‘null’)  Continue to **ALT Step 7**  ELSE  Continue to **Step 8** |
| ALT  On click of “Add Workout Button” on Add Workout screen  Display Message box with error message “Please fill all fields”  and an “OK” button  On click “OK Button” on Message box  Display Workout screen  Highlight empty textbox with red colour |
| Process 8.18.5 | Capture Updated Workout details |
|  | On Click of “Update Button” on the Maintain Workout screen  Capture information from the textboxes |
| [ALT]  On Click “Add Exercise” on the Maintain Workout screen  Invoke Use Case 8.1 “Add Exercise” |
| [ALT]  On Click of “Back Button” on the Maintain Workout Screen  Display Message Box for confirmation  IF Response = YES  Clear all Textboxes  Else Response = NO  Break  END IF |
| Process 8.18.6 | Update Workout Table |
| Pseudo Code | Update Workout Table where criteria are equal to the criteria of the selected Workout Plan |
| [ALT]  Delete from the Workout Table where criteria are equal to the criteria of the selected Workout Plan |
| Process 8.18.7 | Notify employee of successful update |
| Pseudo Code | Display Message Box “Successfully updated Exercise” |
| [ALT]  Display Message Box “Successfully Deleted Exercise” |

8.19 Search Workouts

|  |  |
| --- | --- |
| Process 8.19.1 | Display and populate Workout screen |
| Pseudo code | On click of the “Search Workout Button” on the Navigation Bar  Read Data from Workout Table  Display table with Workout details |
| Process 8.19.2 | Display Workout Tab |
| Pseudo code | On click of “Search Workout Button” on the Navigation Bar  Read Data from Workout Table  Populate Textboxes with the data from the Table  Display Data in the Data grid view  Enable Workout ID Textbox |
| Process 8.19.3 | Capture Search parameters |
| Pseudo code | On click of “Search Workout Button” on Search Workout screen  Capture information from textboxes |
| Process 8.19.4 | Check for blank fields in the textboxes |
| Pseudo Code | On click of “Search Workout Button” on Search Workout screen  Read through textboxes on screen  IF (Textbox data = ‘null’)  Continue to **ALT Step 5**  ELSE  Continue to **Step 6** |
| ALT  On click of “Search Workout Button” on Search Workout screen  Display Message box with error message “Please fill all fields”  and an “OK” button  On click “OK Button” on Message box  Display Workout screen  Highlight empty textbox with red colour |
| Process 8.19.5 | Search Workout |
| Pseudo Code | Read from the Workout Table  IF (Record count of Results = 0)  [ALT] Display Error Message (“Yielded no results”)  ELSE  Continue to process 8.19.6 |
| Process 8.19.6 | Filter Data Grid View |
| Pseudo Code | Display Workout details  Filter Data Grid View by Search Results  Display Data Grid View |

8.20 Add new Exercise Plan

|  |  |
| --- | --- |
| Process 8.20.1 | Display and populate Exercise Plan screen |
| Pseudo code | On click of the “Add Exercise Plan Button” on the Navigation Bar  Read Data from Exercise Plan Table  Display table with Exercise Plan details |
| Process 8.20.2 | Display Exercise Plan Tab |
| Pseudo code | On click of “Add Exercise Plan Button” on the Navigation Bar  Read Data from Exercise Plan Table  Display Data in the Data grid view  Display Exercise Plan Tab |
| Process 8.20.3 | Capture Exercise Plan details |
| Pseudo code | On click of “Add Exercise Plan Button” on Add Exercise Plan screen  Capture information from textboxes |
| Process 8.20.4 | Check for blank fields in the textboxes |
| Pseudo Code | On click of “Add Exercise Plan Button” on Add Exercise Plan screen  Read through textboxes on screen  IF (Textbox data = ‘null’)  Continue to **ALT Step 7**  ELSE  Continue to **Step 8** |
| ALT  On click of “Add Exercise Plan Button” on Add Exercise Plan screen  Display Message box with error message “Please fill all fields”  and an “OK” button  On click “OK Button” on Message box  Display Exercise Plan screen  Highlight empty textbox with red colour |
| Process 8.20.5 | * Validate Exercise Plan information |
| Pseudo Code | Validation requirements from the Exercise Plan Table  IF (Captured information = VALID)  FILTER the Exercise Plan Table by captured information  IF (RECORD = EXIST)  Display Error Message (“record already exists”)  ELSE  Continue to process 8.20.6  END IF  ELSE  Display Error Message (“Invalid Data Entered”)  END IF |
| Process 8.20.6 | Generate a new Exercise Plan ID |
| Pseudo Code | Filter the Exercise Plan Table  IF (Table is empty)  Exercise Plan\_ID = 000001  Else  Read last saved Exercise Plan\_ID = Exercise Plan\_ID + 1  END IF |
| Process 8.20. | Save Exercise Plan details |
|  | Insert Exercise Plan ID and Description into Exercise Plan Table |
|  | VALIDATION |
| Pseudo Code | Exercise Plan\_Description: only alphabets, Max 250 characters, does not already exist |

8.21 Maintain new Exercise Plan

|  |  |
| --- | --- |
| Process 8.21.1 | Display and populate Exercise Plan screen |
| Pseudo code | On click of the “Search Exercise Plan Button” on the Navigation Bar  Read Data from Exercise Plan Table  Display table with Exercise Plan details |
| Process 8.21.2 | Search Exercise Plan |
| Pseudo code | On click of “Search Exercise Plan Button” on the Exercise Plan screen  Invoke Use Case 8.3 “Search Exercise” |
| Process 8.21.3 | Enable controls on the screens |
| Pseudo code | On Load of Exercise Plan screen  Button Update Enabled = True  Button Delete Enabled = False  Textbox Description = Enabled |
| [ALT] On Click of the “Delete Button”  Continue to process 8.21.6 |
| Process 8.21.4 | Check for blank fields in the textboxes |
| Pseudo Code | On click of “Add Exercise Plan Button” on Add Exercise Plan screen  Read through textboxes on screen  IF (Textbox data = ‘null’)  Continue to **ALT Step 7**  ELSE  Continue to **Step 8** |
| ALT  On click of “Add Exercise Plan Button” on Add Exercise Plan screen  Display Message box with error message “Please fill all fields”  and an “OK” button  On click “OK Button” on Message box  Display Exercise Plan screen  Highlight empty textbox with red colour |
| Process 8.21.5 | Capture Updated Exercise Plan details |
|  | On Click of “Update Button” on the Maintain Exercise Plan screen  Capture information from the textboxes |
| [ALT]  On Click “Add Exercise” on the Maintain Exercise Plan screen  Invoke Use Case 8.1 “Add Exercise” |
| [ALT]  On Click of “Back Button” on the Maintain Exercise Plan Screen  Display Message Box for confirmation  IF Response = YES  Clear all Textboxes  Else Response = NO  Break  END IF |
| Process 8.21.6 | Update Exercise Plan Table |
| Pseudo Code | Update Exercise Plan Table where criteria are equal to the criteria of the selected Exercise Plan. |
| [ALT]  Delete from the Exercise Plan Table where criteria are equal to the criteria of the selected Exercise Plan. |
| Process 8.21.7 | Notify employee of successful update |
| Pseudo Code | Display Message Box “Successfully updated Exercise” |
| [ALT]  Display Message Box “Successfully Deleted Exercise” |

8.22 Search Exercise Plans

|  |  |
| --- | --- |
| Process 8.22.1 | Display and populate Exercise Plan screen |
| Pseudo code | On click of the “Search Exercise Plan Button” on the Navigation Bar  Read Data from Exercise Plan Table  Display table with Exercise Plan details |
| Process 8.22.2 | Display Exercise Plan Tab |
| Pseudo code | On click of “Search Exercise Plan Button” on the Navigation Bar  Read Data from Exercise Plan Table  Populate Textboxes with the data from the Table  Display Data in the Data grid view  Enable Exercise Plan ID Textbox |
| Process 8.22.3 | Capture Search parameters |
| Pseudo code | On click of “Search Exercise Plan Button” on Search Exercise Plan screen  Capture information from textboxes |
| Process 8.22.4 | Check for blank fields in the textboxes |
| Pseudo Code | On click of “Search Exercise Plan Button” on Search Exercise Plan screen  Read through textboxes on screen  IF (Textbox data = ‘null’)  Continue to **ALT Step 5**  ELSE  Continue to **Step 6** |
| ALT  On click of “Search Exercise Plan Button” on Search Exercise Plan screen  Display Message box with error message “Please fill all fields”  and an “OK” button  On click “OK Button” on Message box  Display Exercise Plan screen  Highlight empty textbox with red colour |
| Process 8.22.5 | Search Exercise Plan |
| Pseudo Code | Read from the Exercise Plan Table  IF (Record count of Results = 0)  [ALT] Display Error Message (“Yielded no results”)  ELSE  Continue to process 8.22.6 |
| Process 8.22.6 | Filter Data Grid View |
| Pseudo Code | Display Exercise Plan details  Filter Data Grid View by Search Results  Display Data Grid View |

2.6 Conclusion

The second section of deliverable 4 had the process-oriented designs of the proposed system, which held the context diagram, decomposition diagram, a complete set of data flow diagrams for the JessterFIT system, and pseudo code.

**3 Entity Relationship diagram**

**3.1 Introduction**

This section holds the Entity Relationship Diagram of the proposed system database.

**3.2 Entity Relationship Diagram**

Please refer to the Appendix of this document for a larger scale print of the diagram.



**3.3 Attribute Nullability**

The below table indicates the nullable attributes in the ERD.

|  |  |
| --- | --- |
| **Attribute Nullability** | |
| **Attribute Name** | **Table Name** |
| **Prize\_ID** | **Challemge** |
| **Bioform\_ID** | **Bioform\_Challenge** |
| **Injury\_ID** | **Bioform** |
| **Client\_ID** | **Booking** |
| **Subsription\_ID** | **Booking** |

**3.4 Conclusion**

This section holds the Entity Relationship Diagram of the proposed system database.

**4 Size Estimation of Proposed Database**

**4.1 Introduction**

This section contains the size estimation of the proposed database, with each table in the ERD and their expected sizes, multiplied by the expected rows for that table, to get a total size for each table, as well as a 5- year projection of the proposed database size.

**4.2 Size Estimation**

|  |  |
| --- | --- |
| **Exercise\_Body\_Type** | |
| **Entity Size (bytes)** | **25** |
| **Expected Rows** | **12** |
| **Total Size (bytes)** | **300** |

|  |  |
| --- | --- |
| **Exercise\_Type** | |
| **Entity Size (bytes)** | **22** |
| **Expected Rows** | **3** |
| **Total Size (bytes)** | **66** |

|  |  |
| --- | --- |
| **Exercise** | |
| **Entity Size (bytes)** | **300** |
| **Expected Rows** | **48** |
| **Total Size (bytes)** | **14400** |

|  |  |
| --- | --- |
| **Workout** | |
| **Entity Size (bytes)** | **42** |
| **Expected Rows** | **6** |
| **Total Size (bytes)** | **252** |

|  |  |
| --- | --- |
| **Exercise\_Workout\_Plan** | |
| **Entity Size (bytes)** | **20** |
| **Expected Rows** | **6** |
| **Total Size (bytes)** | **120** |

|  |  |
| --- | --- |
| **Set** | |
| **Entity Size (bytes)** | **20** |
| **Expected Rows** | **6** |
| **Total Size (bytes)** | **120** |

|  |  |
| --- | --- |
| **Rep** | |
| **Entity Size (bytes)** | **42** |
| **Expected Rows** | **17** |
| **Total Size (bytes)** | **714** |

|  |  |
| --- | --- |
| **Exercise\_Plan\_Type** | |
| **Entity Size (bytes)** | **20** |
| **Expected Rows** | **3** |
| **Total Size (bytes)** | **60** |

|  |  |
| --- | --- |
| **Exercise\_Plan\_Price** | |
| **Entity Size (bytes)** | **52** |
| **Expected Rows** | **2** |
| **Total Size (bytes)** | **104** |

|  |  |
| --- | --- |
| **Challenge\_Plan** | |
| **Entity Size (bytes)** | **28** |
| **Expected Rows** | **30** |
| **Total Size (bytes)** | **840** |

|  |  |
| --- | --- |
| **Challenge** | |
| **Entity Size (bytes)** | **60** |
| **Expected Rows** | **8** |
| **Total Size (bytes)** | **480** |

|  |  |
| --- | --- |
| **Challenge\_Status** | |
| **Entity Size (bytes)** | **22** |
| **Expected Rows** | **3** |
| **Total Size (bytes)** | **66** |

|  |  |
| --- | --- |
| **Challenge\_Detail** | |
| **Entity Size (bytes)** | **70** |
| **Expected Rows** | **8** |
| **Total Size (bytes)** | **560** |

|  |  |
| --- | --- |
| **Prize** | |
| **Entity Size (bytes)** | **260** |
| **Expected Rows** | **8** |
| **Total Size (bytes)** | **2080** |

|  |  |
| --- | --- |
| **Client\_Challenge** | |
| **Entity Size (bytes)** | **30** |
| **Expected Rows** | **84** |
| **Total Size (bytes)** | **2520** |

|  |  |
| --- | --- |
| **Company** | |
| **Entity Size (bytes)** | **206** |
| **Expected Rows** | **1** |
| **Total Size (bytes)** | **206** |

|  |  |
| --- | --- |
| **Payment\_Status** | |
| **Entity Size (bytes)** | **22** |
| **Expected Rows** | **2** |
| **Total Size (bytes)** | **44** |

|  |  |
| --- | --- |
| **Payment** | |
| **Entity Size (bytes)** | **50** |
| **Expected Rows** | **222** |
| **Total Size (bytes)** | **11100** |

|  |  |
| --- | --- |
| **Client** | |
| **Entity Size (bytes)** | **55** |
| **Expected Rows** | **150** |
| **Total Size (bytes)** | **8250** |

|  |  |
| --- | --- |
| **Bioform** | |
| **Entity Size (bytes)** | **70** |
| **Expected Rows** | **222** |
| **Total Size (bytes)** | **15540** |

|  |  |
| --- | --- |
| **Bioform\_Challenge** | |
| **Entity Size (bytes)** | **70** |
| **Expected Rows** | **84** |
| **Total Size (bytes)** | **5880** |

|  |  |
| --- | --- |
| **Challenge\_Form** | |
| **Entity Size (bytes)** | **42** |
| **Expected Rows** | **672** |
| **Total Size (bytes)** | **28224** |

|  |  |
| --- | --- |
| **Gender** | |
| **Entity Size (bytes)** | **16** |
| **Expected Rows** | **3** |
| **Total Size (bytes)** | **48** |

|  |  |
| --- | --- |
| **Injury** | |
| **Entity Size (bytes)** | **260** |
| **Expected Rows** | **53** |
| **Total Size (bytes)** | **13780** |

|  |  |
| --- | --- |
| **Subscription\_Detail** | |
| **Entity Size (bytes)** | **40** |
| **Expected Rows** | **1** |
| **Total Size (bytes)** | **40** |

|  |  |
| --- | --- |
| **Subscription** | |
| **Entity Size (bytes)** | **48** |
| **Expected Rows** | **4** |
| **Total Size (bytes)** | **192** |

|  |  |
| --- | --- |
| **Subscription\_Length** | |
| **Entity Size (bytes)** | **18** |
| **Expected Rows** | **4** |
| **Total Size (bytes)** | **72** |

|  |  |
| --- | --- |
| **Subscription\_Status** | |
| **Entity Size (bytes)** | **22** |
| **Expected Rows** | **2** |
| **Total Size (bytes)** | **44** |

|  |  |
| --- | --- |
| **User** | |
| **Entity Size (bytes)** | **82** |
| **Expected Rows** | **171** |
| **Total Size (bytes)** | **14022** |

|  |  |
| --- | --- |
| **User\_Status** | |
| **Entity Size (bytes)** | **20** |
| **Expected Rows** | **2** |
| **Total Size (bytes)** | **40** |

|  |  |
| --- | --- |
| **Access\_Level** | |
| **Entity Size (bytes)** | **25** |
| **Expected Rows** | **4** |
| **Total Size (bytes)** | **100** |

|  |  |
| --- | --- |
| **Access\_Functionality** | |
| **Entity Size (bytes)** | **20** |
| **Expected Rows** | **12** |
| **Total Size (bytes)** | **240** |

|  |  |
| --- | --- |
| **Functionality** | |
| **Entity Size (bytes)** | **42** |
| **Expected Rows** | **8** |
| **Total Size (bytes)** | **336** |

|  |  |
| --- | --- |
| **Employee** | |
| **Entity Size (bytes)** | **119** |
| **Expected Rows** | **31** |
| **Total Size (bytes)** | **3689** |

|  |  |
| --- | --- |
| **Employee\_Type** | |
| **Entity Size (bytes)** | **22** |
| **Expected Rows** | **3** |
| **Total Size (bytes)** | **66** |

|  |  |
| --- | --- |
| **Booking** | |
| **Entity Size (bytes)** | **70** |
| **Expected Rows** | **304** |
| **Total Size (bytes)** | **21280** |

|  |  |
| --- | --- |
| **Booking\_Type** | |
| **Entity Size (bytes)** | **22** |
| **Expected Rows** | **2** |
| **Total Size (bytes)** | **44** |

|  |  |
| --- | --- |
| **Booking\_Slot** | |
| **Entity Size (bytes)** | **72** |
| **Expected Rows** | **10292** |
| **Total Size (bytes)** | **741024** |

|  |  |
| --- | --- |
| **Total Size of Database** | |
| **Size (bytes)** | **902355** |

**4.3 Five Year Project Plan**

The following projection is based on information received from the client (Specialized Growers) and their current data, as well as their goals for the foreseeable future.

|  |  |  |  |
| --- | --- | --- | --- |
| **5-Year Project of Proposed Database Size** | | | |
| **Year:** | **Expected Marginal Growth** | **Expected Total Growth** | **Expected Size (bytes)** |
| **Current** | **N/A** | **N/A** | **902355** |
| **2020** | **20%** | **120%** | **1082826** |
| **2021** | **45%** | **165%** | **1488885,75** |
| **2022** | **30%** | **195%** | **1759592,25** |
| **2023** | **10%** | **205%** | **1849827,75** |
| **2024** | **5%** | **210%** | **1894945,5** |
| **Overall Total (MB)** | | **925,2663574** | |

**4.4 Conclusion**

This section contained the size estimation of the proposed database, with each table in the ERD and their expected sizes, multiplied by the expected rows for that table, to get a total size for each table, as well as a 5- year projection of the proposed database size

5. Input Design

5.1 Introduction

This section holds the detailed descriptions of the screens, web pages and menus in the system and mobile app, with their controls and functionality. The screens are displayed from the prototype and listed below.

5.2 Listed Screens

Please refer to appendix.

5.3 Conclusion

This section held the detailed descriptions of the screens, web pages and menus in the system and mobile app, with their controls and functionality.

6. Output Design

6.1 Introduction

This section contains descriptions of the purpose of each report, when it will run, the possible entities involved and the layout of each report.

6.2 Design Principles

6.2.1 Report Designs

6.2.1.1 Exercise Plan Report

.

|  |  |
| --- | --- |
| **General Details** | |
| Report No. | 5.1 |
| Report Name | Generate Exercise Plan Report |
| Short Description | Summarizes exercise plan sales for owner to revue Sales Revenues per day of the selected period. Also shows the amount of each exercise plan type that was bought. |
| Frequency | Ad hoc |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Selection Criteria | | | | | |
| Criteria | Default | Type | All Selection | Multiple selections | Values |
| Any period that has been concluded in the business environment. | N/A | Short DateTime (Textbox) | Y | Y | Start Date and  End Date |

| Report Fields | | | | |
| --- | --- | --- | --- | --- |
| Field Name | Format | Max. Length | Justified | Source |
| Report Date (Start Date) | Short Date (DD/MM/YYYY), bold | 10 | Center | Start Date selection |
| Report Date (End Date) | Short Date (DD/MM/YYYY), bold | 10 | Left | End Date Selection |
| Exercise Plan Name | Text | 10 | Center | **Exercise\_Plan\_Name** |
| Price | Numeric | 10 | Center | **Exercise\_Plan\_Price** |
| Date | Short Date (DD/MM/YYYY) | 10 | Center | **Payment\_Date** |
| Sale amount | numeric | 10 | Center | **Payment\_Amount** |
| Grand Total | Numeric | 10 | Right | Price **++** |
| Top Seller | Text | 10 | Right | Highest ordered exercise plan |
| Worst Seller | Text | 10 | Right | Lowest ordered exercise plan |

Report Layout

|  |  |  |  |
| --- | --- | --- | --- |
| Exercise Plan Name | Price | Date | Sale Amount |
| XXXXXXXXXX | XXXXXXXXXX | XXXXXXXXXX | XXXXXXXXXX |
| XXXXXXXXXX | XXXXXXXXXX | XXXXXXXXXX | XXXXXXXXXX |
| XXXXXXXXXX | XXXXXXXXXX | XXXXXXXXXX | XXXXXXXXXX |
| XXXXXXXXXX | XXXXXXXXXX | XXXXXXXXXX | XXXXXXXXXX |
| XXXXXXXXXX | XXXXXXXXXX | XXXXXXXXXX | XXXXXXXXXX |

Grand Total: R XXXXXXX

Exercise Plan Report

Start Date: XXXXXXXXXX

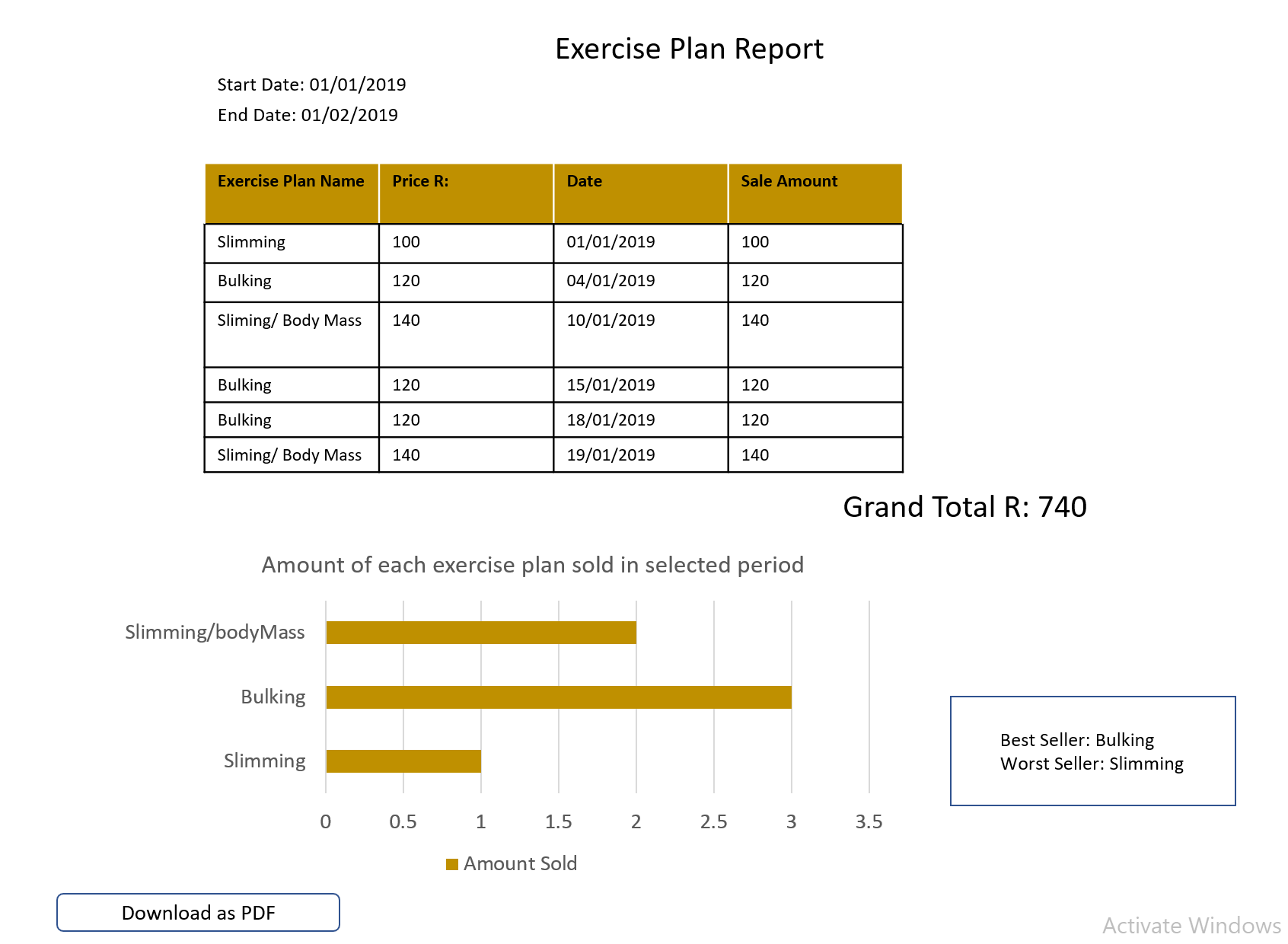
End Date: XXXXXXXXXX

Example Report:

Top Seller: XXXXXXXXXX

Worst Seller: XXXXXXXXXX

Download as PDF



Graph Summary and Purpose

Bar Graph allows the owner to compare the amount of each exercise program bought. Can serve to help with promotional tactics to see if marketing and advertising on Instagram has proven to be successful in attracting new customers to purchase programs and can help owner decide which programs need altering to attract new clients.

6.2.1.2 Subscription Report

|  |  |
| --- | --- |
| General Details | |
| Report No. | 5.2 |
| Report Name | Generate Subscription Report |
| Short Description | Summarizes subscriptions purchased for owner to revue. Sales Revenues will be isolated over a certain period of specified by the owner. Showing grand total of revenues from the selected period. |
| Frequency | Ad hoc |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Selection Criteria | | | | | |
| Criteria | Default | Type | All Selection | Multiple selections | Values |
| Any period that has been concluded in the business environment. | N/A | Short DateTime (Textbox) | Y | Y | Start Date and  End Date |

| Report Fields | | | | |
| --- | --- | --- | --- | --- |
| Field Name | Format | Max. Length | Justified | Source |
| Report Date (Start Date) | Short Date (DD/MM/YYYY), bold | 10 | Left | Start Date selection |
| Report Date (End Date) | Short Date (DD/MM/YYYY), bold | 10 | Left | End Date Selection |
| Client ID | Text | 5 | Center | **Client\_ID** |
| Client Name | Text | 10 | Center | **Client\_Name** |
| Subscription Description | Text | 15 | Center | **Subscription\_Length** |
| Subscription Payment Date | Short Date (DD/MM/YYYY) | 10 | Center | **Payment Date** |
| Subscription Payment Amount | Numeric | 10 | Center | **Payment Amount** |
| Grand Total | Numeric | 10 | Right | **Payment-Amount ++** |
| Top Seller | Text | 10 | Right | Highest bought subscription description |
| Worst Seller | Text | 10 | Right | Lowest bought subscription description. |

Report Layout

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Client ID | Client Name | Subscription Description | Subscription Payment Date | Subscription Payment Amount |
| XXXXX | XXXXXXXXXX | XXXXXXXXXXXXXXX | XXXXXXXXXX | XXXXXXXXXX |
| XXXXX | XXXXXXXXXX | XXXXXXXXXXXXXXX | XXXXXXXXXX | XXXXXXXXXX |
| XXXXX | XXXXXXXXXX | XXXXXXXXXXXXXXX | XXXXXXXXXX | XXXXXXXXXX |
| XXXXX | XXXXXXXXXX | XXXXXXXXXXXXXXX | XXXXXXXXXX | XXXXXXXXXX |
| XXXXX | XXXXXXXXXX | XXXXXXXXXXXXXXX | XXXXXXXXXX | XXXXXXXXXX |

Grand Total: R XXXXXXX

Subscription Report

Start Date: XXXXXXXXXX

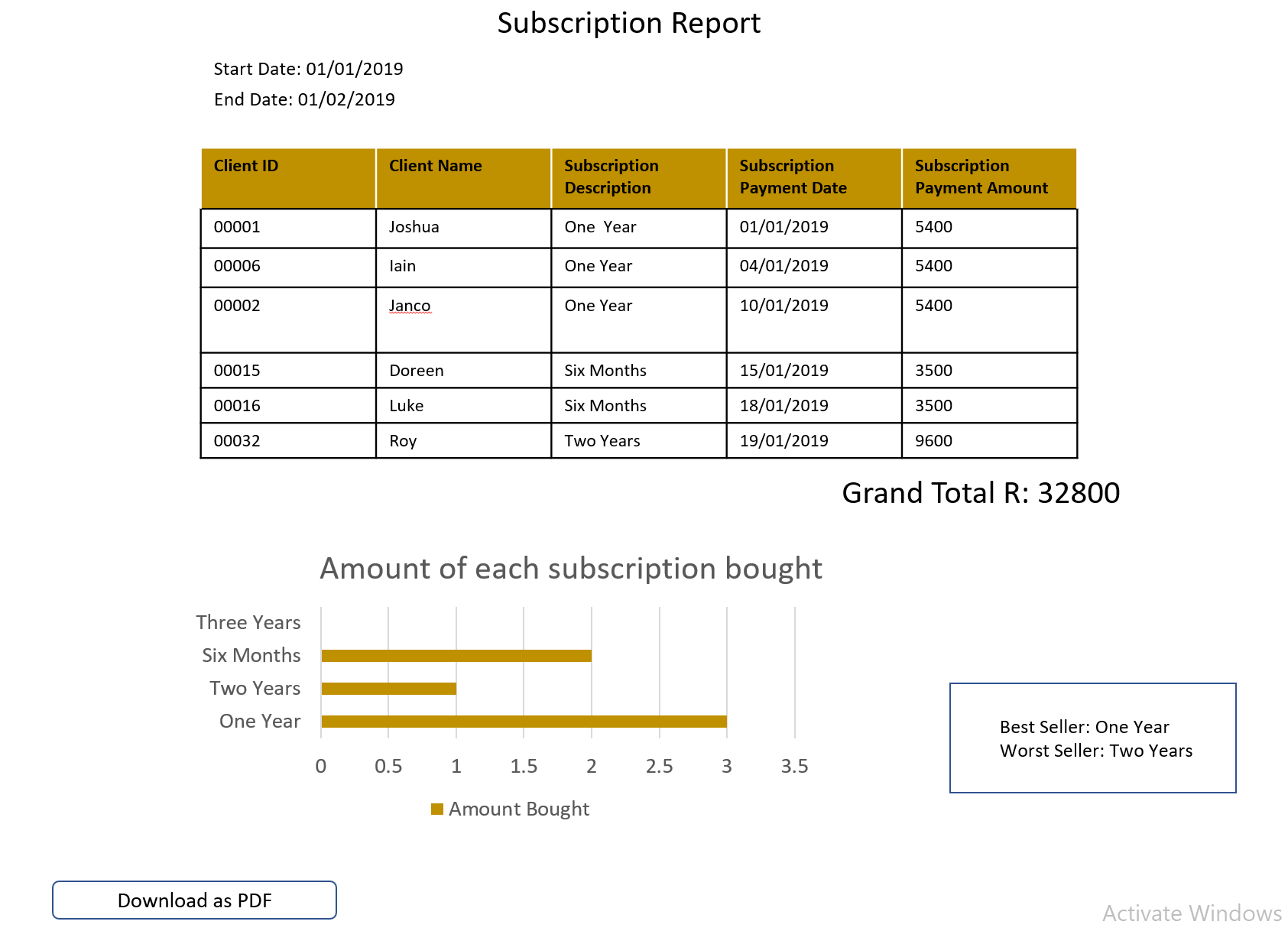
End Date: XXXXXXXXXX

Top Seller: XXXXXXXXXX

Worst Seller: XXXXXXXXXX

Download as PDF

Example Report



Graph Summary and Purpose

Bar Graph allows the owner to compare the amount of each subscription bought. Can serve to help with promotional tactics to see if marketing and advertising on Instagram has proven to be successful in attracting new customers to purchase subscription packages. Also providing guidance on which programs to advertise and promote.

6.2.1.3 Challenge Entry Report

|  |  |
| --- | --- |
| General Details | |
| Report No. | 5.3 |
| Report Name | Generate Challenge Booking Report |
| Short Description | Summarizes challenge purchased/entered by clients for owner to revue. Sales Revenue will be isolated over a certain period of specified by the owner. Showing grand total of revenues from the selected period for challenges and number of individuals registered for a challenge. |
| Frequency | Ad hoc |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Selection Criteria | | | | | |
| Criteria | Default | Type | All Selection | Multiple selections | Values |
| Any currently running challenge or previously running challenge. | N/A | Drop down list of running and previous challenges. | Y | Y | Running and previous challenges |

| Report Fields | | | | |
| --- | --- | --- | --- | --- |
| Field Name | Format | Max. Length | Justified | Source |
| Challenge Name | Text | 15 | Center | **Challenge\_Description** |
| Challenge Running Period (Start Date/ End Date) | Short Date (DD/MM/YYYY), bold | 10 | Center | **Challenge\_Start\_Date**  And  **Challenge\_End\_Date** |
| Client ID | Text | 5 | Center | **Client\_ID** |
| Client Name | Text | 10 | Center | **Client\_Name** |
| Challenge Payment Date | Short Date (DD/MM/YYYY) | 10 | Center | **Payment\_Date** |
| Challenge Payment Amount | Numeric | 10 | Center | **Payment\_Amount** |
| Grand Total | Numeric | 10 | Right | **Payment\_Amount ++** |
| Number of entries | Text | 10 | Right | Number of payments made |

Report Layout

|  |  |  |  |
| --- | --- | --- | --- |
| Client ID | Client Name | Challenge Payment Date | Challenge Payment Amount |
| XXXXX | XXXXXXXXXX | XXXXXXXXXX | XXXXXXXXXX |
| XXXXX | XXXXXXXXXX | XXXXXXXXXX | XXXXXXXXXX |
| XXXXX | XXXXXXXXXX | XXXXXXXXXX | XXXXXXXXXX |
| XXXXX | XXXXXXXXXX | XXXXXXXXXX | XXXXXXXXXX |
| XXXXX | XXXXXXXXXX | XXXXXXXXXX | XXXXXXXXXX |

Grand Total: R XXXXXXX

XXXXXXXXXXXXXXX

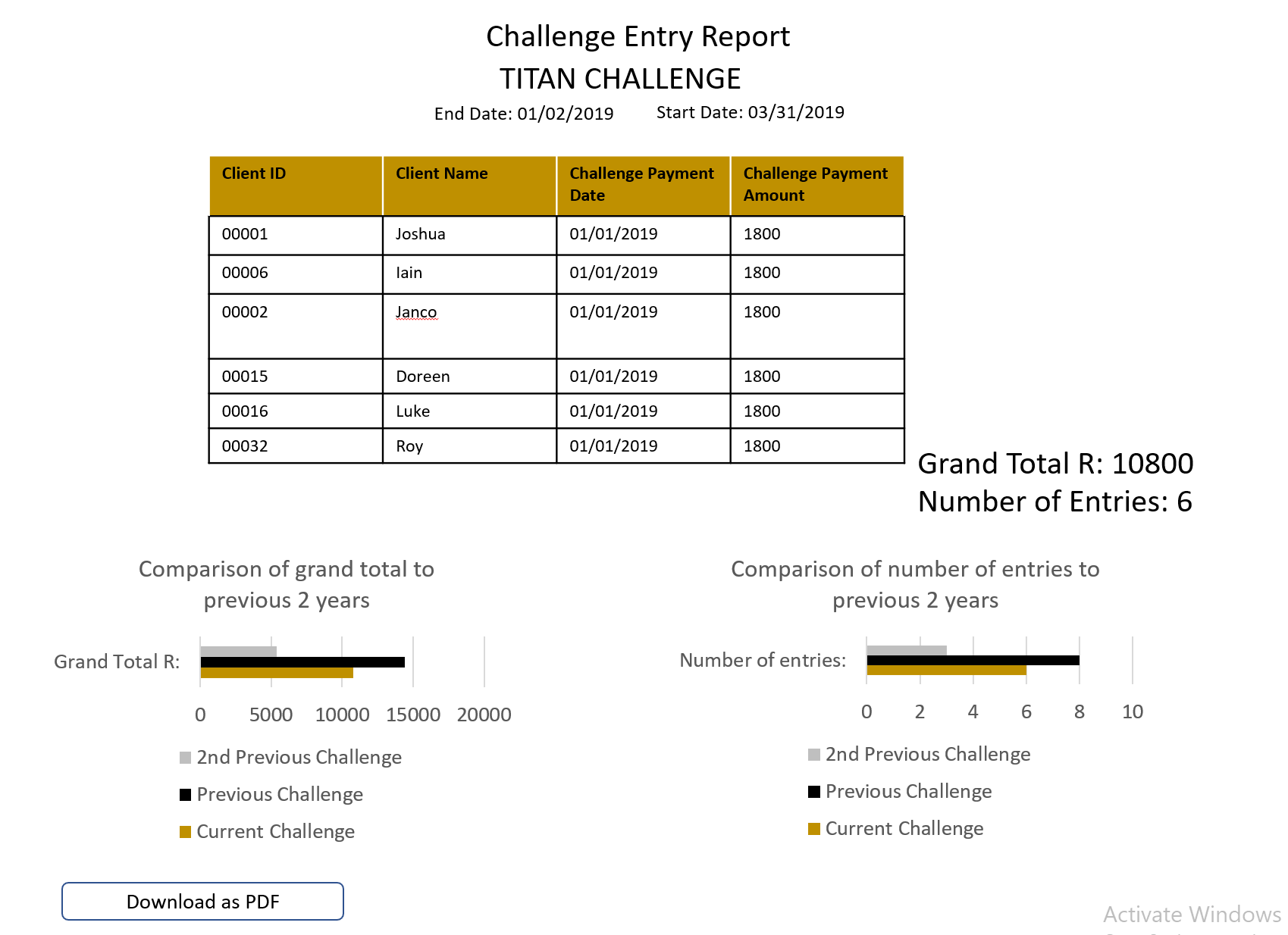
Start Date: XXXXXXXXXX End Date: XXXXXXXXXX

Challenge Entry Report

Number of entries: XXXXXXXXXX

Download as PDF

Report Example



Graphs summary and purpose

Bar Graphs allows the owner to compare the current challenges total revenue and number of entries to that of the previous 2 challenges. Can serve to help with promotional tactics to see if marketing and advertising on Instagram has proven to be successful in attracting new customers to enter challenges.

6.2.1.4 Consultation booking details report

|  |  |
| --- | --- |
| General Details | |
| Report No. | 5.4 |
| Report Name | Generate Consultation Booking Details Report |
| Short Description | Summarizes client details regarding challenge checkup they have booked. Also serves as proof to the booking and must be presented at checkup. |
| Frequency | Ad hoc |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Selection Criteria | | | | | |
| Criteria | Default | Type | All Selection | Multiple selections | Values |
| Booking slot of preference booked from available booking slots. | N/A | Short DateTime (Textbox) | Y | Y | Slot Booked |

| Report Fields | | | | |
| --- | --- | --- | --- | --- |
| Field Name | Format | Max. Length | Justified | Source |
| Report Date (Date booked) | Short Date (DD/MM/YYYY) | 10 | Center | Current Date |
| Client ID | Text | 5 | Left | **Client\_ID** |
| Client Name | Text | 10 | Left | **Client\_Name** |
| Gender | Text | 5 | Left | **Gender\_Description** |
| Date of Birth | Short Date (DD/MM/YYYY) | 10 | Left | **Date\_of\_Birth** |
| Date | Short Date (DD/MM/YYYY) | 15 | Left | **Booking\_Slot\_Date** |
| Time | Time  (HH/MM) | 4 | Left | **Booking\_Slot\_Start** |
| Client Height: m | Numeric | 5 | Left | **Client\_Height** |
| Client Weight:  Kg | Numeric | 5 | Left | **Client\_Weight** |
| Injuries | Text | 25 | Left | **Injuries** |

Report Layout

Consultation Booking Details Report

XXXXXXXXXXXXXXXXXXXXXXXXX

**Injuries to pay attention to:**

Client Height (m): XXXXX

Client Weight (kg): XXXXX

**Bio Form Details:**

**Booking Slot Details**

**Client Details:**

Report Date: XXXXXXXXXX

Client ID: XXXXX

Client Name: XXXXXXXXXX

Gender: XXXXX

Date of Birth: XXXXXXXXXX

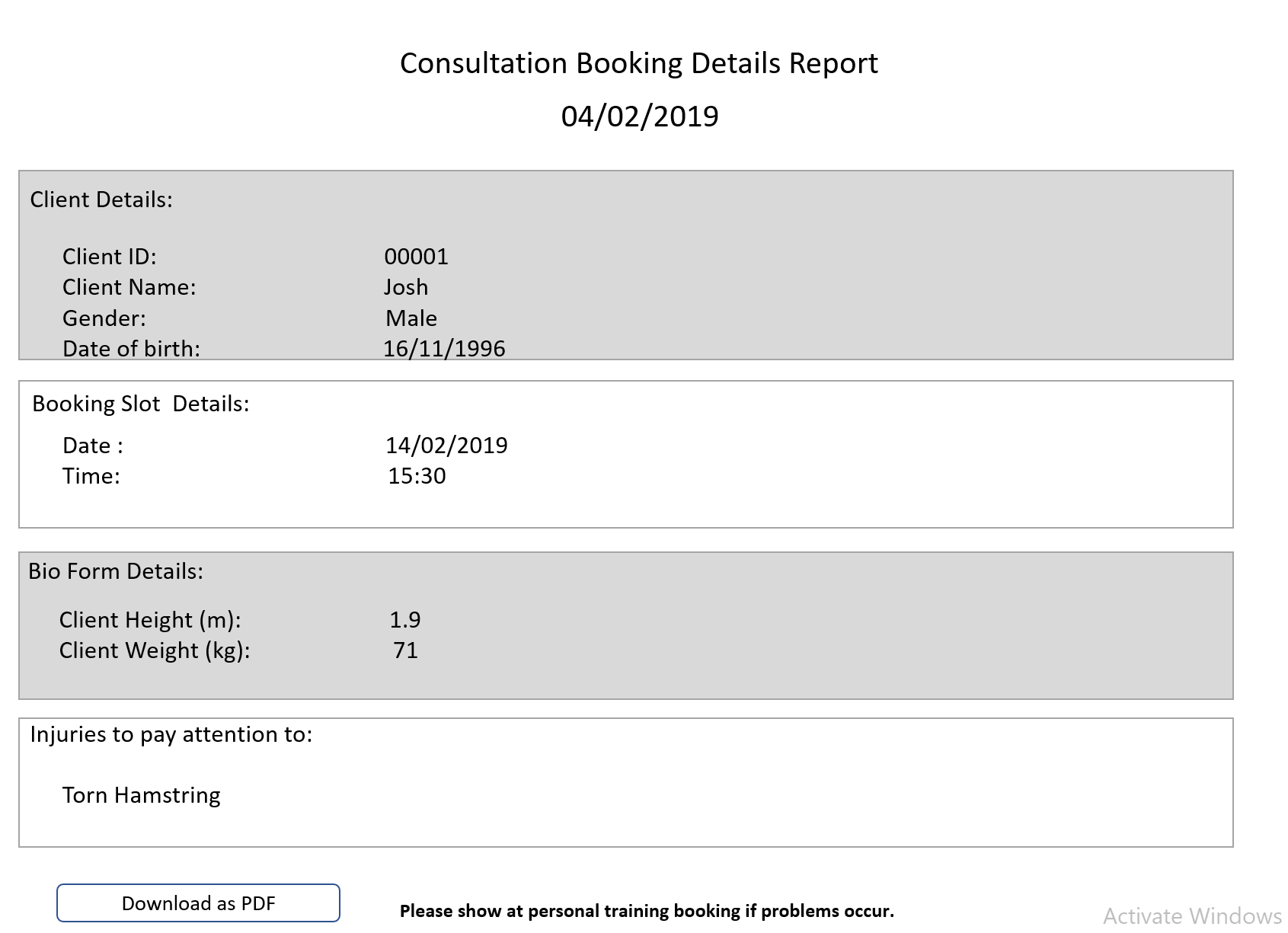
Date: XXXXXXXXXX

Time: XXXX

**Please show at consultation booking if problems occur**

Report Example

Download as PDF



6.2.1.5 Personal training booking details report

|  |  |
| --- | --- |
| General Details | |
| Report No. | 5.5 |
| Report Name | Generate Personal Training Booking Details Report |
| Short Description | Summarizes client details regarding personal they have booked. Also serves as proof to the booking. |
| Frequency | Ad hoc |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Selection Criteria | | | | | |
| Criteria | Default | Type | All Selection | Multiple selections | Values |
| Booking slot of preference booked from available booking slots. | N/A | Short DateTime (Textbox) | Y | Y | Slot Booked |

| Report Fields | | | | |
| --- | --- | --- | --- | --- |
| Field Name | Format | Max. Length | Justified | Source |
| Report Date (Date booked) | Short Date (DD/MM/YYYY) | 10 | Center | Current Date |
| Client ID | Text | 5 | Left | **Client\_ID** |
| Client Name | Text | 10 | Left | **Client\_Name** |
| Date of Birth | Short Date (DD/MM/YYYY) | 10 | Left | **Date\_of\_Birth** |
| Date | Short Date (DD/MM/YYYY) | 15 | Left | **Booking\_Slot\_Date** |
| Time | Time  (HH/MM) | 4 | Left | **Booking\_Slot\_Start** |
| Client Height: m | Numeric | 5 | Left | **Client\_Height** |
| Client Weight:  Kg | Numeric | 5 | Left | **Client\_Weight** |
| Injuries | Text | 25 | Left | **Injuries** |

Report Layout

Personal Training Booking Details Report

Report Date: XXXXXXXXXX

**Client Details:**

Client ID: XXXXX

Client Name: XXXXXXXXXX

Date of Birth: XXXXXXXXXX

**Booking Slot Details**

Date: XXXXXXXXXX

Time: XXXX

**Bio Form Details:**

Client Height (m): XXXXX

Client Weight (kg): XXXXX

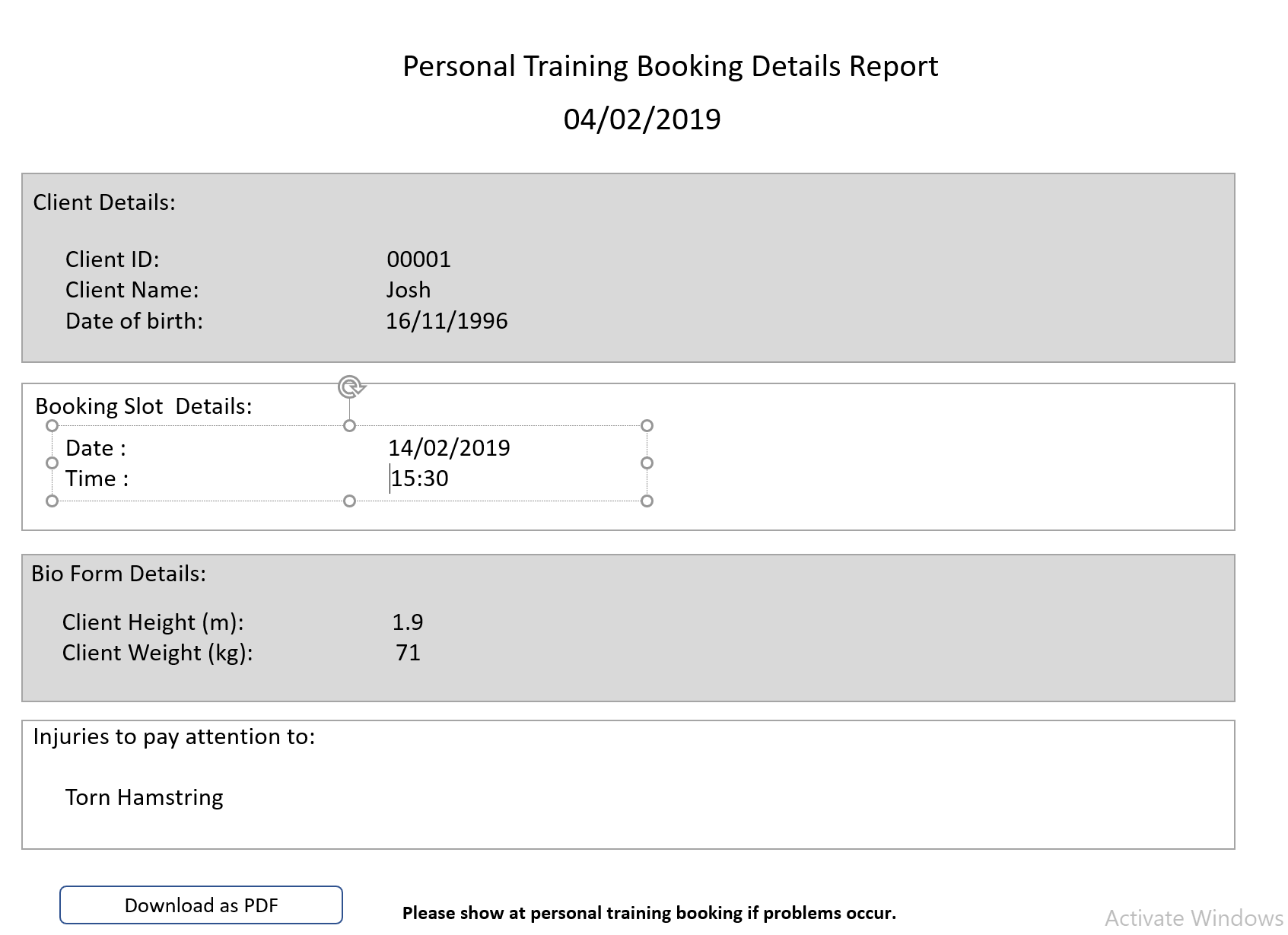
**Injuries to pay attention to:**

Report Example

XXXXXXXXXXXXXXXXXXXXXXXXX

**Please show at personal training booking if problems occur with booking.**

Download as PDF



6.3 Conclusion

This section contained descriptions of the purpose of each report, when it will run, the possible entities involved and the layout of each report.

7. Other Interfaces

7.1 Introduction

This section contains the external application interface technology descriptions required for the system and its proposed functionality.

7.2 Input device interface technical description

7.3 Conclusion

This section contains the external application interface technology descriptions required for the system and its proposed functionality.

8. Test Specifications

## 8.1) Introduction

The Test Specifications section will illustrate and in-depth system test of six core use cases that form the primary function, “Heart”, of our system JessterFIT.

JessterFIT Performance’s primary business function is the signing up for a subscription, Challenge registration, purchasing an exercise plan and making a consultation for a challenge, Namely:

* 3.4 Sign Up for Subscription
* 3.5 Register for challenge
* 3.7 Purchase Exercise Plan
* 3.9 Confirm Payment

Test Scenarios & Test Data

|  |  |
| --- | --- |
| **Test Scenario:** | |
| Tester Name: |  |
| Tester Position: |  |
| Description: | The Use Case describes the event that a client would like to sign up for a subscription. The client will select to sign up and either select a 6-months, 1-year, 2-year or 3-year subscription. Once the client has selected the length of their subscription, they will add it to the basket and be informed that they have to pay for the basket before they logout otherwise it will be removed. Once confirmed by the owner the subscription will begin |
| Precondition: | The client must be logged into the system and must not yet be registered for a subscription to be able to sign up for a subscription. |
|  |  |

|  |  |  |
| --- | --- | --- |
| Test Case: Check response when Signing up for a challenge | | Pass/Fail |
| Test Step 1: | Display Subscription sign up pop up |  |
| Correct Test Data: |  |  |
| Alternative Correct Test Data: |  |  |
| Incorrect Test Data: |  |  |
| Alternative Incorrect Test Data: |  |  |
| Expected Results: |  |  |
| Actual Results: |  |  |

|  |  |
| --- | --- |
| **Test Scenario:** | |
| Tester Name: |  |
| Tester Position: |  |
| Description: | This use case describes the process whereby a client purchases an exercise plan off the online JessterFIT store. The client selects to check out the basket item, adds payment details and then the relevant details are stored by the system in the payment table. |
| Precondition: | The client must be registered and logged in. |
|  |  |

|  |  |  |
| --- | --- | --- |
| Test Case: Check response on a client purchasing an exercise plan using the online store | | Pass/Fail |
| Test Step 1: | Login to JessterFIT Webpage |  |
| Correct Test Data: |  |  |
| Alternative Correct Test Data: |  |  |
| Incorrect Test Data: |  |  |
| Alternative Incorrect Test Data: |  |  |
| Expected Results: | Check the Access Level of the Client( Access\_Level\_ID from the Access\_Level table).  Identify that a client is logged in based on User\_ID. Navigate User to the Client “Home” screen. |  |
| Actual Results: |  |  |
| Test Step 2: | Hover mouse over “Basket” icon on home screen |  |
| Correct Test Data: |  |  |
| Alternative Correct Test Data: |  |  |
| Incorrect Test Data: |  |  |
| Alternative Incorrect Test Data: |  |  |
| Expected Results: | Display Navigation Bar Items:   * Exercise Plans * Subscriptions * Challenges |  |
| Actual Results: |  |  |
| Test Step 3: | Option A: Click on the Exercise Plans  Option B: Click on the Subscriptions  Option C: Click on the Challenges |  |
| Correct Test Data: |  |  |
| Alternative Correct Test Data: |  |  |
| Incorrect Test Data: |  |  |
| Alternative Incorrect Test Data: |  |  |
| Expected Results: | Option A (Exercise Plan):  Display the following Exercise Plan Details  Option B (Subscription):  Option C (Exercise Plan): |  |
| Actual Results: |  |  |
| Test Step 4: | Click the “Add to Cart” button |  |
| Correct Test Data: |  |  |
| Alternative Correct Test Data: |  |  |
| Incorrect Test Data: |  |  |
| Alternative Incorrect Test Data: |  |  |
| Expected Results: | Display a message saying “Your item has been added to the cart successfully”  Update the number of items in the cart and total amount |  |
| Actual Results: |  |  |
| Test Step 5: | Click the “Cart” icon |  |
| Correct Test Data: |  |  |
| Alternative Correct Test Data: |  |  |
| Incorrect Test Data: |  |  |
| Alternative Incorrect Test Data: |  |  |
| Expected Results: | Navigate user to the “Cart” screen containing all the items they added to the cart.  “Cart” Screen Displays:   * Item Name * Item Price * Total cost * Total number of items in the cart |  |
| Actual Results: |  |  |
| Test Step 6: | Click the “Checkout Basket” button |  |
| Correct Test Data: |  |  |
| Alternative Correct Test Data: |  |  |
| Incorrect Test Data: |  |  |
| Alternative Incorrect Test Data: |  |  |
| Expected Results: | Generate a Payment\_ID using the last Payment\_ID in the Payment Table.  Payment screen displays:   * Basket Amount * Client Details * JessterFIT Banking Details |  |
| Actual Results: |  |  |
| Test Step 7: | Click the “Pay Now” button. |  |
| Correct Test Data: |  |  |
| Alternative Correct Test Data: |  |  |
| Incorrect Test Data: |  |  |
| Alternative Incorrect Test Data: |  |  |
| Expected Results: | Generate and store a new Payment entry (Payment Table)  Payment screen displays:   * Payment\_ID * Amount * Status\_ID * Date * Client\_ID   Navigate Client back to the “Client Home” Screen and display a message thanking them for their purcase.  Sent notification to the manager on their “Dashboard” screen using the ‘Unconfirmed payments icon’ Button. |  |
| Actual Results: |  |  |
|  |  |  |

|  |  |
| --- | --- |
| **Test Scenario:** | |
| Tester Name: |  |
| Tester Position: |  |
| Description: | This use case describes the process whereby a manager confirms the payment of an online sales order. The manager also ensures all delivery preparation can begin here and the system informs the client when the sale payment has been confirmed. |
| Precondition: | The client must have paid for a basket item online. |

|  |  |  |
| --- | --- | --- |
| Test Case: Check response on confirming online sale payment. | | Pass/Fail |
| Test Step 1: | Login to JessterFIT Webpage |  |
| Correct Test Data: |  |  |
| Alternative Correct Test Data: |  |  |
| Incorrect Test Data: |  |  |
| Alternative Incorrect Test Data: |  |  |
| Expected Results: | Check the functionality for Access\_Level\_ID.  Identify whether a Manager is logged in based on the Access\_Level\_ID. Navigate the user to the Manager”Dashboard” screen displaying payment details with a description “Unconfirmed” description and a confirm Payment button. |  |
| Actual Results: |  |  |
| Test Step 2: | Option 1: Click on the “confirm Payment” button.  Option 2: Click on the “Close” button. |  |
| Correct Test Data: |  |  |
| Alternative Correct Test Data: |  |  |
| Incorrect Test Data: |  |  |
| Alternative Incorrect Test Data: |  |  |
| Expected Results: | Option 1:  Navigate the user to the “Confirm Payments” screen which displays the details of all the Subscriptions, Challenges and Exercise plans that were paid for via EFT and saved in the Payment table.  Table columns:   * Client\_ID * Payment\_ID * Date * Amount * Confirm Payment Button   Option 2:  Navigate user back to the Home page. |  |
| Actual Results: |  |  |
| Test Step 3: | Option 1 continues: Clicks on the Confirm Payment button in a specific payment column. |  |
| Correct Test Data: |  |  |
| Alternative Correct Test Data: |  |  |
| Incorrect Test Data: |  |  |
| Alternative Incorrect Test Data: |  |  |
| Expected Results: | Option A (Subscription Payment):  System verifies that amount paid is enough to pay for the subscription purchased.  Subscription\_Status\_ID is changed from pending payment to Currently paid in the Subscription\_Status Table.  Option B (Challenge Payment):  System verifies that amount paid is enough to pay for the challenge purchased.  Challenge\_Detail\_ID is changed from pending payment to Currently paid in the Challenge Detail Table.  Option C (Exercise Plan Payment):  System verifies that amount paid is enough to pay for the Exercise Plan purchased.  Exercise\_Plan\_Payment\_ID is changed from pending payment to Currently paid in the Exercise\_Plan\_Payment Table. |  |
| Actual Results: |  |  |
|  |  |  |
| Test Step 4: | Search Client Details |  |
| Correct Test Data: |  |  |
| Alternative Correct Test Data: |  |  |
| Incorrect Test Data: |  |  |
| Alternative Incorrect Test Data: |  |  |
| Expected Results: | Option A (Subscription Payment):  Enable Client access rights for booking by updating User\_Level\_ID to Premium Client.  Option B (Challenge Payment):  Send Client an email to notify them of the payment confirmation and ask them to book a suitable consultation time  Enable Client access rights for booking by updating User\_Level\_ID to in a challenge.  Option C (Exercise Plan Payment):  Email Client with payment confirmation and an exercise plan. |  |
| Actual Results: |  |  |
| Test Step 5: |  |  |
| Correct Test Data: |  |  |
| Alternative Correct Test Data: |  |  |
| Incorrect Test Data: |  |  |
| Alternative Incorrect Test Data: |  |  |
| Expected Results: | Check the functionality for Access\_Level\_ID.  Identify whether a Manager is logged in based on the Access\_Level\_ID. Navigate the user to the Manager” Dashboard” screen displaying payment details with a description “Unconfirmed” |  |
| Actual Results: |  |  |
|  |  |  |

9. Hardware and software requirements

9.1 Introduction

This section contains descriptions of the required hardware and software for the system to run.

9.2 Hardware & Software Requirements

9.2.1 Software requirements:

* A laptop/desktop computer with a HTML5-compatible browser, and operating system of Microsoft Windows 2008 or later.
* SQL server (2014 or above)
* Angular and AngularJS (latest preferred)
* Chart.JS (latest preferred)
* A mobile device running iOS/Android operating system.
* .Net Framework (latest version)

Note: All devices require an internet connection.

9.2.2 Hardware requirements:

The minimum requirements are:

* 500GB hard drive disk memory (more would be recommended)
* 4GB RAM (8GB RAM is recommended)
* Intel Core i5 processor (2.6Ghz)

9.3 Conclusion

This section contained descriptions of the required hardware and software for the system to run.

10. Network and web layout specifications

10.1 Introduction

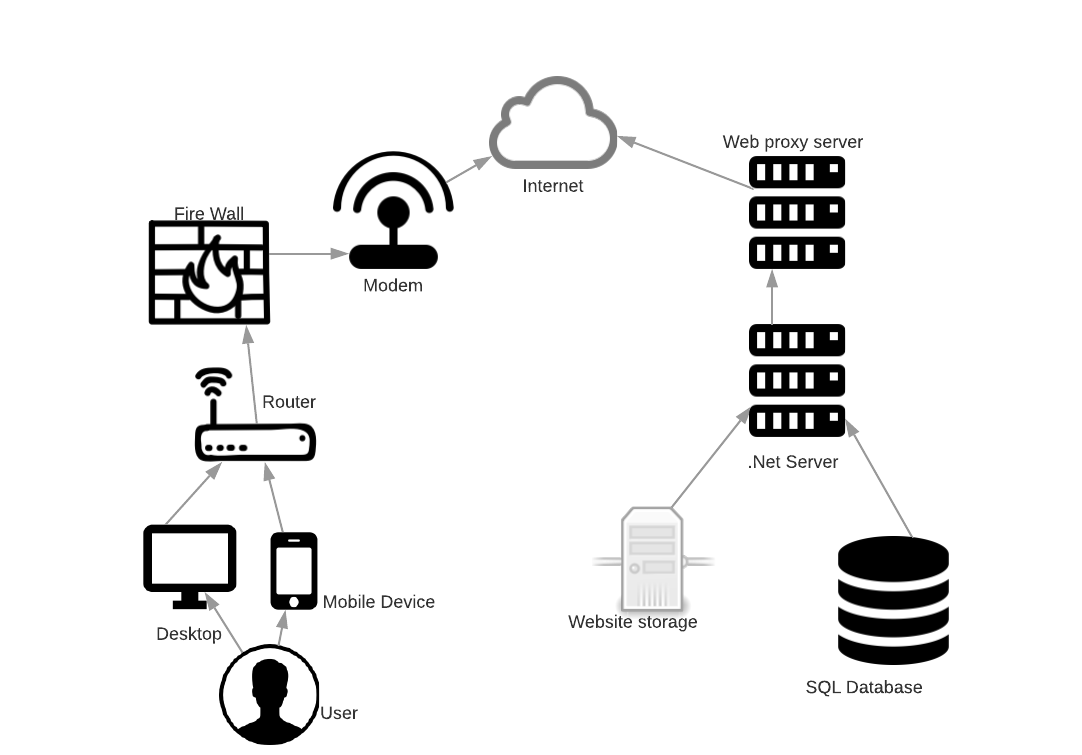
This section includes the network specifications for the system to operate, and layout.

10.2 Network and web Layout Specifications

10.2.1 Network Specifications

* 10Mb/s internet connection
* Web/server-hosting (including support for SQL server, ASP.NET and domain)
* SMTP technology

10.2.2 Network Diagram



11. Validation

11.1 Introduction

This section holds the validation of the details of the functional specification against the requirements.

11.2 Validation

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Subsystem** | Use case | Requirement | Processes | Entities |
| 1. Admin subsystem | 1.1 Add new subscription detail | 1.1 Add new subscription detail | 1.1.1 | None |
| 1.1.2 | None |
| 1.1.3 | None |
| 1.1.4 | None |
| 1.1.5 | Subscription Type |
| 1.1.6 | None |
| 1.1.7 | None |
| 1.1.8 | None |
| 1.1.9 | None |
| 1.2 Maintain subscription detail | 1.2 Maintain subscription detail | 1.2.1 | None |
| 1.2.2 | None |
| 1.2.3 | None |
| 1.2.4 | None |
| 1.2.5 | Not blank/ Subscription |
| 1.2.6 | None |
| 1.2.7 | None |
| 1.3 Search subscription details | 1.3 Search subscription details | 1.3.1 | None |
| 1.3.2 | None |
| 1.3.3 | None |
| 1.3.4 | None |
| 1.3.5 | None |
| 1.3.6 | Subscription type |
| 1.3.7 | None |
| 1.3.8 | None |
| 1.4 Add new challenge | 1.4 Add new challenge | 1.4.1 | None |
| 1.4.2 | None |
| 1.4.3 | None |
| 1.4.4 | None |
| 1.4.5 | None |
| 1.4.6 | None |
| 1.4.7 | None |
| 1.4.8 | Challenge |
| 1.4.9 | None |
| 1.4.10 | None |
| 1.4.11 | None |
| 1.4.12 | None |
| 1.5 Maintain challenge | 1.5 Maintain challenge | 1.5.1 | None |
| 1.5.2 | None |
| 1.5.3 | None |
| 1.5.4 | None |
| 1.5.5 | None |
| 1.5.6 | None |
| 1.5.7 | None |
| 1.5.8 | None |
| 1.5.9 | Challenge |
| 1.5.10 | None |
| 1.5.11 | None |
| 1.5.12 | None |
| 1.6 Search challenge | 1.6 Search challenge | 1.6.1 | None |
| 1.6.2 | None |
| 1.6.3 | None |
| 1.6.4 | None |
| 1.6.5 | Not Blank |
| 1.6.6 | None |
| 1.6.7 | None |
| 1.7 Add new booking slots | 1.7 Add new booking slots | 1.7.1 | None |
| 1.7.2 | None |
| 1.7.3 | None |
| 1.7.4 | None |
| 1.7.5 | None |
| 1.7.6 | None |
| 1.7.7 | None |
| 1.7.8 | Booking |
| 1.7.9 | None |
| 1.7.10 | None |
| 1.7.11 | None |
| 1.7.12 | None |
| 1.8 Maintain booking slots | 1.8 Maintain booking slots | 1.8.1 | None |
| 1.8.2 | None |
| 1.8.3 | None |
| 1.8.4 | None |
| 1.8.5 | None |
| 1.8.6 | None |
| 1.8.7 | None |
| 1.8.8 | None |
| 1.8.9 | Booking |
| 1.8.10 | None |
| 1.8.11 | none |
| 1.8.12 | None |
| 1.9 Search booking slots | 1.9 Search booking slots | 1.9.1 | None |
| 1.9.2 | None |
| 1.9.3 | None |
| 1.9.4 | None |
| 1.9.5 | Not Blank |
| 1.9.6 | None |
| 1.9.7 | None |
| 1.10 Maintain Company | 1.10 Maintain Company | 1.10.1 | None |
| 1.10.2 | None |
| 1.10.3 | None |
| 1.10.4 | None |
| 1.10.5 | None |
| 1.10.6 | None |
| 1.10.7 | None |
| 1.10.8 | None |
| 1.10.9 | Company |
| 1.10.10 | None |
| 1.10.11 | None |
| 1.11 Search booking | 1.11 Search booking | 1.11.1 | None |
| 1.11.2 | None |
| 1.11.3 | None |
| 1.11.4 | None |
| 1.11.5 | Not blank |
| 1.11.6 | None |
| 1.11.7 | None |
| 1.12 Search subscription | 1.12 Search subscription | 1.12.1 | None |
| 1.12.2 | None |
| 1.12.3 | None |
| 1.12.4 | None |
| 1.12.5 | Not blank |
| 1.12.6 | None |
| 1.12.7 | None |
| 1.13 Confirm payment | 1.13 Confirm payment | 1.13.1 | None |
| 1.13.2 | None |
| 1.13.3 | None |
| 1.13.4 | None |
| 1.13.5 | None |
| 1.13.6 | None |
| 1.13.7 | None |
| 1.13.8 | None |
| 1.13.9 | None |
| 1.13.10 | None |
| 2. Employee subsystem | 2.1 Add new employee | 2.1 Add new employee | 2.1.1 | None |
| 2.1.2 | None |
| 2.1.3 | None |
| 2.1.4 | None |
| 2.1.5 | Employee |
| 2.1.6 | None |
| 2.1.7 | None |
| 2.1.8 | None |
| 2.2 Maintain employee | 2.2 Maintain employee | 2.2.1 | None |
| 2.2.2 | None |
| 2.2.3 | None |
| 2.2.4 | None |
| 2.2.5 | none |
| 2.2.6 | none |
| 2.2.7 | None |
| 2.2.8 | Employee |
| 2.2.9 | None |
| 2.2.10 | None |
| 2.3 Search employee | 2.3 Search employee | 2.3.1 | None |
| 2.3.2 | None |
| 2.3.3 | None |
| 2.3.4 | None |
| 2.3.5 | None |
| 2.4 Add employee type | 2.4 Add employee type | 3.4.1 | None |
| 3.4.2 | None |
| 3.4.3 | None |
| 3.4.4 | None |
| 3.4.5 | Employee Type |
| 2.4.6 | None |
| 2.4.7 | None |
| 2.4.8 | None |
| 2.5 Maintain employee type | 2.5 Maintain employee type | 2.5.1 | None |
| 2.5.2 | None |
| 2.5.3 | None |
| 2.5.4 | None |
| 2.5.5 | None |
| 2.5.6 | None |
| 2.5.7 | None |
| 2.5.8 | Employee Type |
| 2.5.9 | none |
| 2.5.10 |  |
| 2.6 Search employee type | 2.6.1 | None |
| 2.6 Search employee type | 2.6.2 | None |
| 2.6.3 | None |
| 2.6.4 | None |
| 2.6.5 | None |
| 2.6.6 | none |
| 2.6.7 | Access level |
| 2.6.8 | None |
| 3. Client Subsystem | 3.1 Register new client | 3.1 Register new client | 4.1.1 | None |
| 4.1.2 | None |
| 4.1.3 | None |
| 4.1.4 | None |
| 4.1.5 | Employee |
| 4.1.6 | none |
| 4.1.7 | None |
| 4.1.8 | None |
| 4.1.9 | none |
| 3.2 Maintain client profile | 3.2 Maintain client profile | 4.2.1 | None |
| 4.2.2 | None |
| 4.2.3 | None |
| 4.2.4 | None |
| 4.2.5 | none |
| 4.2.6 | none |
| 4.2.7 | None |
| 4.2.8 | Employee |
| 4.2.9 | none |
| 4.2.10 | none |
| 3.3 Search Client | 3.3 Search Client | 4.3.1 | None |
| 4.3.2 | None |
| 4.3.3 | None |
| 4.3.4 | None |
| 4.3.5 | none |
| 4.3.6 | Employee |
| 4.3.7 | None |
| 4.3.8 | none |
| 3.4 Sign Up for subscription | 3.4 Sign Up for subscription | 4.2.1 | None |
| 4.2.2 | None |
| 4.2.3 | None |
| 4.2.4 | None |
| 4.2.5 | Employee type |
| 4.2.6 | none |
| 4.2.7 | None |
| 4.2.8 | none |
| 4.2.9 | none |
| 3.5 Register for challenge | 3.5 Register for challenge | 4.5.1 | None |
| 4.5.2 | None |
| 4.5.3 | None |
| 4.5.4 | None |
| 4.5.5 | none |
| 4.5.6 | none |
| 4.5.7 | None |
| 4.5.8 | Employee Type |
| 4.5.9 | none |
| 4.5.10 | none |
| 3.6 Unregister for a challenge | 3.6 Unregister for a challenge | 4.6.1 | None |
| 4.6.2 | None |
| 4.6.3 | None |
| 4.6.4 | None |
| 4.6.5 | none |
| 4.6.6 | none |
| 4.6.7 | Employee Type |
| 4.6.8 | none |
| 4. User subsystem | 4.1 Login | 4.1 Login | 4.1.1 | None |
| 4.1.2 | None |
| 4.1.3 | None |
| 4.1.4 | None |
| 4.1.5 | User |
| 4.1.6 | none |
| 4.1.7 | None |
| 4.1.8 | None |
| 4.2 Update password | 4.2 Update password | 4.2.1 | None |
| 4.2.2 | None |
| 5.2.3 | None |
| 4.2.4 | None |
| 4.2.5 | User |
| 4.2.6 | None |
| 4.2.7 | None |
| 4.2.8 | None |
| 4.2.9 | User |
| 4.2.10 | None |
| 4.2.11 | None |
| 4.3 Forgotten password | 4.3 Forgotten password | 4.3.1 | None |
| 4.3.2 | None |
| 4.3.3 | None |
| 4.3.4 | None |
| 4.3.5 | User |
| 4.3.6 | None |
| 4.3.7 | None |
| 4.4 Logout | 4.4 Logout | 4.4.1 | None |
| 4.4.2 | None |
| 4.4.3 | None |
| 4.4.4 | None |
| 4.4.5 | None |
| 4.5 Add access level | 4.5 Add access level | 4.5.1 | None |
| 4.5.2 | None |
| 4.5.3 | None |
| 4.5.4 | None |
| 4.5.5 | Access Level |
| 4.5.6 | none |
| 4.5.7 | None |
| 4.5.8 | None |
| 4.6 Maintain access level | 4.6 Maintain access level | 4.6.1 | None |
| 4.6.2 | None |
| 4.6.3 | None |
| 4.6.4 | None |
| 4.6.5 | None |
| 4.6.6 | None |
| 4.6.7 | None |
| 4.6.8 | Access Level |
| 4.6.9 | None |
| 4.6.10 | None |
| 4.7 Search access level | 4.7 Search access level | 4.7.1 | None |
| 4.7.2 | None |
| 4.7.3 | None |
| 4.7.4 | None |
| 4.7.5 | none |
| 4.8 Add functionality | 4.8 Add functionality | 4.8.1 | None |
| 4.8.2 | None |
| 4.8.3 | None |
| 4.8.4 | None |
| 4.8.5 | Functionality |
| 4.8.6 | none |
| 4.8.7 | None |
| 4.8.8 | None |
| 4.9 Maintain Functionality | 4.9 Maintain Functionality | 4.9.1 | None |
| 4.9.2 | None |
| 4.9.3 | None |
| 4.9.4 | None |
| 4.9.5 | None |
| 4.9.6 | none |
| 4.9.7 | None |
| 4.9.8 | Functionality |
| 4.9.9 | None |
| 4.9.10 | None |
| 4.10 Search functionality | 4.10 Search functionality | 4.10.1 | None |
| 4.10.2 | None |
| 4.10.3 | None |
| 4.10.4 | None |
| 4.10.5 | None |
| 5. Reporting Subsystem | 5.1 Generate exercise plan report | 5.1 Generate exercise plan report | 5.1.1 | None |
| 5.1.2 | None |
| 5.1.3 | None |
| 5.1.4 | None |
| 5.1.5 | None |
| 5.1.6 | none |
| 5.1.7 | Start date/ end date |
| 5.1.8 | None |
| 5.2 Generate subscription report | 5.2 Generate subscription report | 5.2.1 | None |
| 5.2.2 | None |
| 5.2.3 | None |
| 5.2.4 | None |
| 5.2.5 | None |
| 5.2.6 | none |
| 5.2.7 | Start date/ end date |
| 5.2.8 | None |
| 5.3 Generate challenge entry report | 5.3 Generate challenge entry report | 5.3.1 | None |
| 5.3.2 | None |
| 5.3.3 | None |
| 5.3.4 | None |
| 5.3.5 | None |
| 5.3.6 | none |
| 5.3.7 | None |
| 5.3.8 | None |
| 5.4 Generate consultation booking details report | 5.4 Generate consultation booking details report | 5.4.1 | None |
| 5.4.2 | None |
| 5.4.3 | None |
| 5.5 Generate personal training booking details report | 5.5 Generate personal training booking details report | 5.5.1 | None |
| 5.5.2 | None |
| 5.5.3 | None |
| 6. Exercise subsystem | 6.1 Add new exercise plan type | 6.1 Add new exercise plan type | 6.1.1 | None |
| 6.1.2 | None |
| 6.1.3 | None |
| 6.1.4 | None |
| 6.1.5 | None |
| 6.1.6 | none |
| 6.1.7 | None |
| 6.1.9 | none |
| 6.1.10 | Exercise plan type |
| 6.1.11 | None |
| 6.1.12 | None |
| 6.1.13 | None |
| 6.1.14 | None |
| 6.2 Maintain exercise plan type | 6.2 Maintain exercise plan type | 6.2.1 | None |
| 6.2.2 | None |
| 6.2.3 | None |
| 6.2.4 | None |
| 6.2.5 | none |
| 6.2.6 | none |
| 6.2.7 | None |
| 6.2.8 | None |
| 6.2.9 | Exercise plan type |
| 6.2.10 | none |
| 6.2.11 | None |
| 6.2.12 | none |
| 6.3 Search exercise plan type | 6.3 Search exercise plan type | 6.3.1 | None |
| 6.3.2 | None |
| 6.3.3 | None |
| 6.3.4 | None |
| 6.3.5 | none |
| 6.3.6 | Exercise plan type |
| 6.3.7 | None |
| 6.3.8 | none |
| 6.4 Add new exercise type | 6.4 Add new exercise type | 6.4.1 | None |
| 6.4.2 | None |
| 6.4.3 | None |
| 6.4.4 | None |
| 6.4.5 | None |
| 6.4.6 | none |
| 6.4.7 | None |
| 6.4.8 | Exercise type |
| 6.4.9 | none |
| 6.4.10 | none |
| 6.5 Maintain exercise type | 6.5 Maintain exercise type | 6.5.1 | None |
| 6.5.2 | None |
| 6.5.3 | None |
| 6.5.4 | None |
| 6.5.5 | none |
| 6.5.6 | none |
| 6.5.7 | None |
| 6.5.8 | Exercise type |
| 6.5.9 | none |
| 6.5.10 | none |
| 6.6 Search exercise type | 6.6 Search exercise type | 6.6.1 | None |
| 6.6.2 | None |
| 6.6.3 | None |
| 6.6.4 | None |
| 6.6.5 | No Blanks |
| 6.6.6 | None |
| 6.6.7 | None |
| 6.7 Add new rep | 6.7 Add new rep | 6.7.1 | None |
| 6.7.2 | None |
| 6.7.3 | None |
| 6.7.4 | None |
| 6.7.5 | None |
| 6.7.6 | none |
| 6.7.7 | None |
| 6.7.8 | Rep |
| 6.7.9 | None |
| 6.7.10 | None |
| 6.8 Maintain rep | 6.8 Maintain rep | 6.8.1 | None |
| 6.8.2 | None |
| 6.8.3 | None |
| 6.8.4 | None |
| 6.8.5 | none |
| 6.8.6 | none |
| 6.8.7 | None |
| 6.8.8 | Rep |
| 6.8.9 | none |
| 6.8.10 | none |
| 6.9 Add new set | 6.9 Add new set | 6.9.1 | None |
| 6.9.2 | None |
| 6.9.3 | None |
| 6.9.4 | None |
| 6.9.5 | None |
| 6.9.6 | None |
| 6.9.7 | Set |
| 6.9.8 | none |
| 6.9.9 | none |
| 6.9.10 | None |
| 6.10 Maintain set | 6.10 Maintain set | 6.10.1 | None |
| 6.10.2 | None |
| 6.10.3 | None |
| 6.10.4 | None |
| 6.10.5 | Set |
| 6.10.6 | none |
| 6.10.7 | None |
| 6.10.8 | none |
| 6.11 Add new exercise body type | 6.11 Add new exercise body type | 6.11.1 | None |
| 6.11.2 | None |
| 6.11.3 | None |
| 6.11.4 | None |
| 6.11.5 | none |
| 6.11.6 | Exercise Body Type |
| 6.11.7 | None |
| 6.11.8 | none |
| 6.11.9 | None |
| 6.11.10 | None |
| 6.12 Maintain exercise body type | 6.12 Maintain exercise body type | 6.12.1 | None |
| 6.12.2 | None |
| 6.12.3 | None |
| 6.12.4 | None |
| 6.12.5 | None |
| 6.12.6 | none |
| 6.12.7 | None |
| 6.12.8 | none |
| 6.12.9 | Exercise Body Type |
| 6.12.10 | None |
| 6.12.11 | None |
| 6.12.12 | none |
| 6.13 Search exercise body type | 6.13 Search exercise body type | 6.13.1 | None |
| 6.13.2 | None |
| 6.13.3 | None |
| 6.13.4 | None |
| 6.13.5 | Not blank |
| 6.13.6 | none |
| 6.13.7 | None |
| 6.14 Add new exercise | 6.14 Add new exercise | 6.14.1 | None |
| 6.14.2 | None |
| 6.14.3 | None |
| 6.14.4 | None |
| 6.14.5 | none |
| 6.14.6 | None |
| 6.14.7 | None |
| 6.14.8 | none |
| 6.14.9 | Exercise |
| 6.14.10 | None |
| 6.14.11 | None |
| 6.14.12 | None |
| 6.15 Maintain exercise | 6.15 Maintain exercise | 6.15.1 | None |
| 6.15.2 | None |
| 6.15.3 | None |
| 6.15.4 | None |
| 6.15.5 | None |
| 6.15.6 | none |
| 6.15.7 | None |
| 6.15.8 | none |
| 6.15.9 | Exercise |
| 6.15.10 | None |
| 6.15.11 | None |
| 6.15.12 | None |
| 6.16 Search exercise | 6.16 Search exercise | 6.16.1 | None |
| 6.16.2 | None |
| 6.16.3 | None |
| 6.16.4 | None |
| 6.16.5 | No blanks |
| 6.16.6 | none |
| 6.16.7 | None |
| 6.17 Add new workout | 6.17 Add new workout | 6.17.1 | None |
| 6.17.2 | None |
| 6.17.3 | None |
| 6.17.4 | None |
| 6.17.5 | none |
| 6.17.6 | None |
| 6.17.7 | Workout |
| 6.17.8 | none |
| 6.17.9 | None |
| 6.17.10 | None |
| 6.18 Maintain workout | 6.18 Maintain workout | 6.18.1 | None |
| 6.18.2 | None |
| 6.18.3 | None |
| 6.18.4 | None |
| 6.18.5 | None |
| 6.18.6 | none |
| 6.18.7 | None |
| 6.18.8 | none |
| 6.18.9 | Workout |
| 6.18.10 | None |
| 6.18.11 | None |
| 6.18.12 | None |
| 6.19 Search workout | 6.19 Search workout | 6.19.1 | None |
| 6.19.2 | None |
| 6.19.3 | None |
| 6.19.4 | None |
| 6.19.5 | No blanks |
| 6.19.6 | none |
| 6.19.7 | None |
| 6.20 Add new exercise plan | 6.20 Add new exercise plan | 6.20.1 | None |
| 6.20.2 | None |
| 6.20.3 | None |
| 6.20.4 | None |
| 6.20.5 | none |
| 6.20.6 | None |
| 6.20.7 | None |
| 6.20.8 | none |
| 6.20.9 | None |
| 6.20.10 | Exercise Plan |
| 6.20.11 | None |
| 6.20.12 | None |
| 6.20.13 | None |
| 6.20.14 | None |
| 6.21 Maintain exercise plan | 6.21 Maintain exercise plan | 6.21.1 | None |
| 6.21.2 | None |
| 6.21.3 | None |
| 6.21.4 | None |
| 6.21.5 | None |
| 6.21.6 | None |
| 6.21.7 | None |
| 6.21.8 | None |
| 6.21.9 | Exercise Plan |
| 6.21.10 | None |
| 6.21.11 | None |
| 6.21.12 | None |
| 6.22 Search exercise plan | 6.22 Search exercise plan | 6.22.1 | None |
| 6.22.2 | None |
| 6.22.3 | None |
| 6.22.4 | None |
| 6.22.5 | No blanks |
| 6.22.6 | none |
| 6.22.7 | None |
| 6.23 Create workout | 6.23 Create workout | 6.23.1 | None |
| 6.23.2 | None |
| 6.23.3 | None |
| 6.23.4 | None |
| 6.23.5 | None |
| 6.23.6 | Workout |
| 6.23.7 | None |
| 6.23.8 | none |
| 6.24 Create exercise plan | 6.24 Create exercise plan | 6.24.1 | None |
| 6.24.2 | None |
| 6.24.3 | None |
| 6.24.4 | None |
| 6.24.5 | Workout |
| 6.24.6 | none |
| 6.24.7 | None |
| 6.24.8 | none |

12. Sign Off by Client

12.1 Introduction

This section holds the formally signed page where the client specifies that he has read the document understands it and agrees with its contents.

12.2 Sign-Off

I on behalf JessterFIT, declare that I have read the document, and hereby state that I understand it and fully agree with its content.



12.3 Conclusion

The client has read through and approved of the quality and content of this document and gives consent to its submission.

13. Complexity

13.1 Introduction

This section of Deliverable 4 has the complexity matrix as calculated according to the content presented in the deliverable, indicating the predicted functionality of the proposed system

13.2 Complexity Matrix

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Topic** | **Level** |  | **Marks** | **MAX** |
| **1. Special GUI** | For online applications: Responsive web design | 3 | 3 | **42** |
| For desktop applications: Form design according to design principles (Schneiderman’s golden rule on navigation applies here) |
| Appropriate use of grids/tables |  | 3 |
| Appropriate use of tabs/links |  | 3 |
| Use of graphs in an appropriate business context |  | 4 |
| The storage and display of graphical information, like photos with a good business reason |  | 3 |
| Working e-mail automatically generated from the database in an appropriate business context |  | 2 |
| SMS messages automatically generated from the system in an appropriate business context |  | 2 |
| Extensive user-friendly search facility |  | 3 |
| At least one use of a tree to display data from the database |  | 3 |
| Able to dynamically modify a data tree structure and in doing so adjusting the data in the database |  | 4 |
| At least one use of a calendar view of data (not a date/time picker; not a plug-in such as Google calendar) |  | 3 |
| Uploading a file into the system with appropriate business reason |  | 3 |
| The use of audio/video in an appropriate business context |  | 3 |
| At least one use of an administrator configurable timer in an appropriate business context |  | 3 |
| **2. Database access** | At least 30 tables used (4 member groups) or 40 tables used (5 member groups) | 6 | 6 | **15** |
| Full referential integrity on all tables | 6 | 6 |
| At least one use of master-detail table relationships (Schneiderman’s golden rule on system status applies here) | \* | 3 |
| **3. Reports** | At least 3 simple list reports in a reporting tool (no control breaks, no graphs, single table) | 3 | 3 | **15** |
| At least 2 transactional report with 2 or more control breaks (with heading and calculated values/totals, multiple tables) | \* | 6 |
| At least 1 report with adjustable criteria | 3 | 3 |
| At least 1 management report using a graph |  | 3 |
| **4. Flexibility** | All data that can change in future should not be hard coded but maintained in a sub-module of the system (e.g. Lookup tables) | 6 | 6 | **12** |
| Some business rules are not hard coded but maintained in a sub-module of the system. | 6 | 6 |
| **5. Error handling** | **All** system-generated errors are trapped, and consistent, user-friendly error messages are displayed | 6 | 6 | **12** |
| Appropriate data validation on **all** input fields | 6 | 6 |
| **6. Help** | At least one menu item or other control that opens up a complete help document (HTML, PDF, Help-file) |  | 3 | **15** |
| Extensive context-sensitive help. E.g. calling Help on a specific screen/function will automatically open the specific help for that screen/function. |  | 6 |
| Search Facility on Help |  | 3 |
| Extensive use of hints |  | 3 |
| **7. Security** | Logon screen with user ID and password and fixed user profiles | 3 | 3 | **13** |
| Applying two factor authentications with applicable business reason. |  | 3 |
| Encrypted passwords in database |  | 1 |
| Flexible user profiles (i.e. you can dynamically add user profiles that will enable/disable access to certain parts of the system) | 6 | 6 |
| **8. Audit Trail** | An audit trail of all transactions in the system showing at least date, time, user, transaction type, critical data (such as amount and quantity of transaction) |  | 6 | **9** |
| Able to search the audit trail on any of the following: date, user, transaction type |  | 3 |
| **9. Deployment** | For a desktop application: Fully functional installation disks that take care of application installation requirements (install and uninstall) |  | 3 | **15** |
| For an online application: Deployment of application to a publicly accessible web server |  | 3 |
| For a mobile application: Deployment to an App Market place (such as the Play Store or the AppStore) |  | 6 |
| Deployment of the database to a remote database server |  | 3 |
| **10. Backup and Restore** | A backup and restore subsystem exist that backup/restore all data (system may exit during restore) |  | 3 | **3** |
| **11. Import/Export Data** | Able to open Word or Excel and automatically place data in it based on the parameters provided (with a good business reason) |  | 6 | **9** |
| XML or JSON: At least 1 XML or JSON file for Importing or Exporting of data (with good business reason) |  | 3 |
| **12. External INPUT device** | Simple Link to an external INPUT device using plug-and-play technology, such as a swipe card reader, bar code reader, etc. or a native component such as a QR reader, a GPS component, etc |  | 3 | **18** |
| Loose Link to an external INPUT device using device specific software. Data or images must seamlessly be stored in the database, but device specific software is visible to the user. (This could include a digital camera, scanner, voice recording device, thump print reader, etc.) |  | 6 |
| Tight Link to an external INPUT device using device specific software. Data or images must seamlessly be stored in the database, but device specific software is **not** visible to the user. (This could include a digital camera, scanner, voice recording device, thump print reader, etc.) |  | 9 |
| **13. External APPLICATION / Services** | Integrate an existing web service into your application (with good business reason) |  | 3 | **9** |
| A fully functional link to an installed external application system exists and the interface must be shown to work on the external system. Note that this excludes Microsoft Office Applications |  | 6 |
| **14. Multiplatform processing for an appropriate business reason** | Appropriate business use of static views on an alternative platform. |  | 3 | **27** |
| Appropriate use of dynamic views on an alternative platform (i.e. data is displayed from the system’s database) |  | 3 |
| Appropriate use of substantial dynamic views on an alternative platform (i.e. both reading and writing data from the system’s database) |  | 9 |
| Uploading a file through an alternative platform onto the system’s database. |  | 3 |
| Substantial processing on a third platform (i.e. both reading and writing data from the system’s database) |  | 9 |
| **15. Programming Principles** | The use of a data layer to facilitate interaction between your database and your business layer |  | 3 | **12** |
| The use of an API to facilitate interaction between your business layer and your presentation layer |  | 6 |
| Comprehensive use of stored procedures and/or triggers and/or jobs. |  | 3 |
| **16. Innovative addition to the system** | Any very advanced innovative addition to the system (e.g. machine learning, AI, block chain, text mining, IOT, etc.) | # | 9 | **9** |
|  |  |  |  |  |
|  | **Maximum Complexity Marks** |  |  | **235** |
|  | Group 15 Complexity Marks so far: |  |  | 44/150 |

14. Team Sign Off

14.1 Introduction

This section holds the formally signed page where the team, Aegis Group 15, signs off on the content of this document and its quality.



14.3 Conclusion

Aegis Group 15 is proud of the quality of the content within this document and hands it over as the official submission to the Department of Informatics as the document for deliverable 3 & 4.

15. Document Conclusion

Deliverable Four of our INF 370 project was the technical specification of our system. The technical specification document specified how the functions, as specified in Deliverable 2, will be performed. In this deliverable, we discussed the use cases, the technical use case narratives, the process models of the proposed system (functional decomposition to primitive level), system database design (entity relationship model and size estimation of proposed database with 5-year projection), system interface designs including input designs, output designs and other interfaces. The test specifications of the test plan, test scenarios and procedures, as well as test data were included. Hardware and software requirements, network/web Layout specifications, and validation of the technical specification against the requirements as well.

16. Appendices

16.1 Introduction

This section of our deliverable holds the appendices of the document. The appendices will include the full functional decomposition diagram, the functional decomposition diagram for subsystem six and subsystem five, the entity relational diagram, and input design consisting of listed screens, and other reports.

16.2 List of attached appendices

16.3 Conclusion

This section of our deliverable held the appendices of the document.