Requirements Engineering

DJ Booking System

Submitted By: Colm Gallagher (T00170514)

Computing with Multi Media

Date Submitted: 07/12/2018

**Table of Contents**

[1. Introduction/overview 4](#_Toc528356147)

[2. Functional Components 6](#_Toc528356148)

[3. User Requirements 7](#_Toc528356149)

[3.1. DJSYS will manage client administration. 7](#_Toc528356153)

[3.1.1. DJSYS will register a new client. 7](#_Toc528356154)

[3.1.2. DJSYS will allow client details to be updated. 7](#_Toc528356155)

[3.1.3. DJSYS will allow client details to be withdrawn. 7](#_Toc528356156)

[3.1.4. DJSYS will provide a list of clients. 7](#_Toc528356157)

[3.2. DJSYS will manage services. 7](#_Toc528356158)

[3.2.1. DJSYS will register a new service. 7](#_Toc528356159)

[3.2.2. DJSYS will allow service details to be updated. 7](#_Toc528356160)

[3.2.3. DJSYS will provide a list of services. 7](#_Toc528356161)

[3.3. DJSYS will perform DJ administration. 7](#_Toc528356162)

[3.3.1. DJSYS will register a new DJ. 7](#_Toc528356163)

[3.3.2. DJSYS will allow DJ details to be withdrawn. 7](#_Toc528356164)

[3.3.3. DJSYS will provide a list of DJ bookings. 7](#_Toc528356165)

[3.4. DJSYS will process bookings 7](#_Toc528356166)

[3.4.1. DJSYS will allow a client to make a booking. 7](#_Toc528356167)

[3.4.2. DJSYS will allow a client to cancel a booking. 7](#_Toc528356168)

[3.4.3. DJSYS will allow a client to change a booking. 7](#_Toc528356169)

[3.5. DJSYS will perform administrative reporting 7](#_Toc528356170)

[3.5.1. DJSYS will provide a monthly revenue analysis report. 7](#_Toc528356171)

[3.5.2. DJSYS will provide a service analysis report. 7](#_Toc528356172)

[4. System Requirements 8](#_Toc528356173)

[System Level Use Case Diagram 8](#_Toc528356174)

[4.1. Manage Clients 9](#_Toc528356175)

[**4.1.1.** **Add Client** 9](#_Toc528356176)

[**4.1.2.** **Update Client** 13](#_Toc528356177)

[**4.1.3.** **Withdraw Client** 17](#_Toc528356178)

[**4.1.4.** **Query Client** 20](#_Toc528356179)

[4.2. Manage Services 23](#_Toc528356180)

[**4.2.1.** **Log Service** 23](#_Toc528356181)

[**4.2.2.** **Update Service** 26](#_Toc528356182)

[**4.2.3.** **Query Service** 29](#_Toc528356183)

[4.3. Perform DJ Administration 31](#_Toc528356184)

[**4.3.1.** **Add DJ** 31](#_Toc528356185)

[**4.3.2.** **Withdraw DJ** 34](#_Toc528356186)

[**4.3.3.** **Show DJ Schedule** 37](#_Toc528356187)

[4.4. Process Bookings 42](#_Toc528356188)

[**4.4.1.** **Make Booking** 42](#_Toc528356189)

[**4.4.2.** **Cancel Booking** 51](#_Toc528356190)

[**4.4.3.** **Change Booking** 55](#_Toc528356191)

[4.5. Perform Administration 59](#_Toc528356192)

[**4.5.1.** **Analyse Revenue by Month** 59](#_Toc528356193)

[**4.5.2.** **Analyse Revenue by Service** 61](#_Toc528356194)

[5. System Model 63](#_Toc528356195)

[5.1. Level-0 DFD 63](#_Toc528356196)

[5.2. Level-1 DFD 64](#_Toc528356197)

[5.3. Level-2 DFD (Process P1: Title) 65](#_Toc528356198)

[5.4. Level-2 DFD (Process P2: Title) 66](#_Toc528356199)

[5.5. Level-2 DFD (Process P3: Title) 67](#_Toc528356200)

[6. Data Model (Class Diagram) 70](#_Toc528356201)

[6.1. Class Diagram 71](#_Toc528356202)

[6.2. Relational Schema 72](#_Toc528356203)

[6.3. Database Schema 73](#_Toc528356204)

[7. Conclusion 76](#_Toc528356205)

[8. Appendices 77](#_Toc528356206)

[8.1. Appendix A – Title 77](#_Toc528356207)

[8.2. Appendix B – Title 77](#_Toc528356208)

# Introduction/overview

The aim of the project, “DJSys”, is to create a system which provides a workforce of DJs to be booked by Clients and for I, as the administrator, to be able to run the operation through a system which provides good, practical, functionality.

**The DJ system is split into five main processes:**

## Manage Client administration.

## Manage Service administration.

## Manage DJ administration.

## Process Booking details

## Perform administrative reporting

**Client administration will allow for:**

* Client to be registered enabling them to make Bookings.
* Client to be updated so their records are accurately stored.
* Client to be withdrawn meaning they will no longer be able to make Bookings once any outstanding financial commitments are processed.
* Client to be queried allowing the administrator to browse the Clients Booking details.

**Service administration function provides for:**

* Service to be added to the system which enables it to be Booked.
* Service to be updated meaning the correct details are always stored.
* Service to be queried allowing the administrator to get a more detailed insight into a particular Service.

**DJ administration handled by the following functions:**

* DJ to be registered allowing them to accept a Booking.
* DJ to be withdrawn meaning they will no longer be able to accept Bookings once any outstanding financial commitments are processed.
* DJ schedule to be shown allowing the administrator to attain an up to date record of a particular DJ’s Bookings.

**Booking processes are maintained as follows:**

* Booking to be made by a Client and to be provided by A DJ who are both registered and active on the system.
* Booking to be cancelled by either the Client or the Manager once any outstanding financial commitments are processed.
* Booking to be changed by a Client within an agreed time period set out in business rules.

**Administrative reporting to be carried out using the following functions:**

* Analyse revenue by year which allows the administrator to generate, in graph form, a financial record of the year chosen to be viewed.
* Analyse revenue by Service giving the administrator the opportunity to get further insight into which Services provide the best revenue stream. This is also produced as a graph.

# Functional Components

# User Requirements



## DJSYS will Manage Client administration.

### DJSYS will register a new Client.

### DJSYS will allow Client details to be updated.

### DJSYS will allow Client details to be withdrawn.

### DJSYS will perform a Client query.

## DJSYS will Manage Service administration.

### DJSYS will register a new Service.

### DJSYS will allow Service details to be updated.

### DJSYS will provide a Service query.

## DJSYS will Manage DJ administration.

### DJSYS will register a new DJ.

### DJSYS will allow DJ details to be withdrawn.

### DJSYS will provide a list of DJ Bookings.

## DJSYS will Process Booking details

### DJSYS will record details of a Booking.

### DJSYS will allow a client to cancel a Booking.

### DJSYS will record details of a change of Booking.

## DJSYS will Perform administrative reporting

### DJSYS will provide a yearly revenue analysis report.

### DJSYS will provide a Service revenue analysis report.

# System Requirements

DJSYS consists of five main modules.

The Client module provides functions to add a Client, update a Client, withdraw a Client, and display a list of Clients.

The Service module provides functions to log a new Service, update a Service, and display a list of Services.

The DJs module provides functions to add a DJ, withdraw a DJ, and display a list of DJ Bookings.

The Bookings module provides functions to make a Booking, cancel a Booking, and change a Booking.

The Administration module provides functions to display a yearly revenue analysis, and a Service revenue analysis.

## System Level Use Case Diagram

The diagram displayed below shows the five main functions described above and how they are interacted with, and by which entities.

DJSYS

Manager

Client

DJ

## Manage Client Administration

This component will provide functions to add a Client, update a Client, withdraw a Client, and display a list of Clients.

### **Add Client**

This function records the details of a Client on the system. This Client can then make Bookings.

Manager

Client

<<Includes>>

[Cite your source here.]

<<Extends>>

[Cite your source here.]

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Add Client** | |
| **Use Case Id** | 4.1.1. | |
| **Priority** | High | |
| **Source** | Manager | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** | Client | |
| **Description** | This function records the details of a Client on the system. This Client can then make Bookings. | |
| **Preconditions** |  | |
| **Trigger** | Client must have completed a Client registration form. | |
| **Expected Scenario** | **Actor Action** | **System Response** |
|  | **Step 1:** The Manager invokes the Add Client function.  **Step 4:** The Manager enters the Client details:   * Forename * Surname * Date\_Of\_Birth * Contact\_No * Email * Street * Town * City * County * Eircode   **Step 5:** The Manager confirms that the Client is to be added.  **Step 10:** The Manager selects the send email option. | **Step 2:** The system determines the next Client\_ID.  **Step 3:** The system displays the UI.  **Step 6:** The system validates the data entered:   * All fields must be entered. * Forename must be max length 35 and not numeric. * Surname must be max length 35 and not numeric. * Date\_Of\_Birth must be > 18 years. * Contact\_No must be max length 20 and not numeric. * Email must be valid format. * Street must be max length 50 and not numeric. * Town must be max length 50 and not numeric. * City must be max length 50 and not numeric. * County must be max length 15 and not numeric. * Eircode must be max length 7 and not numeric.   **Step 7:** Set Client status to “A” (Active).  **Step 8:** The system saves the Client details in the ***Client file:***   * Client\_ID * Forename * Surname * Date\_Of\_Birth * Contact\_No * Email * Street * Town * City * County * Eircode * Status   **Step 9:** The system displays a “Client registered” confirmation message.  **Step 11:** The system sends an email to the Client confirming their registration on the system.  **Step 12:** The system resets the UI. |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Invalid Data Entered** |  | **Step 6:** Invalid data is entered.    **Step 7:** The system displays an appropriate error message.  **Step 8:** The system returns the user to re-enter Client details in step 4. |
| **Conclusions** | A new Client is created in the ***Client File***. | |
| **Post conditions** | Client is added and enabled to make Bookings. | |
| **Business Rules** | A Client must be over the age of 18 to register. | |
| **Implementation Constraints** |  | |

### **Update Client**

This function updates the details of a Client on the system. Only “Active” Clients may be updated.

Manager

Client

<<Includes>>

[Cite your source here.]

<<Extends>>

[Cite your source here.]

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Update Client** | |
| **Use Case Id** | 4.1.2. | |
| **Priority** | High | |
| **Source** | Manager | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** | Client | |
| **Description** | This function updates the details of a Client on the system. This Client can then continue to make Bookings. | |
| **Preconditions** | Client must have previously been registered and have an “Active” status. | |
| **Trigger** |  | |
| **Expected Scenario** | **Actor Action** | **System Response** |
|  | **Step 1:** The Manager invokes the Update Client function.  **Step 3:** The Manager enters the Surname (or part of) of the Client to be updated.  **Step 5:** The Manager selects the Client to be updated.  **Step 7:** The Manager updates the Client details:   * Forename * Surname * Date\_Of\_Birth * Contact\_No * Email * Street * Town * City * County * Eircode   **Step 8:** The Manager confirms that the Client is to be updated.  **Step 12:** The Manager selects the send email option. | **Step 2:** The system displays the UI.  **Step 4:** The system retrieves summary details of all “Active” Clients with matching name from the ***Client File*** and displays on the UI in alphabetical order of Surname, Forename.  **Step 6:** The system retrieves all details of the selected Client from the ***Client File*** and displays on the UI for editing.  **Step 9:** The system validates the data entered:   * All fields must be entered. * Forename must be max length 35 and not numeric. * Surname must be max length 35 and not numeric. * Date\_Of\_Birth must be > 18 years. * Contact\_No must be max length 20 and not numeric. * Email must be valid format. * Street must be max length 50 and not numeric. * Town must be max length 50 and not numeric. * City must be max length 50 and not numeric. * County must be max length 15 and not numeric. * Eircode must be max length 7 and not numeric. * Status must be max length 1 and not numeric.   **Step 10:** The system updates the Client details in the ***Client file.***   * Client\_ID * Forename * Surname * Date\_Of\_Birth * Contact\_No * Email * Street * Town * City * County * Eircode * Status   **Step 11:** The system displays a “Client updated” confirmation message.  **Step 13:** The system sends an email to the Client confirming that their details have been updated on the system.  **Step 14:** The system resets the UI. |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Client not found** |  | **Step 4:** No data is displayed.  **Step 5:** The system displays an appropriate error message.  **Step 6:** The system returns the user to re-enter Client details in step 3. |
| **Invalid Data Entered** |  | **Step 9:** Invalid data is entered.    **Step 10:** The system displays an appropriate error message.  **Step 11:** The system returns the user to re-enter Client details in step 7. |
| **Conclusions** | The Client is updated in the ***Client File***. | |
| **Post conditions** | Client is updated and enabled to continue to make Bookings. | |
| **Business Rules** | Only “Active” Clients may be updated. | |
| **Implementation Constraints** |  | |

### **Withdraw Client**

This function changes the status of an “Active” Client to “Withdrawn”. Once a Client is “Withdrawn”, they can no longer make a Booking. A Client cannot be withdrawn if future Bookings exist for the Client.

Manager

Client

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Withdraw Client** | |
| **Use Case Id** | 4.1.3. | |
| **Priority** | High | |
| **Source** | Manager | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** | Client | |
| **Description** | This function withdraws the details of a Client on the system. This Client can then no longer make Bookings. | |
| **Preconditions** | Client must have an “Active” status. | |
| **Trigger** | Client must have requested to be removed OR Manager decides to remove Client from system. | |
| **Expected Scenario** | **Actor Action** | **System Response** |
|  | **Step 1:** The Manager invokes the Withdraw Client function.  **Step 3:** The Manager enters the Surname (or part of) of the Client to be withdrawn.  **Step 5:** The Manager selects the Client to be withdrawn.  **Step 9:** The Manager confirms that the Client is to be withdrawn.  **Step 13:** The Manager selects the send email option. | **Step 2:** The system displays the UI.  **Step 4:** The system retrieves summary details of all “Active” clients with matching name from the ***Client File*** and displays on the UI in alphabetical order of Surname, Forename.  **Step 6:** The system retrieves all details of the selected Client from the ***Client File*** and displays on the UI for viewing.  **Step 7:** The system retrieves all details of any “Active” Bookings for the selected Client from the ***Booking File*** and displays on the UI for viewing.  **Step 8:** If no future Bookings exist for the Client, go to step 9.  **Step 10:** Set Client status to:   * “W” (Withdrawn)***.***   **Step 11:** The system updates the Client details in the ***Client file:***   * Client\_ID * Forename * Surname * Date\_Of\_Birth * Contact\_No * Email * Street * Town * City * County * Eircode * Status   **Step 12:** The system displays a “Client Withdrawn” confirmation message.  **Step 14:** The system sends an email to the Client confirming that their details have been withdrawn from the system.  **Step 15:** The system resets the UI. |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Client not found** |  | **Step 4:** No data is displayed.  **Step 5:** The system displays an appropriate error message.  **Step 6:** The system returns the user to re-enter Client details in step 3. |
| **Future Bookings Found** | **Step 10**: The Manager selects the Print Invoice option.  **Step 10**: The Manager selects the send email option. | **Step 8:** Future Bookings for Client found.  **Step 9:** Thesystem calculates any monies owed to the Client as outlined in “Cancel Booking” refund policy and displays on the UI.  **Step 11:** The system prints the balance displayed on the UI.  **Step 13:** The system sends an email to the Client confirming that their details have been withdrawn from the system.  **Step 14:** The system resets the UI. |
| **Conclusions** | The Client status is updated in the ***Client File***. | |
| **Post conditions** | Client details are updated to “W” (Withdrawn) and they will no longer be able to make Bookings. | |
| **Business Rules** | Only “Active” Clients may be withdrawn.  Client cannot be withdrawn if they have future Bookings. | |
| **Implementation Constraints** |  | |

### **4.1.4.** **Query Client**

This function lists the details of a Client on the system.

Manager

Client

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Query Client** | |
| **Use Case Id** | 4.1.4. | |
| **Priority** | Low | |
| **Source** | Manager | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** | Client | |
| **Description** | This function lists the details of a Client on the system. | |
| **Preconditions** |  | |
| **Trigger** |  | |
| **Expected Scenario** | **Actor Action** | **System Response** |
|  | **Step 1:** The Manager invokes the Query Client function.  **Step 3:** The Manager enters the Surname (or part of) of the Client to be queried.  **Step 5:** The Manager selects the Client to be queried. | **Step 2:** The system displays the UI.  **Step 4:** The system retrieves summary details of all “Active” Clients with matching surname from the ***Client File*** and displays on the UI in alphabetical order of Surname.  **Step 6:** The system retrieves all details of the selected Client from the ***Client File*** and displays on the UI. |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Client not found** |  | **Step 4:** No data is displayed.  **Step 5:** The system displays an appropriate error message.  **Step 6:** The system returns the user to re-enter Client details in step 3. |
| **Conclusions** | The Client details are retrieved and displayed in list form from the ***Client File***. | |
| **Post conditions** |  | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

A sample listing of the Query Client process is shown in Figure 1 - Appendix A.

## Manage Service Administration

This component will provide functions to log a Service, update a Service, and display a list of Services.

### **Log Service**

This function records the details of a Service on the system. This Service can then be booked by a Client.

Manager

<<Includes>>

[Cite your source here.]

<<Extends>>

[Cite your source here.]

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Log Service** | |
| **Use Case Id** | 4.2.1. | |
| **Priority** | High | |
| **Source** | Manager | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** | None | |
| **Description** | This function records the details of a Service on the system. This Service can then be booked by a Client. | |
| **Preconditions** |  | |
| **Trigger** |  | |
| **Expected Scenario** | **Actor Action** | **System Response** |
|  | **Step 1:** The Manager invokes the Log Service function.  **Step 3:** The Manager enters the Service details:   * Service\_ID * Description * Base\_Rate   **Step 4:** The Manager confirms that the Service is to be added. | **Step 2:** The system displays the UI.  **Step 5:** The system validates the data entered:   * All fields must be entered * Service\_ID must be length 4 and not numeric. * Description must be max length 30 and not numeric. * Base\_Rate must be decimal and max length six characters, with two decimal places.   **Step 6:** Set Service status to “A” (Active).  **Step 7:** The system saves the Service details in the ***Service file:***   * Service\_ID * Description * Base\_Rate * Status   **Step 8:** The system displays a “Service registered” confirmation message.  **Step 9:** The system resets the UI. |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Invalid Data Entered** |  | **Step 5:** Invalid data is entered.    **Step 6:** The system displays an appropriate error message.  **Step 7:** The system returns the user to re-enter Services details in step 3. |
| **Conclusions** | A new Service is created in the ***Service file***. | |
| **Post conditions** | Service is added and can then be booked. | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

### **Update Service**

This function updates the details of a Service on the system. Only “Active” Services may be updated.

Manager

<<Includes>>

[Cite your source here.]

<<Extends>>

[Cite your source here.]

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Update Service** | |
| **Use Case Id** | 4.2.2. | |
| **Priority** | Medium | |
| **Source** | Manager | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** | None | |
| **Description** | This function updates the details of a Service on the system. This Service can then continue to be booked by a Client. | |
| **Preconditions** | The Service must have previously been logged on the system, and have an “Active” status. | |
| **Trigger** | A situation arises making it necessary for a Service to be updated. | |
| **Expected Scenario** | **Actor Action** | **System Response** |
|  | **Step 1:** The Manager invokes the Update Service function.  **Step 4:** The Manager selects the Service to be updated.  **Step 6:** The Manager updates the Service details:   * Description * Base\_Rate   **Step 7:** The Manager confirms that the Service is to be updated. | **Step 2:** The system displays the UI.  **Step 3:** The system retrieves details of all “Active” Services from the ***Service File*** and displays on the UI for viewing.  **Step 5:** The system retrieves all details of the selected Service from the ***Service File*** and displays on the UI for editing.  **Step 8:** The system validates the data entered:   * All fields must be entered * Description must be max length 30 and not numeric. * Base\_Rate must be decimal and max length six characters, with two decimal places. * Status must be max length 1 and not numeric.   **Step 9:** The system updates the Service details in the ***Service file.***   * Service\_ID * Description * Base\_Rate * Status   **Step 10:** The system displays a "Service Updated” confirmation message.  **Step 11:** The system resets the UI. |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Invalid Data Entered** |  | **Step 9:** Invalid data is entered.    **Step 10:** The system displays an appropriate error message.  **Step 11:** The system returns the user to re-enter Service details in step 6. |
| **Conclusions** | The Service is updated in the ***Service file***. | |
| **Post conditions** | Service is updated and can then continue to be booked by a Client. | |
| **Business Rules** | Only “Active” Services may be updated. | |
| **Implementation Constraints** |  | |

### **Query Service**

This function lists the details of a Service on the system.

Manager

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Query Service** | |
| **Use Case Id** | 4.2.3. | |
| **Priority** | Low | |
| **Source** | Manager | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** | None | |
| **Description** | This function lists the details of a Service on the system. | |
| **Preconditions** |  | |
| **Trigger** |  | |
| **Expected Scenario** | **Actor Action** | **System Response** |
|  | **Step 1:** The manager invokes the Query Services function.  **Step 4:** The manager selects the Service to be queried. | **Step 2:** The system displays the UI.  **Step 3:** The system retrieves details of all “Active” Services from the ***Service File*** and displays on the UI for viewing.  **Step 5:** The system retrieves all details of the selected Services from the ***Service file*** and displays on the UI. |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
|  |  |  |
| **Conclusions** | The Service details are retrieved and displayed in the form of a list from the ***Service file***. | |
| **Post conditions** |  | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

A sample listing of the Query Service process is shown in Figure 2 - Appendix B.

## Manage DJ Administration

This component will provide functions to add a DJ, withdraw a DJ, and display a DJs schedule in list form.

### **Add DJ**

This function records the details of a DJ on the system. This DJ can then take Bookings.

Manager

DJ

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Add DJ** | |
| **Use Case Id** | 4.3.1. | |
| **Priority** | High | |
| **Source** | Manager | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** | DJ | |
| **Description** | This function records the details of a DJ on the system. This DJ can then take Bookings. | |
| **Preconditions** |  | |
| **Trigger** | The DJ must have completed an interview and DJ registration form. | |
| **Expected Scenario** | **Actor Action** | **System Response** |
|  | **Step 1:** The Manager invokes the Add DJ function.  **Step 4:** The Manager enters the DJ details:   * Forename * Surname * Date\_Of\_Birth * Contact\_No * Email * Street * Town * City * County * Eircode * Hourly\_Rate   **Step 5:** The Manager confirms that the DJ is to be added.  **Step 10:** The Manager selects the send email option. | **Step 2:** The system determines the next DJ\_ID  **Step 3:** The system displays the UI.  **Step 6:** The system validates the data entered:   * All fields must be entered * Forename must be max length 35 and not numeric. * Surname must be max length 35 and not numeric. * Date\_Of\_Birth must be > 18 years. * Contact\_No must be max length 20 and not numeric. * Email must be valid format. * Street must be max length 50 and not numeric. * Town must be max length 50 and not numeric. * City must be max length 50 and not numeric. * County must be max length 15 and not numeric. * Eircode must be max length 7 and not numeric. * Hourly\_Rate must be decimal and max length six characters, with two decimal places.   **Step 7**: Set DJ status to “A” (Active).  **Step 8:** The system saves the DJ details in the ***DJ file:***   * DJ\_ID * Forename * Surname * Date\_Of\_Birth * Contact\_No * Email * Street * Town * City * County * Eircode * Hourly\_Rate * Status   **Step 9:** The system displays a “DJ registered” confirmation message.  **Step 11:** The system sends an email to the DJ confirming their registration on the system.  **Step 12:** The system resets the UI. |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Invalid Data Entered** |  | **Step 6:** Invalid data is entered.    **Step 7:** The system displays an appropriate error message.  **Step 8:** The system returns the user to re-enter DJ details in step 4. |
| **Conclusions** | A new DJ is created in the ***DJ File***. | |
| **Post conditions** | DJ is added and enabled to take Bookings. | |
| **Business Rules** | A DJ must be over the age of 18 to be an employee. | |
| **Implementation Constraints** |  | |

### **Withdraw DJ**

This function changes the status of an “Active” DJ to “Withdrawn”. Once a DJ is “Withdrawn”, Bookings cannot be made for this DJ. A DJ cannot be “Withdrawn” if future Bookings exist for the DJ.

Manager

DJ

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Withdraw DJ** | |
| **Use Case Id** | 4.3.2. | |
| **Priority** | High | |
| **Source** | Manager | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** | DJ | |
| **Description** | This function withdraws the details of a DJ on the system. This DJ can then no longer take Bookings. | |
| **Preconditions** |  | |
| **Trigger** | DJ must have requested to be removed OR Manager decides to remove DJ from system. | |
| **Expected Scenario** | **Actor Action** | **Actor Action** |
|  | **Step 1:** The Manager invokes the Withdraw DJ function.  **Step 3:** The Manager enters the Surname (or part of) of the DJ to be withdrawn.  **Step 5:** The Manager selects the DJ to be withdrawn.  **Step 9:** The Manager confirms that the DJ is to be withdrawn.  **Step 13:** The Manager selects the send email option. | **Step 2:** The system displays the UI.  **Step 4:** The system retrieves summary details of all “Active”DJs with matching name from the ***DJ File*** and displays on the UI in alphabetical order of Surname.  **Step 6:** The system retrieves all details of the selected DJ from the ***DJ File*** and displays on the UI for viewing.  **Step 7:** The system retrieves all details of any future “Active” Bookings for the selected DJ from the ***Booking File*** and displays on the UI for viewing.  **Step 8:** If no future Bookings exist for the DJ, go to step 8.  **Step 10:** Set DJ status to:   * “W” (Withdrawn)***.***   **Step 11:** The system updates the DJ details in the ***DJ file:***   * DJ\_ID * Forename * Surname * Date\_Of\_Birth * Contact\_No * Email * Street * Town * City * County * Eircode * Hourly\_Rate * Status   **Step 12:** The system displays a “DJ Withdrawn” confirmation message.  **Step 14:** The system sends an email to the DJ confirming that their details have been withdrawn from the system.  **Step 15:** The system resets the UI. |
| **Alternate Scenarios** | **Actor Action** | **Actor Action** |
| **DJ not found** |  | **Step 4:** No data found  **Step 5:** The system displays an appropriate error message.  **Step 6:** The system returns the user to re-enter DJ details in step 3. |
| **Future Bookings Found** | **Step 9:** The Manager selects the Print option.  **Step 11:** The Manager selects the send email option. | **Step 8:** Future Bookings for DJ found  **Step 10:** The system prints a copy of the remaining Bookings to be fulfilled by the DJ.  **Step 12:** The system sends an email to the DJ confirming the remaining Bookings for them to fulfil before they can be withdrawn from the system.  **Step 13:** The system resets the UI. |
| **Conclusions** | The DJ status is updated in the ***DJ File***. | |
| **Post conditions** | DJ details are updated to “W” (Withdrawn) and they will no longer be able to take Bookings. | |
| **Business Rules** | Only “Active” DJs may be withdrawn.  A DJ may not be withdrawn if future Bookings exist for the DJ. | |
| **Implementation Constraints** |  | |

### **Show DJ Schedule**

This function lists the details of a DJs Schedule on the system.

Manager

DJ

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Show DJ Schedule** | |
| **Use Case Id** | 4.3.3. | |
| **Priority** | Medium | |
| **Source** | Manager | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** | DJ | |
| **Description** | This function lists the details of a DJs Schedule on the system. | |
| **Preconditions** |  | |
| **Trigger** |  | |
| **Expected Scenario** | **Actor Action** | **System Response** |
|  | **Step 1:** The Manager invokes the Show DJ Schedule function.  **Step 4:** The Manager enters the Surname (or part of) of the DJ to be queried.  **Step 6:** The Manager selects the DJ to be queried. | **Step 2:** The system displays the UI.  **Step 3:** The system retrieves details of all “Active” DJs from the ***DJ File*** and displays on the UI for viewing.  **Step 5:** The system retrieves summary details of all “Active” DJs with matching Surname from the ***DJ File*** and displays on the UI in alphabetical order of Surname.  **Step 7:** The system retrieves all details of the selected DJ from the ***DJ File*** and displays on the UI for viewing.  **Step 8:** The system retrieves all details of any future “Active” Bookings for the selected DJ from the ***Booking File*** and displays on the UI for viewing. |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **DJ not found** |  | **Step 5:** No data is displayed.  **Step 6:** The system displays an appropriate error message.  **Step 7:** The system returns the user to re-enter DJ details in step 4. |
| **Conclusions** | The DJ Schedule details are retrieved and displayed in the form of a list from both the ***DJ File,*** *and the* ***Booking file***. | |
| **Post conditions** |  | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

A sample listing of the Show DJ Schedule process is shown in Figure 3 - Appendix C.

## Process Booking Details

This component will provide functions to make a Booking, cancel a Booking, and change a Booking.

### **Make Booking**

This function enables a Client who is registered and “Active” on the system to make a Booking. All Bookings are paid for in full at the time the Booking is made.

|  |  |
| --- | --- |
| Activity Diagram: Make Booking | |
| Manager | System |
| Select Client for Booking  Manager enters Client Surname (part of)  Invoke Make Booking Function | Assign Booking\_ID  Retrieve Details of all “Active” Services from ***Service file*** and display on UI  Yes  No  Error Message  Valid  Retrieve Details of selected Client from ***Client file*** and display on UI  Retrieve Details of “Active” Clients from ***Client file*** with matching Surname in order of Surname, Forename  Display UI |
| Error Message  Yes  Valid  No  Enter Booking Details:   * Event\_Date * Service\_ID * DJ\_ID | Retrieve Details of all “Active” Bookings from ***Booking file*** in order of Event-Date, DJ\_ID, Service\_ID, and display on UI  Validate Booking Details  Retrieve Details of all “Active” DJs from ***DJ file*** and display on UI |
| Enter remaining Booking Details:   * Start\_Time * End\_Time * Duration * Location * Eircode   Manager Confirms Booking to be processed | No  Yes  No  Error Message  Valid  Validate Booking Details  Error Message  Yes  Available  System checks to see if Event\_Date, DJ\_ID, and Service\_ID are available |
|  | Set Booking Status to “A” (Active)  Set Payment Status to “P” (Paid)  Set Booking Date to = System Date  System calculates Total\_Cost and displays on UI |
| Manager selects the send email option | System sends Client an email confirming the Booking has been processed  Reset UI  System displays “Booking Confirmed” message on UI  Save Booking Details in ***Booking file:***   * Booking\_ID * Booking\_Date * Event\_Date * Start\_Time * End\_Time * Duration * Location * Eircode * Total\_Cost * Payment\_Status * Refund\_Percentage * Refund\_Due * Booking\_Status * Client\_ID * Service\_ID * DJ\_ID |

Manager

Client

<<Includes>>

[Cite your source here.]

<<Includes>>

[Cite your source here.]

<<Extends>>

[Cite your source here.]

<<Extends>>

[Cite your source here.]

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Make Booking** | |
| **Use Case Id** | 4.4.1. | |
| **Priority** | High | |
| **Source** | Client | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** | Client | |
| **Description** | This function enables a Client who is registered, and “Active”, on the system to make a Booking. | |
| **Preconditions** | The Client, the Service, and the DJ, must all be must be registered on the system, and have an “Active” status. | |
| **Trigger** |  | |
| **Expected Scenario** | **Actor Action** | **System Response** |
|  | **Step 1:** The Manager invokes the Make Booking function.  **Step 4:** The Manager enters the Surname (or part of) of the Client.  **Step 6:** The Manager selects the Client for the Booking.  **Step 10:** The Manager enters the requested data:   * Event\_Date * Service\_ID * DJ\_ID   **Step 14:** The Manager enters the remaining Booking details:   * Start\_Time * End\_Time * Duration * Location * Eircode   **Step 15:** The Manager confirms that the Booking is to be processed.  **Step 23:** The Manager selects the send email option. | **Step 2:** The system determines the next Booking\_ID.  **Step 3:** The system displays the UI.  **Step 5:** The system retrieves summary details of all “Active” Clients with matching surname from the ***Client File*** in order of Surname, Forename.  **Step 7:** The system retrieves all details for the selected Client from the ***Client File*** and displays on the UI for viewing.  **Step 8:** The system retrieves details of all “Active” Services from the ***Service File*** and displays on the UI for viewing.  **Step 9:** The system retrieves details of all “Active” DJs from the ***DJ File*** in order of Surname, Forename and displays on the UI for viewing.  **Step 11:** The system validates the data entered:   * All fields must be entered * Event\_Date must be valid format. * Service\_ID must be length 4 and not numeric. * DJ\_ID must be max length 3.   **Step 12:** The system retrieves details of all “Active” Bookings from the ***Booking File*** in order of Event\_Date, DJ\_ID, Service\_ID, and displays on the UI for viewing.  **Step 13:** If Event\_Date, DJ\_ID, and Service\_ID are available, go to step 14.  **Step 16:** The system validates the data entered:   * All fields must be entered * Event\_Date must be valid format. * Start\_Time must be valid format. * End\_Time must be valid format. * Duration must be numeric and max length 2. * Location must be max length 200 and not numeric. * Eircode must be max length 7 and not numeric.   **Step 17:** The system calculates total cost and displays on UI. (The Total\_Cost is calculated by the following sum):   * **Service base rate + (DJ hourly rate \* Booking duration)**   **Step 18:** Payment\_Status set to:   * “P” (Paid)   **Step 19:** Booking\_Status set to:   * “A” (Active)   **Step 20:** Booking\_Date set to:   * = System Date   **Step 21:** The system saves the Booking details in the ***Booking file.***   * Booking\_ID * Booking\_Date * Event\_Date * Start\_Time * End\_Time * Duration * Location * Eircode * Total\_Cost * Payment\_Status * Refund\_Percentage * Refund\_Due * Booking\_Status * Client\_ID * Service\_ID * DJ\_ID   **Step 22:** The system displays a “Booking Made” confirmation message.  **Step 24:** The system sends an email to the Client confirming that their Booking has been registered on the system.  **Step 25:** The system resets the UI. |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Client not found** |  | **Step 5:** No data is displayed.  **Step 6:** The system displays an appropriate error message.  **Step 7:** The system returns the user to re-enter Client details in step 4. |
| **Invalid Data Entered** |  | **Step 11:** Invalid data is entered.    **Step 12:** The system displays an appropriate error message.  **Step 13:** The system returns the user to re-enter event date in step 10. |
| **Availability** |  | **Step 13:** Either Event\_Date, DJ\_ID, or Service\_ID are unavailable.    **Step 14:** The system displays an appropriate error message.  **Step 15:** The system returns the user to re-enter details in step 10. |
| **Invalid Data Entered** |  | **Step 16:** Invalid data is entered.    **Step 17:** The system displays an appropriate error message.  **Step 18:** The system returns the user to re-enter remaining Booking details in step 14. |
| **Conclusions** | A new Booking is created in the ***Booking file.*** | |
| **Post conditions** | Booking is added, and data is updated in all appropriate files. | |
| **Business Rules** | All money due must be paid in full at time of Booking. | |
| **Implementation Constraints** |  | |

### **Cancel Booking**

This function enables a Client who is registered, and “Active”, on the system to cancel a Booking.

Manager

Client

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Cancel Booking** | |
| **Use Case Id** | 4.4.2. | |
| **Priority** | High | |
| **Source** | Client | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** | Client | |
| **Description** | This function enables a Client who is registered, and “Active”, on the system to cancel a Booking. | |
| **Preconditions** | Client must have requested to cancel the Booking OR manager decides to cancel Booking from system. | |
| **Trigger** |  | |
| **Expected Scenario** | **Actor Action** | **System Response** |
|  | **Step 1:** The Manager invokes the Cancel Booking function.  **Step 3:** The Manager enters the Surname (or part of) of the Client.  **Step 5:** The Manager selects the Client whose Booking is to be cancelled.  **Step 8:** The Manager selects the Booking to be cancelled.  **Step 10:** The Manager confirms that the Booking is to be cancelled.  **Step 13:** The Manager selects the Print Invoice option.  **Step 23:** The Manager selects the send email option. | **Step 2:** The system displays the UI.  **Step 4:** The system retrieves summary details of all “Active” Clients with matching Surname from the ***Client File*** in order of Surname, Forename.  **Step 6:** The system retrieves all details for the selected Client from the ***Client File*** and displays on the UI for viewing.  **Step 7:** The system retrieves summary details of all “Active” Bookings made by the Client from the ***Booking File*** and displays on the UI in order of Event\_Date, Booking\_ID, and Client\_ID.  **Step 9:** The system retrieves all details of the selected Booking from the ***Booking File*** and displays on the UI.  **Step 11:** The system calculates if any refund is due to the Client as outlined in business rules.  (The Refund\_Percentage is calculated by checking the event date and the current system date, and then determining which percentage of a refund (if any) is due to the Client.  **Step 12:** The system calculates the Refund\_Due and displays on the UI.  (Refund\_Due is calculated by the following sum):   * **(Total Cost / 100) \* Refund\_Percentage**   **Step 14:** The system prints the balance displayed on the UI.  **Step 15:** The system recalculates the total cost and saves to the ***Booking file***.  (The Total\_Cost is recalculated by the following sum):   * **Total Cost - Refund\_Due**   **Step 16:** Booking\_Status set to:   * “C” (Cancelled).   **Step 17:** Payment\_Status set to:   * “R” (Refunded).   **Step 18:** Event\_Date set to:   * “NULL”.   **Step 19:** Start\_Time set to:   * “NULL”.   **Step 20:** End\_Time set to:   * “NULL”.   **Step 21:** The system saves the updated Booking details in the ***Booking file.***   * Booking\_ID * Booking\_Date * Event\_Date * Start\_Time * End\_Time * Duration * Location * Eircode * Total\_Cost * Payment\_Status * Refund\_Percentage * Refund\_Due * Booking\_Status * Client\_ID * Service\_ID * DJ\_ID   **Step 22:** The system displays a “Booking Cancelled” confirmation message.  **Step 24:** The system sends an email to the Client confirming the Booking cancellation and any refund due to them.  **Step 25:** The system resets the UI. |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Client not found** |  | **Step 4:** No data is displayed.  **Step 5:** The system displays an appropriate error message.  **Step 6:** The system returns the user to re-enter Client details in step 3. |
| **Conclusions** | The Booking is cancelled, and data is saved in the ***Booking file.*** | |
| **Post conditions** | Booking is cancelled, and data is updated in all appropriate files. | |
| **Business Rules** | If Client cancels more than 28 days before event date, a 100% refund is returned.  If Client cancels between 14 and 27 days before event date, a 75% refund is returned.  If Client cancels between 7 and 13 days before event date, a 50% refund is returned.  If Client cancels less than 7 days before event date, no refund is returned.  If Manager cancels Booking before event date, a 100% refund is returned. | |
| **Implementation Constraints** |  | |

### **Change Booking**

This function enables a Client who is registered, and “Active”, on the system to change a Booking.

Manager

Client

<<Includes>>

[Cite your source here.]

<<Includes>>

[Cite your source here.]

<<Extends>>

[Cite your source here.]

<<Extends>>

[Cite your source here.]

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Change Booking** | |
| **Use Case Id** | 4.4.3. | |
| **Priority** | Medium | |
| **Source** | Client | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** | Client | |
| **Description** | This function enables an “Active” Booking previously recorded on the system to be changed. | |
| **Preconditions** | The Client and the DJ must both be registered on the system, and have an “Active” status. | |
| **Trigger** |  | |
| **Expected Scenario** | **Actor Action** | **System Response** |
|  | **Step 1:** The Manager invokes the Change Booking function.  **Step 3:** The Manager enters the Surname (or part of) of the Client.  **Step 5:** The Manager selects the Client whose Booking is to be changed.  **Step 8:** The Manager selects the Booking to be changed.  **Step 11:** The Manager enters the requested change of DJ details:   * DJ\_ID   **Step 15:** The manager enters the Booking details eligible to be changed:   * Start\_Time * End\_Time * Duration * Location * Eircode   **Step 16:** The manager confirms that the Booking is to be changed. | **Step 2:** The system displays the UI.  **Step 4:** The system retrieves summary details of all “Active” Clients with matching Surname from the ***Client File*** in order of Surname, Forename.  **Step 6:** The system retrieves all details for the selected Client from the ***Client File*** and displays on the UI for viewing.  **Step 7:** The system retrieves summary details of all “Active” Bookings made by the Client from the ***Booking File*** and displays on the UI in order of Event\_Date, Booking\_ID, and Client\_ID.  **Step 9:** The system retrieves all details of the selected Booking from the ***Booking File*** and displays on the UI.  **Step 10:** The system retrieves details of all “Active” DJs from the ***DJ File*** in order of surname, forename and displays on the UI for viewing.  **Step 12:** The system validates the data entered:   * All fields must be entered * DJ\_ID must be max length 3.   **Step 13:** The system retrieves details of all “Active” Bookings from the ***Booking File*** in order of Event\_Date, DJ\_ID, and displays on the UI for viewing.  **Step 14:** If Event\_Date and DJ\_ID are available, go to step 15.  **Step 17:** The system validates the data entered:   * All fields eligible must be entered * Start\_Time must be valid format. * End\_Time must be valid format. * Duration must be numeric and max length 2. * Location must be max length 200 and not numeric. * Eircode must be max length 7 and not numeric.   **Step 18:** Booking\_Date re-set to:  = System Date  **Step 19:** System calculates new total cost and displays on UI. (The Total\_Cost is calculated by the following sum):  **Service base rate + (DJ hourly rate \* Booking duration)**  **Step 20:** The system saves the new Booking details in the ***Booking file.***   * Booking\_ID * Booking\_Date * Event\_Date * Start\_Time * End\_Time * Duration * Location * Eircode * Total\_Cost * Payment\_Status * Refund\_Percentage * Refund\_Due * Booking\_Status * Client\_ID * Service\_ID * DJ\_ID   **Step 21:** The system displays a “Booking Changed” confirmation message.  **Step 22:** The system clears the UI. |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Client not found** |  | **Step 4:** No data is displayed.  **Step 5:** The system displays an appropriate error message.  **Step 6:** The system returns the user to re-enter Client details in step 3. |
| **Invalid Data Entered** |  | **Step 12:** Invalid data is entered.    **Step 13:** The system displays an appropriate error message.  **Step 14:** The system returns the user to re-enter Booking details in step 11. |
| **DJ status displayed as “Unavailable”** |  | **Step 14:** The DJ\_ID is not available to be Booked.  **Step 15:** The system displays an appropriate error message.  **Step 16:** The system returns the user to re-enter DJ details in step 11. |
| **Invalid Data Entered** |  | **Step 17:** Invalid data is entered.    **Step 18:** The system displays an appropriate error message.  **Step 19:** The system returns the user to re-enter Booking details in step 15. |
| **Conclusions** | The Booking is changed and saved in the ***Booking file.*** | |
| **Post conditions** | Booking is changed, and data is updated in all appropriate files. | |
| **Business Rules** | Client must give at least 28 days’ notice to change a Booking. | |
| **Implementation Constraints** |  | |

## Perform Administration

This component will provide functions to analyse revenue by both Year and Service.

### **Analyse Revenue by Year**

This function provides an analysis of the yearly revenue in graph form.

Manager

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Analyse Revenue by Year** | |
| **Use Case Id** | 4.5.1. | |
| **Priority** | Medium | |
| **Source** | Manager | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** | None | |
| **Description** | This function provides an analysis of the yearly revenue in graph form. | |
| **Preconditions** |  | |
| **Trigger** |  | |
| **Expected Scenario** | **Actor Action** | **System Response** |
|  | **Step 1:** The Manager invokes the Analyse Revenue by Year function.  **Step 3:** The Manager selects the year to analyse.  **Step 5:** The Manager selects the print option. | **Step 2:** The system displays the UI.  **Step 4:** The system retrieves all details of the selected years’ Bookings from the ***Booking file*** and displays as a graph on the UI.  **Step 6:** The system prints the graph  **Step 7:** The system resets the UI. |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
|  |  |  |
| **Conclusions** | A graph of the yearly revenue is produced. | |
| **Post conditions** |  | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

A sample graph of the Analyse Revenue by Year process is shown in Figure 4 - Appendix D.

### **Analyse Revenue by Service**

This function provides an analysis of revenue specific to a Service in graph form. This will enable the Manager to determine the most cost beneficial Services.

Manager

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Analyse Revenue by Service** | |
| **Use Case Id** | 4.5.2. | |
| **Priority** | Medium | |
| **Source** | Manager | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** | None | |
| **Description** | This function provides an analysis of revenue specific to a Service in graph form. This will enable the Manager to determine the most cost beneficial Services. | |
| **Preconditions** |  | |
| **Trigger** |  | |
| **Expected Scenario** | **Actor Action** | **System Response** |
|  | **Step 1:** The Manager invokes the Analyse Revenue by Service function.  **Step 3:** The Manager selects the year to analyse.  **Step 5:** The Manager selects the print option. | **Step 2:** The system displays the UI.  **Step 4:** The system retrieves all details of the selected years’ Bookings from both the ***Booking file and the Service file*** and displays as a graph on the UI.  **Step 6:** The system prints the graph.  **Step 7:** The system resets the UI. |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
|  |  |  |
| **Conclusions** | A graph of the monthly revenue by Service over 12 months is produced. | |
| **Post conditions** |  | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

A sample graph of the Analyse Revenue by Service process is shown in Figure 5 - Appendix E.

# System Model

The following dataflow diagrams have been produced for the system:

## Level-0 DFD

Service

DJ Booking System

Client

Booking Request

## Level-1 DFD

DJ Details

D3

DJ File

Client

Client Details

Booking Request

P3

Manage DJs

DJ Details

P1

Manage Clients

Booking Details

Client Details

D1

Client File

Booking Details

Booking Details

D1

Booking File

Client Details

P4

Process Bookings

D1

Booking File

Booking Details

Service Details

D2

Service File

P5

Perform Analysis

P2

Manage Services

Service Details

Service Details

## Level-2 DFD (Process P1: Manage Clients)

New Client Details

Client

P1.2

Update Client

Client Details

Client Details

Client Details

P1.1

Add Client

Modified Client Details

D1

Client File

Client Details

Modified Client Details

Client Details

Client Details

P1.3

Withdraw Client

P1.4

Query Client

D4

Booking File

Booking Details

## Level-2 DFD (Process P2: Manage Services)

P2.3

Query Service

P2.1

Log Service

Service Details

Service Details

Modified Service Details

D2

Service File

Service Details

P2.2

Update Service

## Level-2 DFD (Process P3: Manage DJs)

P3.1

Add DJ

P3.3

Show DJ Schedule

DJ Details

DJ Details

Booking Details

D3

DJ File

D4

Booking File

DJ Details

Modified DJ Details

P3.2

Withdraw DJ

Booking Details

## Level-2 DFD (Process P4: Process Bookings)

Client

DJ Details

DJ Details

D3

DJ File

Booking Request

P4.1

Make Booking

P4.3

Change Booking

Booking Details

Client Details

Booking Details

D1

Client File

Modified Booking Details

Service Details

D1

Client File

Booking Details

D4

Booking File

Booking Details

Client Details

D2

Service File

P4.2

Cancel Booking

Modified Booking Details

Client Details

Client

## Level-2 DFD (Process P5: Perform Analysis)

P5.1

Perform Yearly Analysis

P5.2

Perform Service Analysis

D2

Service File

Service Details

Booking Details

Booking Details

D4

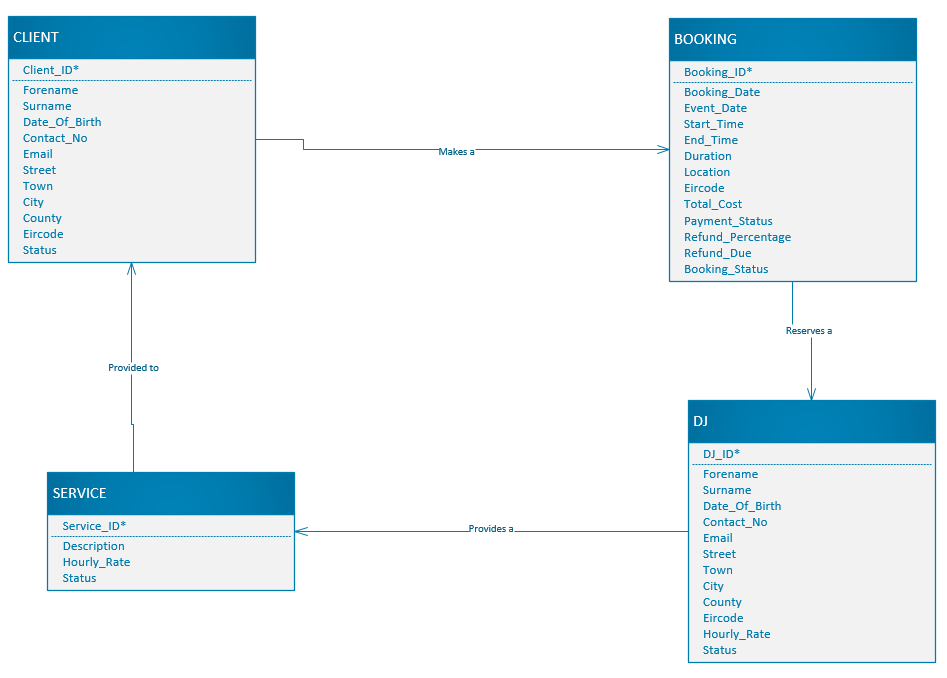
Booking File

# Data Model (Class Diagram)

The data model outlines the structure flow of the DJ Booking System by displaying the following:

* **Class Diagram**
* **Relational Schema**
* **Database Schema**

## Class Diagram



## Relational Schema

Client (Client\_ID {PK}, Forename, Surname, Date\_Of\_Birth, Contact\_No, Email, Street, Town, City, County, Eircode, Status)

DJs (DJ\_ID {PK}, Forename, Surname, Date\_Of\_Birth, Contact\_No, Email, Street, Town, City, County, Eircode, Hourly\_Rate, Status)

Services (Service\_ID {PK}, Description, Hourly\_Rate, Status)

Booking (Booking\_ID {PK}, Booking\_Date, Event\_Date, Start\_Time, End\_Time, Duration, Location, Eircode, Total\_Cost, Payment\_Status, Refund\_Percentage, Refund\_Due, Booking\_Status, Client\_ID, DJ\_ID, Service\_ID)

## Database Schema

**Schema: DJSys**

**Relation: Client**

Attributes:

Client\_ID smallint (5) UNIQUE NOT NULL

Forename varchar2 (35) NOT NULL

Surname varchar2 (35) NOT NULL

Date\_Of\_Birth date NOT NULL

Contact\_No varchar2 (20) NOT NULL

Email varchar2 (255) NOT NULL

Street varchar2 (50) NOT NULL

Town varchar2 (50) NOT NULL

City varchar2 (50)

County varchar2 (15) NOT NULL

Eircode varchar2 (7)

Status varchar2 (9) NOT NULL

**Primary Key:** Client\_ID

**Relation:** DJs

Attributes:

DJ\_ID tinyint (3) UNIQUE NOT NULL

Forename varchar2 (35) NOT NULL

Surname varchar2 (35) NOT NULL

Date\_Of\_Birth date NOT NULL

Contact\_No varchar2 (20) NOT NULL

Email varchar2 (255) NOT NULL

Street varchar2 (50) NOT NULL

Town varchar2 (50) NOT NULL

City varchar2 (50)

County varchar2 (15) NOT NULL

Eircode varchar2 (7)

Hourly\_Rate decimal (3, 2) NOT NULL

Status varchar2 (9) NOT NULL

**Primary Key:**  DJ\_ID

**Relation:** Services

Attributes:

Service\_ID char (4) NOT NULL

Description varchar2 (30) NOT NULL

Hourly\_Rate decimal (5, 2) NOT NULL

Status varchar2 (9) NOT NULL

**Primary Key:** Service\_ID

**Relation:** Bookings

Attributes:

Booking\_ID smallint (5) UNIQUE NOT NULL

Booking\_Date date NOT NULL

Event\_Date date NOT NULL

Start\_Time time NOT NULL

End\_Time time NOT NULL

Duration smallint (2) NOT NULL

Location varchar2 (200) NOT NULL

Eircode varchar2 (7) NOT NULL

Total\_Cost decimal (6, 2) NOT NULL

Payment\_Status varchar2 (8) NOT NULL

Refund\_Percentage decimal (3, 2) NOT NULL

Refund\_Due decimal (6, 2) NOT NULL

Booking\_Status varchar2 (9) NOT NULL

Client\_ID smallint (5) UNIQUE NOT NULL

DJ\_ID tinyint (3) UNIQUE NOT NULL

Service\_ID char (4) UNIQUE NOT NULL

**Primary Key:** Booking\_ID

**Foreign Key:** Client\_ID REFERENCES Clients (Client\_ID)

**Foreign Key:** DJ\_ID REFERENCES DJs (DJ\_ID)

**Foreign Key:** Service\_ID REFERENCES Services (Service\_ID)

# Conclusion

The “DJSys” project was designed to facilitate the management of a DJ hire business. It enables Clients to Book a DJ across a variety of Services, and for the administrator to maintain the running of these processes using the following functions:

* Add Client
* Update Client
* Withdraw Client
* Query Client
* Log Service
* Update Service
* Query Service
* Add DJ
* Withdraw DJ
* Show DJ Schedule
* Make Booking
* Cancel Booking
* Change Booking
* Analyse Revenue by Year
* Analyse Revenue by Service

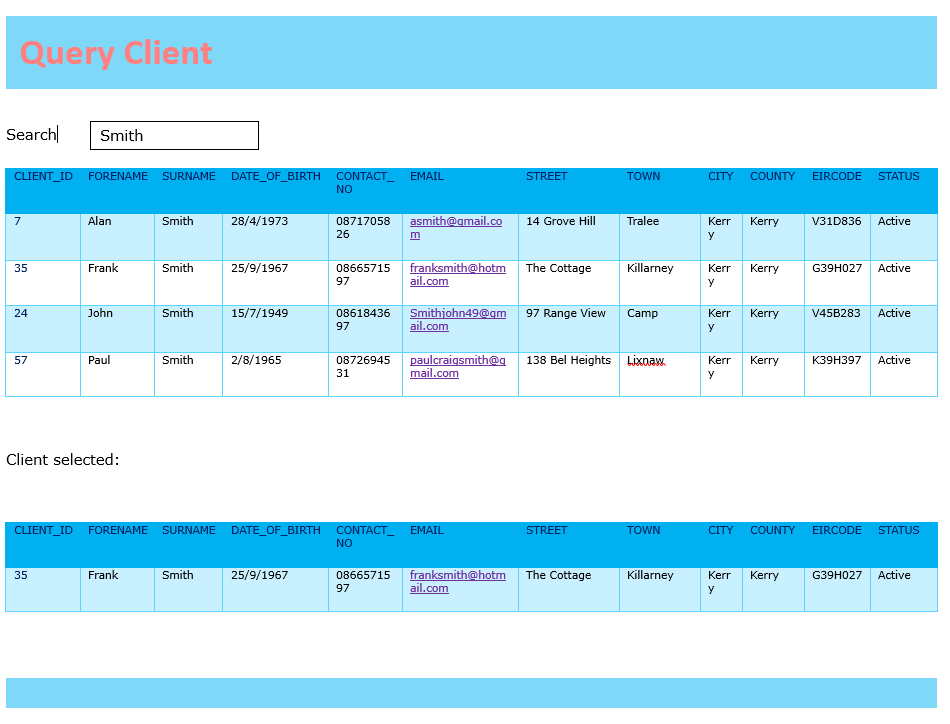
In conclusion, I am relatively pleased with the functionality of the system. It is easy to use and provides the key requirements needed for the process to run effectively.

If I were to make any changes it would be to identify and remove non-essential components which would allow for a more efficient system, and better use of resources. The area this would most relate to would be the making and cancellation of Bookings.

Another thing to look at would be enabling the user themselves to have more interactivity with the system, such as making changes through an online interface. This, would in turn, free up the administrator to perform other duties.

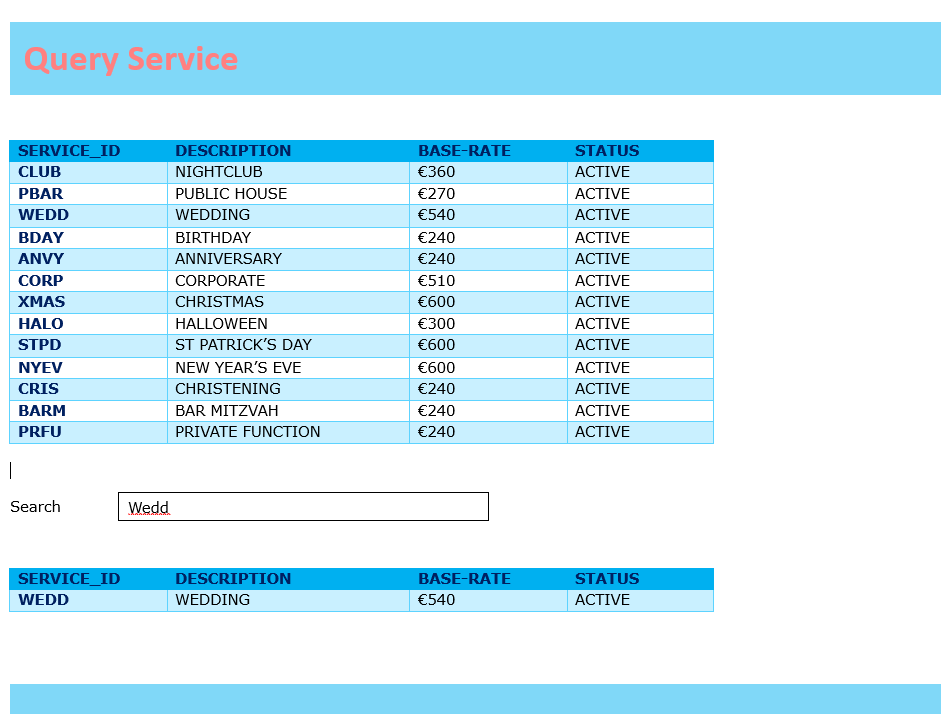
# Appendices

## Appendix A – Query Client

Figure 1 – Sample listing of Query Client process (User ID 4.1.4.).

## Appendix B – Query Service

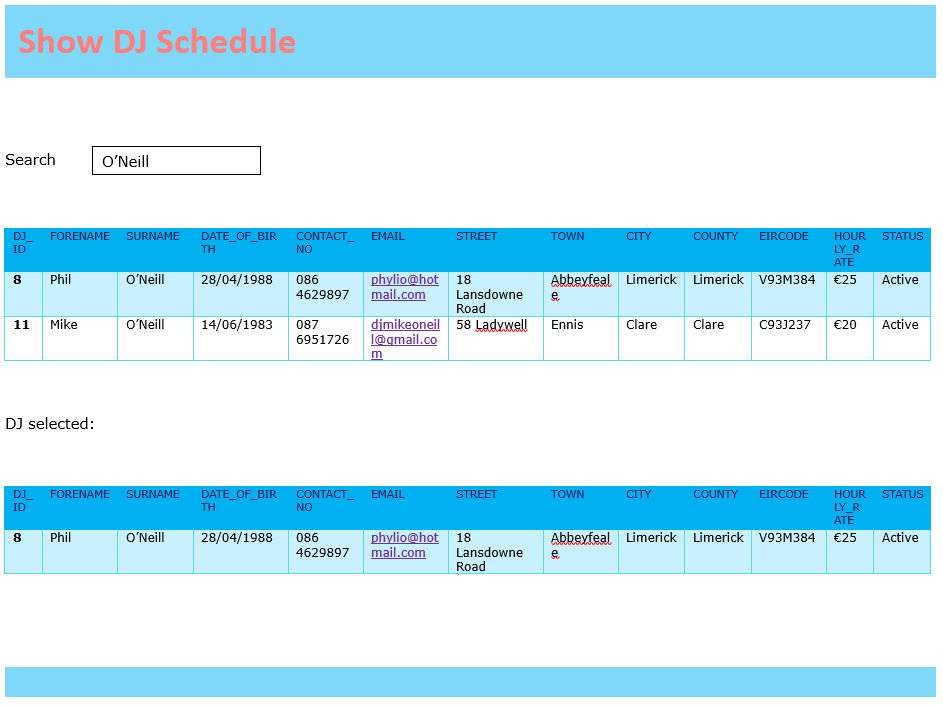
Figure 2 – Sample listing of Query Service process (User ID 4.2.3.).



## Appendix C – Show DJ Schedule

Figure 3 – Sample listing of Show DJ Schedule process (User ID 4.3.3.).







## Appendix D – Analyse Revenue by Year

Figure 4 – Sample graph of Analyse Revenue by Year process (User ID 4.5.1.).

## Appendix E – Analyse Revenue by Service

Figure 5 – Sample graph of Analyse Revenue by Service process (User ID 4.5.2.).