Requirements Engineering Project

Kestrel Falconry

Submitted By: Kass Smith T00240394

Computing with Games Development

Date Submitted: 21/04/2024

**Table of Contents**

[1. Introduction/overview 4](#_Toc164602377)

[2. Functional Components 5](#_Toc164602378)

[3. User Requirements 6](#_Toc164602379)

[3.1. FalconrySYS will manage Birds 6](#_Toc164602380)

[3.2. FalconrySYS will manage Trainers 6](#_Toc164602381)

[3.3. FalconrySYS will process Bookings 6](#_Toc164602382)

[3.4. FalconrySYS will perform administrative reporting 6](#_Toc164602383)

[4. System Requirements 7](#_Toc164602384)

[4.1. System Level Use Case Diagram 7](#_Toc164602385)

[4.2. Manage Birds 8](#_Toc164602386)

[4.2.1. Register Bird 8](#_Toc164602387)

[4.2.2. Update Bird 10](#_Toc164602388)

[4.2.3. Deregister Bird 12](#_Toc164602389)

[4.3. Manage Trainers 13](#_Toc164602390)

[4.3.1. Register Trainer 13](#_Toc164602391)

[4.3.2. Update Trainer 15](#_Toc164602392)

[4.3.3. Deregister Trainer 17](#_Toc164602393)

[4.4. Process Bookings 18](#_Toc164602394)

[4.4.1. Set Experience Type 18](#_Toc164602395)

[4.4.2. Update Experience Type 20](#_Toc164602396)

[4.4.3. Create Booking 22](#_Toc164602397)

[4.4.4. Update Booking 25](#_Toc164602398)

[4.4.5. Remove Booking 27](#_Toc164602399)

[4.4.6. Rate Booking 29](#_Toc164602400)

[4.5. Perform Administrative Reporting 31](#_Toc164602401)

[4.5.1. Analyse Yearly Revenue 31](#_Toc164602402)

[4.5.2. Analyse Yearly Experience 33](#_Toc164602403)

[5. System Model 35](#_Toc164602404)

[5.1. Level-0 DFD 35](#_Toc164602405)

[5.2. Level-1 DFD 36](#_Toc164602406)

[5.3. Level-2 DFD (Process P1: Manage Birds) 37](#_Toc164602407)

[5.4. Level-2 DFD (Process P2: Manage Trainers) 38](#_Toc164602408)

[5.5. Level-2 DFD (Process P3: Process Bookings) 39](#_Toc164602409)

[5.6. Level-2 DFD (Process P3: Perform Administrative Reporting) 40](#_Toc164602410)

[6. Data Model (Class Diagram) 41](#_Toc164602411)

[6.1. Class Diagram 41](#_Toc164602412)

[6.2. Relational Schema 42](#_Toc164602413)

[6.3. Database Schema 43](#_Toc164602414)

[7. Conclusion 45](#_Toc164602415)

# Introduction/overview

FalconrySYS is my database system designed for falconry management for Kestrel Falconry. The system is made from seven key tables: Birds, Trainers, Experiences, Bookings, Ratings, Species, and Genders. Each of these tables is dedicated to capturing and organizing data in their respective areas. The Birds table is a comprehensive record of bird details, while the Trainers table holds all necessary information about the trainers. The Experiences table is a catalogue of the various customer experiences available for booking.

FalconrySYS makes sure recording and retriving data is simple and efficient. For example, the Birds table holds all the information about the birds is the falconry, including species, age, and gender. The Trainers table, on the other hand, focuses on the human aspect of falconry, documenting the basic details of the trainer such as their date of birth, name and gender. The Experiences table provides information each different falconry experience that can be booked.

The system also includes a Bookings table for tracking customer reservations and a Ratings table for recording customer feedback. The Species table is holds a list of different bird species, while the Genders table holds the genders for both birds and trainers. This structured database model is a fundamental tool for managing falconry data, ensuring clear and efficient management of each aspect of this fascinating field.

The Bookings table in FalconrySYS, records every customer interaction. This allows for efficient tracking of bookings, cancellations, and updates. The Ratings table provides an easy way for customers to share their feedback, which can improve the overall falconry experience. In summary, FalconrySYS offers a structured way to manage all the details of a falconry in an easy and concise manner.

# Functional Components

# User Requirements

## FalconrySYS will manage Birds

* + 1. FalconrySYS will register a bird
    2. FalconrySYS will update a bird
    3. FalconrySYS will deregister a bird

## FalconrySYS will manage Trainers

* + 1. FalconrySYS will register a trainer
    2. FalconrySYS will update a trainer
    3. FalconrySYS will deregister a trainer

## FalconrySYS will process Bookings

* + 1. FalconrySYS will set the experience type
    2. FalconrySYS will update the experience type
    3. FalconrySYS will create a booking
    4. FalconrySYS will update a booking
    5. FalconrySYS will remove a booking

## FalconrySYS will perform administrative reporting

* + 1. FalconrySYS will produce a yearly revenue analysis
    2. FalconrySYS will produce a yearly experience analysis

# System Requirements

The system requirements of the system are presented in this section of the document.

The main modules in the system are Manage Birds, Manage Trainers, Process Bookings and Perform Administration.

## System Level Use Case Diagram

The following system level use case diagram illustrates the high-level system requirements.

Manager

Trainer

Receptionist

Customer

## Manage Birds

This module provides functions to register, update and deregister a bird.

### **Register Bird**

This function registers a bird in the falconry.

Receptionist

Manager

<<included>>

<<extends>>

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | Register Bird | |
| **Use Case Id** | 01 | |
| **Priority** | High | |
| **Source** | Manager | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** | Receptionist | |
| **Description** | This function registers the details of a Bird. Registered birds become part of the system's database, enabling trainers and administrators to manage them efficiently. | |
| **Expected Scenario** | **Manager** | **System** |
|  | **Step 1:** Invoke Register Bird function.  **Step 5:** Insert the required data:   * Name * SpeciesID * DOB * GenderID   **Step 8:** Confirm. | **Step 2:** Retrieve gender details from Gender File.  **Step 3:** Retrieve species details from Species File and load on UI.  **Step 4:** Display UI.  **Step 6:** Validate the data:   * All fields must be entered * Name must not be numeric * SpeciesID must not be numeric * DOB must be a valid date. * GenderID must not be numeric   **Step 7:** Display confirmation message.  **Step 9:** Assign next BirdID  **Step 10:** Set bird status to ‘A’ for available.  **Step 11:** Register bird details in the Bird file:   * BirdID * Name * SpeciesID * DOB * Status * GenderID   **Step 12:** Reset UI. |
| **Alternate Scenarios** | **Manager** | **System** |
| **Invalid Data Entered** |  | **Step 6:** Invalid data detected.  **Step 7:** Display an appropriate error message.  **Step 8:** Return to Step 5. |
| **Conclusions** | The bird’s details have been registered. | |
| **Post conditions** | The bird’s details can be updated.  The bird can be deregistered. | |
| **Business Rules** | Bird ID must not exist in the database | |

### **Update Bird**

This function updates the details of a bird in the falconry.

Receptionist

Manager

<<included>>

<<extends>>

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | Update Bird | |
| **Use Case Id** | 02 | |
| **Priority** | Low | |
| **Source** | Manager | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** | Receptionist | |
| **Description** | This function updates the details of a Bird. This is used if the original gender of the bird was incorrect, or the bird has aged. | |
| **Expected Scenario** | **Manager** | **System** |
|  | **Step 1:** Invoke Update Bird function.  **Step 6:** Select the species of the bird that is to be updated.  **Step 7:** Select the name of the bird that is to be updated  **Step 8:** Change the required data:   * Name * SpeciesID * DOB * Status * GenderID   **Step 10:** Confirm. | **Step 2:** Retrieve summary of bird details from Bird File  **Step 3:** Retrieve gender details from Gender File  **Step 4:** Retrieve species details from Species File and load on UI.  **Step 5:** Display UI.  **Step 9:** Validate the data:   * All fields must be entered. * Name must not be numeric. * SpeciesID must not be numeric. * DOB must be a valid date. * Status must be ‘A’ for available or ‘U’ for unavailable. * GenderID must not be numeric.   **Step 11:** Update selected bird details in the Bird file:   * Name * Species * DOB * Status * Gender   **Step 12:** Show confirmation message.  **Step 13:** Reset UI. |
| **Alternate Scenarios** | **Manager** | **System** |
| **Invalid Data Entered** |  | **Step 9:** Invalid data detected.  **Step 10:** Display an appropriate error message.  **Step 11:** Return to Step 8. |
| **Conclusions** | The bird’s details have been updated. | |

### **Deregister Bird**

This function deregisters a bird from the falconry

Manager

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | Deregister Bird | |
| **Use Case Id** | 03 | |
| **Priority** | Medium | |
| **Source** | Manager | |
| **Primary Business Actor** | Manager | |
| **Description** | This function deregisters a bird from the falconry. This is used if the bird has passed, has been let go, or has been moved to a different falconry. | |
| **Expected Scenario** | **Manager** | **System** |
|  | **Step 1:** Invoke Deregister Bird function.  **Step 5:** Select the species of the bird that is to be deregistered.  **Step 6:** Select the name of the bird that is to be deregistered.  **Step 7:** Confirm. | **Step 2:** Retrieve summary of bird details from Bird File  **Step 3:** Retrieve species details from Species File and load on UI.  **Step 4:** Display UI.  **Step 8:** Set bird status to ‘D’ for deregistered.  **Step 9:** Display confirmation message  **Step 10:** Reset UI. |
| **Conclusions** | The bird has been deregistered from the falconry | |

## Manage Trainers

### **Register Trainer**

This function registers a trainer in the falconry for employment.

Manager

<<included>>

<<extends>>

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | Register Trainer | |
| **Use Case Id** | 04 | |
| **Priority** | High | |
| **Source** | Manager | |
| **Primary Business Actor** | Manager | |
| **Description** | This function registers the details of a Trainer. Registered trainers become employed in the falconry. | |
| **Expected Scenario** | **Manager** | **System** |
|  | **Step 1:** Invoke Register Trainer function.  **Step 4:** Insert the required data:   * Name * DOB * GenderID   **Step 6:** Confirm. | **Step 2:** Retrieve gender details from Gender File and load on UI.  **Step 3:** Display UI.  **Step 5:** Validate the data:   * All fields must be entered * Name must not be numeric. * DOB must be a valid date. * GenderID must not be numeric.   **Step 7:** Assign next TrainerID  **Step 8:** Set Trainer status to ‘A’ for available.  **Step 9:** Register trainer details in the Trainers file:   * TrainerID * Name * DOB * Status * Gender   **Step 10:** Display confirmation message.  **Step 11:** Reset UI. |
| **Alternate Scenarios** | **Manager** | **System** |
| **Invalid Data Entered** |  | **Step 5:** Invalid data detected.  **Step 6:** Display an appropriate error message.  **Step 7:** Return to Step 4. |
| **Conclusions** | The trainer’s details have been registered. | |
| **Post conditions** | The trainer’s details can be updated.  The trainer can be deregistered. | |

### **Update Trainer**

This function updates the details of a bird in the falconry.

Manager

<<included>>

<<extends>>

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | Update Trainer | |
| **Use Case Id** | 05 | |
| **Priority** | Low | |
| **Source** | Manager | |
| **Primary Business Actor** | Manager | |
| **Description** | This function updates the details of a trainer. This is used if the trainer is promoted/demoted, or the trainer has aged. | |
| **Expected Scenario** | **Manager** | **System** |
|  | **Step 1:** Invoke Update Trainer function.  **Step 5:** Select name of the trainer that is to be updated.  **Step 6:** Change the required data:   * Name * DOB * Status * GenderID   **Step 8:** Confirm. | **Step 2:** Retrieve summary of trainer details from Trainer File.  **Step 3:** Retrieve gender details from Gender File and load on UI.  **Step 4:** Display UI.  **Step 7:** Validate the data:   * All fields must be entered * Name must not be numeric. * DOB must be a valid date. * Status must be ‘A’ for available or ‘U’ for unavailable. * GenderID must not be numeric.   **Step 9:** Update trainer details in the Trainers file:   * Name * DOB * Status * Gender   **Step 10:** Display confirmation message.  **Step 11:** Reset UI. |
| **Alternate Scenarios** | **Manager** | **System** |
| **Invalid Data Entered** |  | **Step 7:** Invalid data detected.  **Step 8:** Display an appropriate error message.  **Step 9:** Return to Step 6. |
| **Conclusions** | The trainer’s details have been updated. | |

### **Deregister Trainer**

This function deregisters a trainer from the database.

Manager

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | Deregister Trainer | |
| **Use Case Id** | 06 | |
| **Priority** | Medium | |
| **Source** | Manager | |
| **Primary Business Actor** | Manager | |
| **Description** | This function deregisters a trainer employment from the falconry. This is used if the trainer has been let go or has vacated the position. | |
| **Expected Scenario** | **Manager** | **System** |
|  | **Step 1:** Invoke Deregister Trainer function.  **Step 4:** Select the name of the trainer that is to be deregistered.  **Step 5:** Confirm. | **Step 2:** Retrieve summary of trainer details from Trainer File.  **Step 3:** Display UI.  **Step 6:** Set trainer status to ‘D’ for deregistered  **Step 7:** Display confirmation message.  **Step 8:** Reset UI. |
| **Conclusions** | The trainer’s employment has been deregistered from the falconry | |

## Process Bookings

### **Set Experience Type**

This function creates an experience type for the falconry.

<<included>>

Manager

<<extends>>

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | Set Experience Type | |
| **Use Case Id** | 07 | |
| **Priority** | High | |
| **Source** | Manager | |
| **Primary Business Actor** | Manager | |
| **Description** | This function set the type of experience a customer can get. Experience types are selected when making a booking | |
| **Expected Scenario** | **Manager** | **System** |
|  | **Step 1:** Invoke Set Experience Type function.  **Step 3:** Insert the required data:   * ExperienceID * Cost * Description   **Step 6:** Confirm. | **Step 2:** Display UI.  **Step 4:** Validate the data:   * All fields must be entered * ExperienceID must not be numeric * Cost must be numeric * Description must not be numeric   **Step 7:** Set Experience status to ‘A’ for available.  **Step 8:** Save Experience Type details in the Experience file:   * ExperienceID * Status * Cost * Description   **Step 9:** Display confirmation message.  **Step 10:** Reset UI. |
| **Alternate Scenarios** | **Manager** | **System** |
| **Invalid Data Entered** |  | **Step 4:** Invalid data detected.  **Step 5:** Display an appropriate error message.  **Step 6:** Return to Step 3. |
| **Conclusions** | Experience Type has been created | |
| **Post conditions** | The experience type details can be updated. | |
| **Business Rules** | ExperienceID must not exist in the database | |

### **Update Experience Type**

This function updates the previously created experience types.

<<included>>

Manager

<<extends>>

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | Update Experience Type | |
| **Use Case Id** | 08 | |
| **Priority** | Medium | |
| **Source** | Manager | |
| **Primary Business Actor** | Manager | |
| **Description** | This function updates the previously created experience types. This is used if the cost has changed, or the experience is closed temporarily. | |
| **Expected Scenario** | **Manager** | **System** |
|  | **Step 1:** Invoke Update Experience Type function.  **Step 4:** Select the Experience Type to be updated.  **Step 6:** Change the required data:   * ExperienceID * Status * Cost * Description   **Step 8:** Confirm. | **Step 2:** Retrieve summary of Experience Type details from Experience file  **Step 3:** Display UI.  **Step 5:** Update UI  **Step 7:** Validate the data:   * All fields must be entered * ExperienceID must not be numeric * Status must be set to ‘A’ for available or ‘U’ for unavailable. * Cost must be numeric * Description must not be numeric   **Step 9:** Update Experience Type details in the Experience file:   * Status * Cost * Description   **Step 10:** Display confirmation message.  **Step 11:** Reset UI. |
| **Alternate Scenarios** | **Manager** | **System** |
| **Invalid Data Entered** |  | **Step 7:** Invalid data detected.  **Step 8:** Display an appropriate error message.  **Step 9:** Return to Step 6. |
| **Conclusions** | Experience Type has been updated | |

### **Create Booking**

This function creates a booking in the database.

Receptionist

Customer

<<included>>

<<extends>>

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | Create Booking | |
| **Use Case Id** | 09 | |
| **Priority** | High | |
| **Source** | Customer | |
| **Primary Business Actor** | Receptionist | |
| **Other Participating Actors** | Customer | |
| **Description** | This function creates a booking for the customer. | |
| **Expected Scenario** | **Customer** | **System** |
|  | **Step 1:** Invoke Create Booking function.  **Step 6:** Insert or select the required data:   * ExperienceType * Date * Time * NoOfPersons * Name * Phone Number * Email   **Step 8:** Confirm. | **Step 2:** Retrieve summary of Experience Type details from Experience file.  **Step 3:** Retrieve summary of Trainer details from the Trainer File  **Step 4:** Retrieve available dates and times from Booking File.  **Step 5:** Display UI.  **Step 7:** Validate the data:   * NoOfPersons must be numeric. * Name must not be numeric. * Phone Number must be numeric. * Email must not be numeric.   **Step 9:** Assign next BookingID  **Step 10:** Assign Trainer to Booking  **Step 11:** Set cost according to ExperienceType and NoOfPersons. (+€10 pp)  **Step 12:** Register Booking details in the Bookings File:   * BookingID * ExperienceType * Date * Time * NoOfPersons * Name * PhoneNumber * Email * Cost * TrainerID   **Step 13:** Display confirmation message.  **Step 14:** Reset UI |
| **Alternate Scenarios** | **Manager** | **System** |
| **Invalid Data Entered** |  | **Step 7:** Invalid data detected.  **Step 8:** Display an appropriate error message.  **Step 9:** Return to Step 6. |
| **Conclusions** | The booking has been created. | |
| **Post conditions** | The booking can be updated.  The booking can be removed. | |

Display UI.

Valid?

Invoke Create Booking function.

Retrieve Experience Types

Retrieve available dates and times.

Insert Required Data

Validate Data.

Set Cost

Update UI

Insert Required Data

Validate Data.

Valid?

Display confirmation message.

Confirm

Register details in database.

Reset UI

Error Message

Error Message

[Y]

[Y]

[N]

[N]

Customer

System

### **Update Booking**

This function updates a booking in the database.

Receptionist

Customer

<<included>>

<<extends>>

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | Update Booking | |
| **Use Case Id** | 10 | |
| **Priority** | Medium | |
| **Source** | Customer | |
| **Primary Business Actor** | Receptionist | |
| **Other Participating Actors** | Customer | |
| **Description** | This function updates the details of a Booking. This is used if the customer wants to change the date or add more participants. | |
| **Expected Scenario** | **Receptionist** | **System** |
|  | **Step 1:** Invoke Update Booking function.  **Step 6:** Enter Email  **Step 9:** Change the required data:   * ExperienceType * Date * Time * NoOfPersons * Name * PhoneNumber * Email   **Step 11:** Confirm | **Step 2:** Retrieve summary of booking details from Booking File.  **Step 3:** Retrieve summary of Experience Type details from Experience file.  **Step 4:** Check available dates and times.  **Step 5:** Display UI.  **Step 7:** Check Booking File for email.  **Step 8:** Display details of selected booking.  **Step 10:** Validate the data:   * NoOfPersons must be numeric. * Name must not be numeric. * Phone Number must be numeric. * Email must not be numeric.   **Step 12:** Set new cost according to ExperienceType and NoOfPersons. (+€10 pp)  **Step 13:** Update Booking details in the Bookings file:   * ExperienceType * Date * Time * NoOfPersons * Name * PhoneNumber * Email * Cost   **Step 14:** Display confirmation message.  **Step 15:** Reset UI. |
| **Alternate Scenarios** | **Receptionist** | **System** |
| **Email Not Found** |  | **Step 7:** Email not found  **Step 8:** Display an appropriate error message.  **Step 9:** Return to Step 6. |
| **Invalid Experience Data Entered** |  | **Step 10:** Invalid data detected.  **Step 11:** Display an appropriate error message.  **Step 12:** Return to Step 9. |
| **Conclusions** | The booking’s details have been updated. | |

### **Remove Booking**

This function removes a booking from the database.

Receptionist

Customer

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | Remove Booking | |
| **Use Case Id** | 11 | |
| **Priority** | Low | |
| **Source** | Customer | |
| **Primary Business Actor** | Receptionist | |
| **Other Participating Actors** | Customer | |
| **Description** | This function removes the booking from the database if the customer cancels. | |
| **Expected Scenario** | **Receptionist** | **System** |
|  | **Step 1:** Invoke Remove Booking function.  **Step 4:** Enter Email  **Step 7:** Confirm. | **Step 2:** Retrieve summary of booking details from Booking File.  **Step 3:** Display UI.  **Step 5:** Check Booking File for email.  **Step 6:** Display details of selected booking.  **Step 8:** Remove booking details from Booking File.  **Step 9:** Display confirmation message.  **Step 10:** Reset UI. |
| **Alternate Scenarios** | **Receptionist** | **System** |
| **Email Not Found** |  | **Step 5:** Email not found  **Step 6:** Display an appropriate error message.  **Step 7:** Return to Step 4. |
| **Conclusions** | The Booking has been removed | |

### **Rate Booking**

This function allows the customer to rate their booking.

Customer

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | Rate Booking | |
| **Use Case Id** | 12 | |
| **Priority** | Low | |
| **Source** | Customer | |
| **Primary Business Actor** | Customer | |
| **Description** | This function enables the customer to rate their booking | |
| **Expected Scenario** | **Customer** | **System** |
|  | **Step 1:** Invoke Rate Booking function.  **Step 4:** Enter email.  **Step 7:** Insert the required data:   * Feedback * Rating * Name   **Step 9:** Confirm | **Step 2:** Retrieve summary of Booking details from Booking File.  **Step 3:** Display UI.  **Step 5:** Check Booking File for email.  **Step 6:** Display details of most recent booking with email.  **Step 8:** Validate data:   * Feedback does not require to be entered * Rating must be numeric between 1 and 5. * Name must not be numeric.   **Step 10:** Assign next RatingID  **Step 11:** Register rating details in Ratings file:   * RatingID * Feedback * Rating * Name * Date * BookingID   **Step 12:** Display confirmation message.  **Step 13:** Reset UI. |
| **Alternate Scenarios** | **Customer** | **System** |
| **Email Not Found** |  | **Step 5:** Email not found  **Step 6:** Display an appropriate error message.  **Step 7:** Return to Step 4. |
| **Rating already made for Booking** |  | **Step 6:** Booking already rated  **Step 7:** Display an appropriate error message  **Step 9:** Return to step 4. |
| **Invalid Data Entered** |  | **Step 8:** Invalid data detected.  **Step 9:** Display an appropriate error message.  **Step 10:** Return to Step 7. |
| **Conclusions** | The customer’s feedback and rating are saved in the database. | |
| **Business Rules** | The function can only analyse data for completed years. | |

## Perform Administrative Reporting

### **Analyse Yearly Revenue**

This function analyses the yearly revenue.

Manager

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | Analyse Yearly Revenue | |
| **Use Case Id** | 13 | |
| **Priority** | High | |
| **Source** | Manager | |
| **Primary Business Actor** | Manager | |
| **Description** | This function analyses the yearly revenue for the Falconry. | |
| **Expected Scenario** | **Manager** | **System** |
|  | **Step 1:** Invoke Analyse Yealy Revenue function.  **Step 5:** Select year for which the revenue analysis is performed.  **Step 9:** Confirm | **Step 2:** Retrieve summary of booking details from Booking File.  **Step 3:** Retrieve summary of Experience Type details from Experience File.  **Step 4:** Display UI.  **Step 6:** Retrieve cost from bookings in the selected year from the Booking File.  **Step 7:** Generate report with the following data:   * Total Revenue * Monthly Revenue * Experience Revenue   **Step 8:** Display report on the UI.  **Step 10:** Reset UI. |
| **Conclusions** | Yearly Revenue is displayed for selected year. | |
| **Business Rules** | The function can only analyse data for completed years. | |

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Jan | Feb | Mar | Apr | May | Jun | Jul | Aug | Sep | Oct | Nov | Dec |
| 891 | 476 | 930 | 938 | 812 | 944 | 534 | 748 | 448 | 823 | 931 | 884 |

|  |  |  |  |
| --- | --- | --- | --- |
| HW - HalkWalk | TR- Training | PB - Public Group | PV - Private Group |
| 1500 | 750 | 1400 | 2500 |

### **Analyse Yearly Experience**

This function analyses the yearly experience ratings.

Manager

Receptionist

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | Analyse Yearly Experience | |
| **Use Case Id** | 14 | |
| **Priority** | Medium | |
| **Source** | Manager | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** | Receptionist | |
| **Description** | This function analyses the yearly experience ratings as rated by the customers. | |
| **Expected Scenario** | **Manager** | **System** |
|  | **Step 1:** Invoke Analyse Yearly Experience function.  **Step 5:** Select year for which the experience analysis is performed.  **Step 10:** Confirm | **Step 2:** Retrieve summary of rating details from Ratings File.  **Step 3:** Retrieve summary of Experience Type details from Experience File.  **Step 4:** Display UI.  **Step 6:** Update UI.  **Step 7:** Retrieve rating data from the Ratings file for the selected year and Experience Type.  **Step 8:** Generates report with the following data:   * Average Rating * Highest Rated Experience * Lowest Rated Experience   **Step 9:** Display report on the UI.  **Step 11:** Reset UI. |
| **Conclusions** | Yearly Experience Rating is displayed for selected year. | |
| **Business Rules** | The function can only analyse data for completed years. | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | HW | TR | PB | PV |
| 1 Star | 6 | 4 | 6 | 2 |
| 2 Star | 15 | 17 | 19 | 13 |
| 3 Star | 29 | 15 | 18 | 20 |
| 4 Star | 28 | 26 | 24 | 19 |
| 5 Star | 42 | 44 | 36 | 48 |
| AVG | 24 | 21.2 | 20.6 | 20.4 |

# System Model

The following dataflow diagrams have been produced for the system:

## Level-0 DFD

Falconry System

Customer

Booking Details

Confirmation

## Level-1 DFD

P1

Manage Birds

P4

Perform Administrative Reporting

D3

Bird File

D2

Trainer File

Bird Details

D1

Experience File

Experience Type Details

D4

Booking File

D5

Ratings File

Rating Details

Payment Details

P2

Manage Trainers

Experience Type Details

Rating Details

Trainer Details

Booking Details

P3

Process Bookings

Customer

Booking Details

Booking Details

Trainer Details

Gender Details

D6

Gender File Perform Administrative Reporting Perform Administrative Reporting Perform Administrative Reporting Perform Administrative Reporting

Gender Details

Rating Details

Species File

D7

Customer

Species Details

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

D6

Gender File Perform Administrative Reporting Perform Administrative Reporting Perform Administrative Reporting Perform Administrative Reporting

D7

Species File Perform Administrative Reporting Perform Administrative Reporting Perform Administrative Reporting Perform Administrative Reporting

D3

Bird File Perform Administrative Reporting Perform Administrative Reporting Perform Administrative Reporting Perform Administrative Reporting

P1.1

Register Bird

P1.2

Update Bird

P1.3

Deregister Bird

Gender Details

Gender Details

Species Details

Species Details

Bird Details

Bird Details

Updated Bird Details

Bird Details

Bird ID

D7

Species File Perform Administrative Reporting Perform Administrative Reporting Perform Administrative Reporting Perform Administrative Reporting

Species Details

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

D6

Gender File Perform Administrative Reporting Perform Administrative Reporting Perform Administrative Reporting Perform Administrative Reporting

D3

Trainer File Perform Administrative Reporting Perform Administrative Reporting Perform Administrative Reporting Perform Administrative Reporting

P2.1

Register Trainer

P2.2

Update Trainer

P2.3

Deregister Trainer

Gender Details

Gender Details

Trainer Details

Trainer Details

Updated Trainer Details

Trainer Details

Trainer ID

## Level-2 DFD (Process P3: Process Bookings)

Booking Details

Updated Booking Details

Trainer Details

D2

Trainer File

Rating Details

Customer

Booking Details

Booking ID

P3.5

Remove Booking

Rating Details

D5

Ratings File

Experience Type Details

Experience Type Details

Booking Details

P3.4

Update Booking

P3.6

Rate Booking

Available Dates

Experience Type Details

Booking Details

Booking Details

D4

Booking File

Customer

P3.3

Create Booking

Experience Type Details

Experience Type Details

Updated Experience Type Details

D1

Experience File

P3.2

Update Experience Type

P3.1

Set Experience Type

## Level-2 DFD (Process P3: Perform Administrative Reporting)

Rating Details

Experience Type Details

D5

Ratings File

P4.2

Analyse Yearly Experience

Experience Type Details

D1

Experience File

Payment Details

D4

Booking File

P4.1

Analyse Yearly Revenue

# Data Model (Class Diagram)

The FalconrySYS Data Model consists of seven tables: Birds ,Trainers, Experiences, Bookings, Ratings, Species, Genders.

The Birds table records the details of birds in the system.

The Trainers table records the details of trainers in the system.

The Experiences table records the details of each experience the customer can book.

The Bookings table records the details of each booking created by a customer.

The Ratings table records the details of the ratings customers can leave on their booking.

The Species table records the details of each species of bird a bird can be assigned to.

The Genders table records the details of each gender a bird or trainer can be assigned to.

## Class Diagram

EXPERIENCE

-ExperienceID: String

-Status: String

-Cost: decimal

-Description: String

-Rating: int

+getAllExperiences()

+getExperience(String)

+addExperience()

+updateExperience()

+findExperience(String)

+getAverageRating(String)

SPECIES

-SpeciesID: int

-Description: String

+getAllSpecies()

+findSpeciesID(String)

+findSpeciesDescription(String)

-TrainerID: int

-Name: String

-DOB: DateTime

-Status: String

TRAINER

+getAllTrainers()

+getTrainer(int)

+addTrainer ()

+updateTrainer()

+findTrainer(String)

+getNextTrainerID()

+findTrainerID(String)

+findTrainerName(String)

GENDER

+getAllGenders()

+findGenderID(String)

+findGenderDescription(String)

-Description: String

-GenderID: int

1

0..\*

has a

runs

1

1

trains

1

is a

0..\*

has a

0..\*

BIRD

+getAllBirds(String)

+getBird(int)

()

+addBird()

+updateBird()

+getNextBirdID()

-BirdID: int

-Name: String

-DOB: DateTime

-Status: String

1

RATING

-RatingID: int

-Feedback: String

-Rating: int

-Date: DateTime

+getAllRatings()

+addRating()

+getNextRatingID()

+getAvgRatingForYear(int, String)

decides

0..\*

1

BOOKING

-BookingID: int

-DateAndTime: DateTime

-NoOfPersons: int

-Name: String

-PhoneNumber: String

-Email: String

-Cost: decimal

+getAllBookings()

+getBooking(String)

+addBooking()

+updateBooking()

+getNextBookingID()

+updateStatus()

+analyseRevenue(String, int)

+analyseExperience(String, int)

1

0..1

1

has a

1..\*

## Relational Schema

Relational schema for the data requirements - Using ***bracket notation***

**Birds**(BirdID, Name, DOB, Status, TrainerID, SpeciesID, GenderID)

**Trainers**(TrainerID, Name, DOB, Status, GenderID)

**Experiences**(ExperienceID, Status, Cost, Description)

**Bookings**(BookingID, Date, Time, NoOfPersons, Name, PhoneNumber, Email, Cost, ExperienceID, TrainerID)

**Ratings**(RatingID, Feedback, Rating, Name, Date, BookingID)

**Species**(SpeciesID, Description)

**Genders**(GenderID, Description)

## Database Schema

A definition of the database to be implemented.

This includes primary key, foreign key, and other constraints to be implemented.

Schema: FalconrySYS

Relation: Birds

Attributes:

BirdID numeric (7) NOT NULL

Name char (20) NOT NULL

DOB date NOT NULL

Status char (1) DEFAULT A

TrainerID numeric (7) NOT NULL

SpeciesID char (2) NOT NULL

GenderID char (1) NOT NULL

Primary Key: BirdID

Foreign Key: SpeciesID References Species

Foreign Key: GenderID References Genders

Relation: Trainers

Attributes:

TrainerID numeric (7) NOT NULL

Name char (20) NOT NULL

DOB date NOT NULL

Status char (1) DEFAULT A

GenderID char (1) NOT NULL

Primary Key: TrainerID

Foreign Key: GenderID references Genders

Relation: Experiences

Attributes:

ExperienceID char (2) NOT NULL

Status char (1) NOT NULL

Cost numeric (3,2) NOT NULL

Description char (20) NOT NULL

Primary Key: ExperienceID

Relation: Bookings

Attributes:

BookingID numeric (7) NOT NULL

Date date NOT NULL

Time time NOT NULL

NoOfPersons numeric (2) NOT NULL

Name char (20) NOT NULL

PhoneNumber numeric (10)

Email char (100)

Cost numeric (3,2) NOT NULL

ExperienceID char (2) NOT NULL

TrainerID numeric (7) NOT NULL

Primary Key: BookingID

Foreign Key: ExperienceID References Experiences

Foreign Key: TrainerID References Trainers

Relation: Ratings

Attributes:

RatingID numeric (7) NOT NULL

Feedback char (250)

Rating numeric (1) NOT NULL

Name char (20)

Date date NOT NULL

BookingID numeric (7) NOT NULL

Primary Key: RatingID

Foreign Key: BookingID References Bookings

Relation: Species

Attributes:

SpeciesID char (2) NOT NULL

Description char (30) NOT NULL

Primary Key: SpeciesID

Relation: Genders

Attributes:

GenderID char (1) NOT NULL

Description char (10) NOT NULL

Primary Key: GenderID

# Conclusion

In conclusion, FalconrySYS is an easy to manage database system. It offers an easy way to handle all the different components of falconry. The database has 4 components, Bookings, Birds, Trainers and Admin, each with different functions. These functions access and append seven tables with important information about birds, trainers, experiences, bookings, ratings, species and genders. FalconrySYS is easy to use and helps users handle all the details of falconry, making it easy to record and find data.

The Birds and Trainers components records and manages the data of the birds and the people involved in falconry and writes the data to their respective tables. The Bookings component focuses on managing the experiences available for booking, the bookings themselves, being able to create, update and remove bookings. The Rate Booking function allows users to leave feedback about their booking, which greatly improves the overall falconry experience.

The Admin component enables analysis of the falconry, analysing yearly revenue and experience ratings. The data these functions use come from the other tables such as the Booking table and the Ratings table. FalconrySYS makes it easier to manage and organise all the details of Kestrel Falconry, allowing users and administrators to access and update data for the falconry.