Software Engineering Project

Salon Management System

Submitted By: Emily Switzer (T00208234)

Computing with Software Development

Date Submitted: 15/5/2020

Table of Contents

[1. Introduction/overview 4](#_Toc40429731)

[2. Functional Components 5](#_Toc40429732)

[3. User Requirements 6](#_Toc40429733)

[3.1. *SalonSYS* will manage Services. 6](#_Toc40429734)

[3.2. *SalonSYS* will manage Employees. 6](#_Toc40429735)

[3.3. *SalonSYS* will manage Appointments. 6](#_Toc40429736)

[3.4. *SalonSYS* will process Administration. 6](#_Toc40429737)

[4. System Requirements 7](#_Toc40429738)

[4.1. System Level Use Case Diagram 7](#_Toc40429739)

[4.2. Manage Services 8](#_Toc40429740)

[4.2.1. Add Service 8](#_Toc40429741)

[4.2.2. Update Service 11](#_Toc40429742)

[4.2.3. Delete Service 13](#_Toc40429743)

[4.3. Manage Employees 15](#_Toc40429744)

[4.3.1. Add Employee 15](#_Toc40429745)

[4.3.2. Delete Employee 17](#_Toc40429746)

[4.4. Manage Appointments 19](#_Toc40429747)

[4.4.1. Make Appointment 19](#_Toc40429748)

[4.4.2. Cancel Appointment 24](#_Toc40429749)

[4.5. Process Administration 26](#_Toc40429750)

[4.5.1. Check availability 26](#_Toc40429751)

[4.5.2. Generate revenue analysis 28](#_Toc40429752)

[4.5.3. Generate service analysis 30](#_Toc40429753)

[5. System Model 32](#_Toc40429754)

[5.1. Level-0 DFD 32](#_Toc40429755)

[5.2. Level-1 DFD 33](#_Toc40429756)

[5.3. Level-2 DFD Manage Services 34](#_Toc40429757)

[5.4. Level-2 DFD Manage Employees 35](#_Toc40429758)

[5.5. Level-2 DFD Manage Appointments 36](#_Toc40429759)

[5.6. Level-2 DFD Process Administration 37](#_Toc40429760)

[6. Data Model (Class Diagram) 38](#_Toc40429761)

[6.1. Class Diagram 38](#_Toc40429762)

[6.2. Relational Schema 39](#_Toc40429763)

[6.3. Database Schema 40](#_Toc40429764)

[7. Conclusion 42](#_Toc40429765)

[8. Appendices 43](#_Toc40429766)

[8.1. Appendix A – Forms 43](#_Toc40429767)

[8.2. Appendix B – Revenue Analysis Report 47](#_Toc40429768)

[8.3. Appendix C – Service Analysis Report 48](#_Toc40429769)

# Introduction/overview

This document specifies the software requirements for a salon appointment booking software system. Requirements have been specified as both User requirements and System requirements.

A system model (using Gane & Sarson notation) has been included to model the interaction between software components and files. The use case model shows the processes which are the features contained in the system. For each process there is a use case narrative which describes the process steps inside each use case.

The proposed system ***‘SalonSYS’*** is a salon appointment booking software system that manages services, employees, appointments and administration while providing an easy to use and attractive interface to view information and generate reports to higher management. The system was developed using Windows Visual and C# programming language.

# Functional Components

This section presets the functional components of the proposed software system.

There are 4 main componets in the proposed system:

# User Requirements

This section describes the user requirements (functional components) as high-level abstract statements.

## *SalonSYS* will manage Services.

* + 1. *SalonSYS* will add a service.
    2. *SalonSYS* will update a service.
    3. *SalonSYS* will delete a service.

## *SalonSYS* will manage Employees.

* + 1. *SalonSYS* will add employees.
    2. *SalonSYS* will delete employees.

## *SalonSYS* will manage Appointments.

* + 1. *SalonSYS* will make appointments.
    2. *SalonSYS* will cancel appointment.

## *SalonSYS* will process Administration.

* + 1. *SalonSYS* will check availability.
    2. *SalonSYS* will generate revenue analysis.
    3. *SalonSYS* will generate service analysis.

# System Requirements

SalonSYS consists of four main modules.

The services’ module provides functions to add a new service, update an existing service and delete an existing service.

The employees’ module provides functions to add a new employee and delete an employee who no longer works in the salon.

The appointments’ module provides functions to create new appointments and to also delete an appointment.

The administration module provides functions to check availability of appointments by day or by employee if the client wants an appointment with a certain employee. It also provides a function to generate revenue analysis and the generate service analysis.

## System Level Use Case Diagram

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

Employee

Manager

Receptionist

Client

## Manage Services

This module provides functions to create a new service, update an active service and delete an active service.

### Add Service

This function adds a new service to *SalonSYS*. Each service is identified by a unique ServiceID which is assigned by the system. Refer to appendix A for the Add Service UI.

Manager

<<includes>>

<<extends>>

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Add Service** | |
| **Use Case Id** | 1 | |
| **Priority** | 1 | |
| **Source** | Owner | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** |  | |
| **Description** | This function adds a new service to the *SalonSYS*. | |
| **Preconditions** |  | |
| **Trigger** |  | |
| **Expected Scenario** | **Manager** | **System Response** |
|  | **Step 1:** The manager invokes the add service function.  **Step 6:** The manager enters the required data:   * ServDescription (100) * DurationID * Cost (5)     **Step 7:**  The manager confirms the new service is to be added. | **Step 2:** The system assigns the next ServiceID.  **Step 3**: The system retrieves details of all **active** services from the Services file.  **Step 4:** The system loads service durations from the *Times file* to the UI.  **Step 5:** The system displays the UI.  **Step 8:** The system validates the data that is entered:   * All fields must be entered. * Description must be alpha. * Description must not already exist in the *Services* file. * Cost must be numeric. * Cost must be more than €1.   **Step 9:** The system assigns a default status of “Available” to the service.  **Step 10:**  The system saves the data in the S*ervices* file:   * ServiceID * ServDescription * DurationID * Cost * ServStatus (‘A’)   **Step 11:** The system displays a confirmation message.  **Step 12:** Reset UI. |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| Invalid Data Entered |  | **Step 7:** Invalid data detected.  **Step 8:** Display appropriate error message.  **Step 9:** Return to Step 6. |
| All fields not entered |  | **Step 7:** All fields not entered are detected.  **Step 8:** Display appropriate error message.  **Step 9:** Return to Step 6. |
| **Conclusions** | The new service is added to the *Services* file. | |
| **Post conditions** | The service may now be booked. | |
| **Business Rules** | The service must not already exist in the database. | |
| **Implementation Constraints** |  | |

### Update Service

This function updates an **active** service on the *SalonSYS*.

Manager

<<includes>>

<<extends>>

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Update Service** | |
| **Use Case Id** | 2 | |
| **Priority** | 1 | |
| **Source** | Owner | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** |  | |
| **Description** | This function updates an **active** service on the *SalonSYS*. | |
| **Preconditions** | There must be an active service on the SalonSYS. | |
| **Trigger** |  | |
| **Expected Scenario** | **Manager** | **System Response** |
|  | **Step 1:** The manager invokes the update service function  **Step 3:** The manager chooses the service they wish to update  **Step 6:** The manager updates the required data:   * ServDescription * DurationID * Cost   **Step 7:**  The manager confirms changes to service. | **Step 2:** The system loads summary details of **active** services from the *Services* file.  **Step 4:** The system retrieves all details for the selected service from the *Services* file and displays on the UI for editing.  **Step 5:**  The system loads all durations from the *Times* file.  **Step 8:** The system validates the data that is updated:   * At least one field must be entered. * Description must be alpha. * Cost must be numeric. * Cost must be more than €1   **Step 9:**  The system updates the changed data in the *Services* file:   * ServDescription * DurationID * Cost     **Step 10:** The system displays a confirmation message.  **Step 11:** Reset UI. |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| Invalid Data Entered |  | **Step 8:** Invalid data detected.  **Step 9:** Display appropriate error message.  **Step 10:** Return to step 6. |
| At least one field not entered |  | **Step 8:** All least one field not changed detected.  **Step 9:** Display appropriate error message.  **Step 10:** Return to step 6. |
| **Conclusions** | The service is updated in the *Services* file | |
| **Post conditions** |  | |
| **Business Rules** | Only **active** services can be updated. | |
| **Implementation Constraints** |  | |

### Delete Service

This function deletes an **active** service on SalonSYS. The service will no longer be available in the SalonSYS.

Manager

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Delete Service** | |
| **Use Case Id** | 3 | |
| **Priority** | 1 | |
| **Source** | Owner | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** |  | |
| **Description** | This function deletes an **active** service on the SalonSYS. | |
| **Preconditions** | There must be an active service on the SalonSYS. | |
| **Trigger** |  | |
| **Expected Scenario** | **Manager** | **System Response** |
|  | **Step 1:** The manager invokes the delete service function  **Step 4:** The manager chooses the service they wish to delete  **Step 6:** The owner confirms the service is to be deleted. | **Step 2:** The system loads summary details of **active** services from the *Services* file  **Step 3:** The system displays the UI.  **Step 5:**  The system retrieves all the details of selected the service from the *Services* file and displays on the UI for viewing only  **Step 7:** The system checks the *Appointments* file to see that there are no future appointments for the service chosen. If no appointments for this exist in the Appointments file the process continues.  **Step 8:** The system updates the service status to “Deleted” in the Service file (ServStatus = “D”);  **Step 9:** The system displays a confirmation message.  **Step 10:** The system resets the UI. |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| Future Appointments Exist |  | **Step 7:** Future appointments found.  **Step 8:** The system displays an appropriate error message.  **Step 9:**  Return to step 4 |
| **Conclusions** | The service is flagged as “Deleted” from the SalonSYS. | |
| **Post conditions** | This service cannot be booked. | |
| **Business Rules** | Only **active** services can be deleted. | |
| **Implementation Constraints** |  | |

## Manage Employees

This module provides functions to add a new employee and delete an active employee.

### Add Employee

This function adds a new employee to be added to SalonSYS. Each employee is identified by a unique EmployeeID which is assigned by the system. Refer to Appendix A for the Add Employee UI.

Manager

<<includes>>

<<extends>>

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Add Employee** | |
| **Use Case Id** | 4 | |
| **Priority** | 1 | |
| **Source** | Owner | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** |  | |
| **Description** | This function adds a new employee to the SalonSYS. | |
| **Preconditions** |  | |
| **Trigger** |  | |
| **Expected Scenario** | **Manager** | **System Response** |
|  | **Step 1:** The manager invokes the add employee function  **Step 5:** The manager enters the required data:   * EmpName (15) * EmpSurname (20) * Email (50) * ServiceID (5)   **Step 6:**  The manager confirms the new employee is to be added. | **Step 2:** The system assigns the next EmployeeID  **Step 3**: The system loads all the **active** services from the *Services* file.  **Step 4:** The system displays the UI  **Step 7:** The system validates the data that is entered:   * All fields must be entered * Forename and Surname must be alpha. * Email must be valid   **Step 8:** The system assigns a default status of “Available” to the employee.  **Step 9:**  The system saves the data in the Employees file:   * EmployeeID * EmpName * EmpSurname * Email * ServiceID * EmpStatus (‘A’)   **Step 10:** The system displays a confirmation message  **Step 11:** Reset UI |
| **Alternate Scenarios** | **Manager** | **System Response** |
| **Invalid Data Entered** |  | **Step 7:** Invalid data detected.  **Step 8:** Display appropriate error message.  **Step 9:** Return to step 5. |
| **All Fields Not Entered** |  | **Step 7:** Fields not entered detected.  **Step 8:** Display appropriate error message.  **Step 9:** Return to step 5. |
| **Conclusions** | The new employee is added to the *Employees* file. | |
| **Post conditions** | The new employee is now available for bookings. | |
| **Business Rules** | The employee must not already exist on the database. | |
| **Implementation Constraints** |  | |

### Delete Employee

This function deletes an active employee from the SalonSYS

Manager

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Delete Employee** | |
| **Use Case Id** | 5 | |
| **Priority** | 1 | |
| **Source** | Owner | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** |  | |
| **Description** | This function deletes an active employee on the SalonSYS. | |
| **Preconditions** | There must be an active employee on the SalonSYS. | |
| **Trigger** |  | |
| **Expected Scenario** | **Manager** | **System Response** |
|  | **Step 1:** The owner invokes the delete employee function  **Step 4:** The manager chooses the employee they wish to delete  **Step 6:** The manager confirms employee is to be deleted. | **Step 2:** The system loads summary details of **active** employees from the *Employee* file  **Step 3:** The system displays the UI  **Step 5:** System retrieves all details of selected staff member from the *Employee* file and displays on UI for viewing only.  **Step 7:**  The system checks the Appointment file to see that there are no future appointments with the employee.  **Step 8:** The system updates the employee status to “Deleted” in the Employee file (EmpStatus = “D”);  **Step 9:** The system displays a confirmation message.  **Step 10:** The system resets the UI. |
| **Alternate Scenarios** | **Manager** | **System Response** |
| **Future Appointments Exist** |  | **Step 7:** Future appointments found  **Step 8:** The system displays an appropriate error message  **Step 9:**  Return to step 4 |
| **Conclusions** | The employee is flagged as “Deleted” in the *Employees* file | |
| **Post conditions** | This employee cannot be booked | |
| **Business Rules** | Only **active** employees can be deleted. | |
| **Implementation Constraints** |  | |

## Manage Appointments

This module provides functions to make appointments and to cancel appointments.

### Make Appointment

This function makes appointments on the SalonSYS. Each employee is identified by a unique AppointmentID which is assigned by the system. Refer to appendix A for the Make Appointment UI.

Receptionist

Make Appointment Activity Diagram

A screenshot of a cell phone

Description automatically generated

Figure 1: Activity Diagram for Make Appointment

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Make Appointment** | |
| **Use Case Id** | 6 | |
| **Priority** | 1 | |
| **Source** | Owner | |
| **Primary Business Actor** | Receptionist | |
| **Other Participating Actors** |  | |
| **Description** | This function makes a new appointment on the SalonSYS. | |
| **Preconditions** |  | |
| **Trigger** |  | |
| **Expected Scenario** | **Receptionist** | **System Response** |
|  | **Step 1:** The receptionist invokes the make appointment function  **Step 5:** The receptionist enters the service for the appointment.  **Step 10:** The receptionist enters the remaining required data:   * AppDate * EmployeeID * TimeID * Client Name * Phone Number   **Step 11:** The system calculates an endtime for the appointment.  **Step 12:**  The receptionist confirms the new appointment is to be added to the system. | **Step 2:** The system assigns the next AppointmentID  **Step 3**: The system loads the list of **active** services from the *Services* file on the UI.  **Step 4:** The system displays the UI.  **Step 6**: The system loads the list of available **active** employees offering the chosen service from the *EmployeeServices* file to the UI.  **Step 7**: The system displays the cost of the chosen service from the *Services* file to the UI.  **Step 8:** The system loads the list of appointment times from the *Times* file to the UI.  **Step 9**: The system loads a calendar with dates from the current day onward to the UI.  **Step 13:** The system validates the data that is entered:   * All fields must be entered * Client name must be alpha * Client phone must be numeric * Appointment must be available in the Appointments file.   **Step 14:**  If the appointment is available the system saves the data in the Appointments file:   * AppointmentID * ServiceID * EmployeeID * TimeID * EndTime * AppDate * Client * Phone   **Step 16:** The system displays a confirmation message  **Step 17:** Reset UI |
| **Alternate Scenarios** | **Receptionist** | **System Response** |
| **Appointment not available** |  | **Step 13:** Appointment not available.  **Step 14:** Display appropriate error message.  **Step 15:** Return to step 10. |
| **Invalid Data Entered** |  | **Step 13:** Invalid data detected.  **Step 14:** Display appropriate error message.  **Step 15:** Return to step 10. |
| **All Fields Not Entered** |  | **Step 13:** All fields not entered detected.  **Step 14:** Display appropriate error message.  **Step 15:** Return to step 10. |
| **Conclusions** | The new appointment is added to the *Appointments* file | |
| **Post conditions** |  | |
| **Business Rules** |  | |

### Cancel Appointment

This function cancels an **active** appointment on the SalonSYS. Refer to appendix A for the cancel appointment UI.

Receptionist

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Cancel Appointment** | |
| **Use Case Id** | 7 | |
| **Priority** | 1 | |
| **Source** | Owner | |
| **Primary Business Actor** | Receptionist | |
| **Other Participating Actors** |  | |
| **Description** | This function cancels a booking on the SalonSYS. | |
| **Preconditions** | The cancel appointment function can only be invoked on a date in the future. | |
| **Trigger** |  | |
| **Expected Scenario** | **Receptionist** | **System Response** |
|  | **Step 1:** The receptionist invokes the cancel appointment function  **Step 4:** The receptionist chooses the appointment they wish to cancel.  **Step 6:** The receptionist confirms appointment is to be cancelled. | **Step 2:** The system loads AppointmentID’s of future appointments from the *Appointments* file.  **Step 3:** The system displays the UI  **Step 5:** The system retrieves full details of the appointment from the Appointment file, the Employees file and the Services file and displays on the UI for viewing only.  **Step 7:** The system removes the appointment from the *Appointments* file.  **Step 8:** The system displays confirmation message.  **Step 9:** The system resets the UI. |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
|  |  |  |
| **Conclusions** | The appointment is deleted from the Appointments file | |
| **Post conditions** | The time slot and employee are available for booking. | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

## Process Administration

This module provides functions to check availability, generate a revenue analysis for the year and to generate a service analysis for the year.

### Check availability

This function generates a list of times that each employee is unavailable for booking. Refer to appendix A for the check availability UI.

Receptionist

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Check Availability** | |
| **Use Case Id** | 8 | |
| **Priority** | 1 | |
| **Source** | Owner | |
| **Primary Business Actor** | Receptionist | |
| **Other Participating Actors** |  | |
| **Description** | This function generates a list of appointments for the employee. | |
| **Preconditions** |  | |
| **Trigger** |  | |
| **Expected Scenario** | **Receptionist** | **System Response** |
|  | **Step 1:** The receptionist invokes the check availability function  **Step 5:** The receptionist enters the required data:   * EmployeeID * Date | **Step 2:** The systemloads the **active** employees from the *Employees* file  **Step 3**: The system loads a calendar with dates from the current day onward to the UI.  **Step 4:** The system displays the UI.  **Step 6:** The system checks the appointments file for chosen employee on the date chosen.  **Step 7:** If appointments are found, the system retrieve all appointments found and will display a summary of the data found to the UI for viewing only. |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **No appointments found** |  | **Step 6:** No appointments found.  **Step 7:** Display confirmation message that the employee is available. |
| **Conclusions** | The system displays unavailable appointments to the user. | |
| **Post conditions** |  | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

### 4.5.2. Generate revenue analysis

This function generates a revenue analysis of a chosen year. Refer to appendix B for sample analyses.

Owner

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Generate Revenue Analysis** | |
| **Use Case Id** | 9 | |
| **Priority** | 1 | |
| **Source** | Owner | |
| **Primary Business Actor** | Owner | |
| **Other Participating Actors** |  | |
| **Description** | This function generates an analysis of revenue from sales per annum. | |
| **Preconditions** |  | |
| **Trigger** |  | |
| **Expected Scenario** | **Owner** | **System Response** |
|  | **Step 1:** The owner invokes the generate revenue analysis function  **Step 4:** The owner chooses the year they wish to generate the revenue analysis for. | **Step 2:** The system loads a list of previous years the Salon has been in business.  **Step 3:** The system displays the UI.  **Step 5:** The system loads all the revenue details from the *Appointments* file for the chosen year.  **Step 6:**  The system calculates to total revenue that was earned for the chosen year. It then divides the revenue earned for the chosen year to each month that is was earned.  **Step 7:** The system displays the revenue analysis on a chart to the UI  **Step 8:** Reset UI |
| **Alternate Scenarios** | **Owner** | **System Response** |
|  |  |  |
| **Conclusions** | The system displays a revenue analysis for the year. | |
| **Post conditions** |  | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

### Generate service analysis

This function generates a service analysis of a chosen year. Refer to appendix B for sample analyses.

Owner

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Generate Service Analysis** | |
| **Use Case Id** | 10 | |
| **Priority** | 1 | |
| **Source** | Owner | |
| **Primary Business Actor** | Owner | |
| **Other Participating Actors** | None | |
| **Description** | This function generates an analysis of revenue from services per annum. | |
| **Preconditions** | None | |
| **Trigger** | None | |
| **Expected Scenario** | **Owner** | **System Response** |
|  | **Step 1:** The owner invokes the generate service analysis function  **Step 4:** The owner chooses the year they wish to generate the service analysis for. | **Step 2:** The system loads a list of previous years the Salon has been in business.  **Step 3:** The system displays the UI.  **Step 5:** The system loads all the total service details from the *Appointments* file for the chosen year.  **Step 6:**  The system calculates to total revenue that was earned from each service for the chosen year from each appointment that was for the service chosen.  **Step 7:** The system displays the service analysis on a pie chart to the UI  **Step 8:** Reset UI |
| **Alternate Scenarios** | **Owner** | **System Response** |
|  |  |  |
| **Conclusions** | The system displays the service analysis for the year | |
| **Post conditions** |  | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

# System Model

The following dataflow diagrams have been produced for the system:

## Level-0 DFD

Appointment Request

Customer

SalonSYS

Booking Reference

## Level-1 DFD

Employee Details

Service Details

P1

Manage Services

D2

Employees File

P2

Manage Employees

D4

EmployeeServices File

Employee Details

Service Details

Employee Details

EmployeeService Details

D1

Services File

Service Details

Service Details

Employee

Appointment Details

P3

Manage Appointments

P4

Perform Administration

Customer

Appointment Reference

Appointment Details

Appointment Details

D3

Appointments File

## Level-2 DFD Manage Services

P1.1

Add Service

P1.2

Change Service

Service Details

Service Details

Service Details

D1

Services File

Service Details

Service ID

P1.3

Delete Service

Service Details

D3

Appointments File

## Level-2 DFD Manage Employees

D3

Appointments File

Employee Details

P2.1

Add New Employee

P2.2

Delete Employee

Employee Details

Employee Details

Employee Details

Employee ID

c

D2

Employees File

D1

Servicess File

## Level-2 DFD Manage Appointments

Appointment Details

Appointment Details

Customer

P3.1

Make Appointment

Appointment ID

P3.2

Cancel Appointment

Appointment Details

Appointment Details

Appointment Reference

D3

Appointments File

Employee Details

Service Details

Appointment Details

D2

Employees File

D1

Services File

Service Details

## Level-2 DFD Process Administration

P4.1

Check Availability

P4.2

Generate Revenue Analysis

Appointment Details

Appointment Details

D3

Appointments File File

Service Details

P4.3

Generate Service Analysis

# Data Model (Class Diagram)

## Class Diagram

Employee

EmployeeID\*

EmpName

EmpSurname

Email

EmpStatus

Service

ServiceID\*

ServDescription

Cost

ServStatus

TimeSlot

DurationID\*

Duration

has a

🡪

1

0..\*

1

1

1

0..\*

has

🡪

Is for

🡪

EmployeeService

🡪

Is for

1..\*

0..\*

🡪

Is with

0..\*

🡪

Appointment

AppID\*

AppDate

EndTime

AppCost

Client

Phone

0..\*

AppTime

AppTimeID\*

AppTime

0..\*

has a

🡪

1

All\_Month

Month\*

## Relational Schema

Relational schema for the data requirements

Services (ServiceID, ServDescription, DurationID, Cost, ServStatus)

Employees (EmployeeID, EmpName, EmpSurname, Email, EmpStatus)

EmployeeServices (EmployeeID, ServiceID)

Appointments (AppID, AppDate, AppTimeID, EndTime, ServiceID, EmployeeID, Cost, Client, Phone)

AppTimes (AppTimeID, AppTime)

TimeSlots (DurationID, Duration)

All\_Months(Month)

## Database Schema

**Schema:** **SalonSYS**

**Relation: Employees**

Attributes:

EmployeeID numeric (8) NOT NULL

EmpName varchar2 (15) NOT NULL

EmpSurname varchar2 (20) NOT NULL

Email varchar2 (30) NOT NULL UNIQUE

EmpStatus char (1) NOT NULL

**Primary Key:** EmployeeID

**Relation: TimeSlots**

Attributes:

DurationID numeric (5) NOT NULL

Duration varchar2 (5) NOT NULL UNIQUE

**Primary Key:** DurationID

**Relation: AppTimes**

Attributes:

AppTimeID numeric (2) NOT NULL

AppTime varchar2 (5) NOT NULL UNIQUE

**Primary Key:** AppTimeID

**Relation: Services**

Attributes:

ServiceID numeric (8) NOT NULL

ServDescription varchar2 (100) NOT NULL UNIQUE

DurationID numeric (5) NOT NULL

Cost numeric (5) NOT NULL

ServStatus char (1) NOT NULL

**Primary Key:** ServiceID

**Foreign Key:** DurationID references TimeSlots

**Relation: EmployeeServices**

Attributes:

ServiceID numeric (8) NOT NULL

EmployeeID numeric (8) NOT NULL

**Primary Key:** ServiceID, EmployeeID

**Foreign Key:** ServiceID references Services

**Foreign Key:** EmployeeID references Employees

**Relation: Appointments**

Attributes:

AppID numeric (8) NOT NULL UNIQUE

AppDate Date NOT NULL

AppTimeID numeric (5) NOT NULL

EndTime varchar2 (5) NOT NULL

ServiceID numeric (5) NOT NULL

EmployeeID numeric (8) NOT NULL

AppCost numeric (5) NOT NULL

Client varchar2 (15) NOT NULL

Phone numeric (10) NOT NULL

**Primary Key:** AppID

**Foreign Key:** ServiceID references Services

**Foreign Key:** EmployeeID references Employees

**Foreign Key:** AppTimeID references AppTimes

**Relation: All\_Months**

Attributes:

Month numeric (8) NOT NULL UNIQUE

**Primary Key:** Month

# Conclusion

SalonSYS is a system that helps to manage salon appointments much more efficiently and effectively. The system allows user to insert different types of information such as insert appointments, employees and services. The system allows the user to modify the data of services should a service need to alter its’ cost for promotional purposes. It will also display appointments in both calendar and planner form and generate reports on the performance of services. The SalonSYS will generate a yearly revenue analysis to provide the salon's performance to any stakeholders. The SalonSYS determines the availability of an employee for a specified date for effective appointment management. In conclusion, all the agreed functions for the system have been specified in this document.

# Appendices

## Appendix A – Forms

A screenshot of a cell phone

Description automatically generated

Figure 2: Add Employee

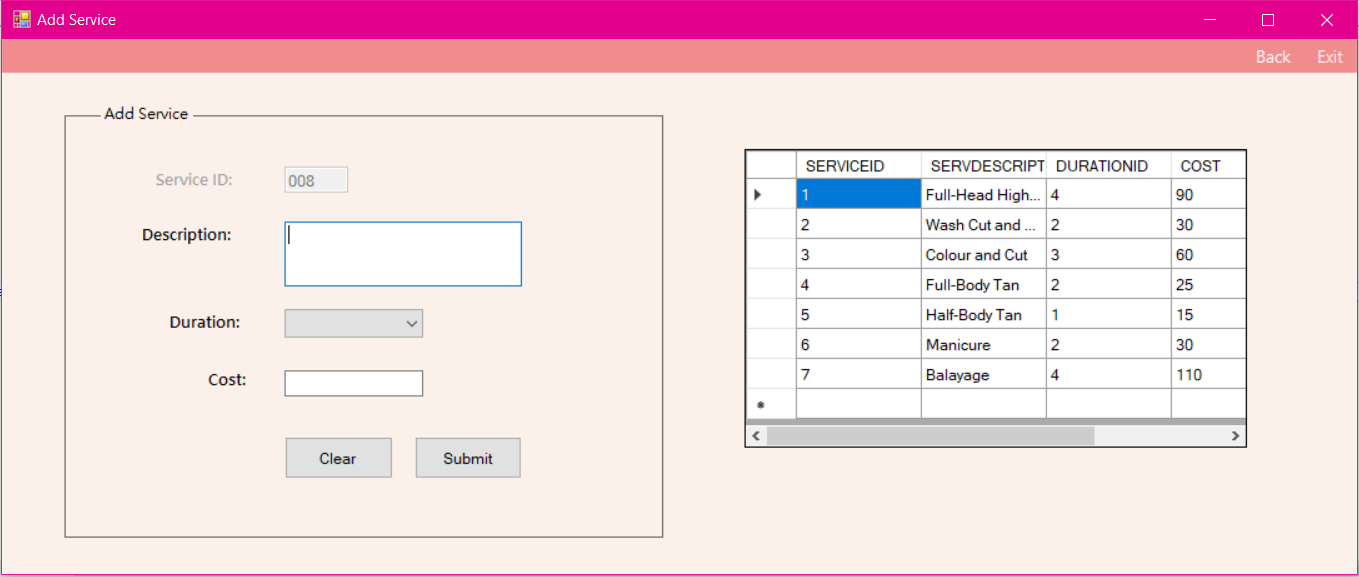


Figure 3: Add Service

A screenshot of a cell phone

Description automatically generated

Figure 3: Check Availability

A screenshot of a cell phone

Description automatically generated

Figure 4: Cancel Appointment

A screenshot of a cell phone

Description automatically generated

Figure 5: Make Appointment

## Appendix B – Revenue Analysis Report

A screenshot of a cell phone

Description automatically generated

Figure 6: Revenue Analysis Report for 2017

## Appendix C – Service Analysis Report

A screenshot of a cell phone

Description automatically generated

Figure 7: Service Analysis Report for 2019