Swimming Pool Booking System-Case Study

Submitted By: Katie Griffiths

T00175748

Computing with Multimedia – Stage 2

Date Submitted: 21/12/2015

**Table of Contents**

[1. Introduction/overview 4](#_Toc437597453)

[2. Functional Components 5](#_Toc437597454)

[3. Functional User Requirements 6](#_Toc437597455)

[3.1 Pool System will manage members. 6](#_Toc437597456)

[3.2 Pool System will manage activities. 6](#_Toc437597457)

[3.3 Pool System will perform registrations. 6](#_Toc437597458)

[4. Functional System Requirements 7](#_Toc437597459)

[4.2 Manage Members 7](#_Toc437597460)

[**4.2.1** **New Member** 7](#_Toc437597461)

[**4.2.2** **Amend Member** 9](#_Toc437597462)

[**4.2.3** **Block Member** 12](#_Toc437597463)

[**4.2.4** **List All Members** 14](#_Toc437597464)

[4.3 Manage Activities 16](#_Toc437597465)

[**4.3.1** **Register New Activity** 16](#_Toc437597466)

[**4.3.2** **View all Activities** 18](#_Toc437597467)

[**4.3.3** **Amend Activity** 19](#_Toc437597468)

[4.4 Perform Registrations 21](#_Toc437597469)

[**4.4.1** **Member Activity Registration** 21](#_Toc437597470)

[**4.4.2** **Show Activity Rota** 23](#_Toc437597471)

[**4.4.3** **Analyse Activity Revenue** 25](#_Toc437597472)

[5. System Model 27](#_Toc437597475)

[5.1. Level-0 DFD 28](#_Toc437597476)

[5.2. Level-1 DFD 29](#_Toc437597477)

[Level-2 DFD (Process P1: Manage Members) 29](#_Toc437597478)

[Level-2 DFD (Process P2: Manage Activity) 31](#_Toc437597479)

[Level-2 DFD (Process P3: Perform Registrations) 32](#_Toc437597480)

[32](#_Toc437597481)

[6. Data Model (Class Diagram) 33](#_Toc437597482)

[7. Database Design 34](#_Toc437597483)

[8. Conclusion 35](#_Toc437597484)

[9. Appendices 36](#_Toc437597485)

[Appendix A – Member Listing 36](#_Toc437597486)

# Introduction/overview

The manager of a local swimming pool has asked you to develop a membership and activity platform to replace their current paper based system. The platform needs to handle membership periods and rates, as well as individual class/activity bookings. The manager has specified that classes can change on a regular basis, and both existing members in good standing, as well as guests might wish to attend. Currently, membership is available in quarterly, six monthly and yearly intervals, but the management company has expressed interest in adding shorter time periods in the future.

# Functional Components

**Hierarchy Chart**

# Functional User Requirements

## Pool System will manage members.

* + 1. Pool System will register a new member.
    2. Pool System will amend a member.
    3. Pool System will block a member.
    4. Pool System will list all members

## Pool System will manage activities.

* + 1. Pool System will register a new activities.
    2. Pool System will view the activities.
    3. Pool System will amend the activity.

## Pool System will perform registrations.

* + 1. Pool System will provide a list activity registrations.
    2. Pool System will create an activity rota.
    3. Pool System will analyse the revenue for the activities.

# Functional System Requirements

The Pool System will develop functions to allow the system to deal with memberships and activities and their registrations. The system will allow for members to be managed in a way of blocking a member from their membership and any activity and for a full listing of each member with membership details included. When the system is dealing with the managing of activities there will be functional components in place to perform a list and produce a rota of all activities which can be analysed for revenue.

## Manage Members

This function will allow new members to register and to produce a list of all members, the ability to block a member and to amend any member that is in the system.

### **New Member**

This function will allow the system to register a new member and to hold the members details in the pool system.

Member

Manager

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **New member** | |
| **Use Case Id** | 1 | |
| **Priority** | 1 | |
| **Source** | Manager | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** | Member | |
| **Description** | This use case will allow new members to register in the system. | |
| **Preconditions** |  | |
| **Trigger** |  | |
| **Typical Scenario** | **Actor Action** | **System Response** |
|  | **Step 1:** The manager will call on the new member function.  **Step 3:** The manager will enter the member’s details:   * Surname * Forename * House Number * Street * Town * County * Phone * Email   **Step 4:** The manager confirms for a new member to be registered. | **Step 2:** The system determines the next member ID anddisplays the UI.  **Step 5:** The system validates the data entered:   * All fields must be entered * Name must be alphabetic characters only. * Phone must only contain numeric values. * Email must be a valid email address. * Email should not already be in use.   **Step 6:** The system saves the members details in the members file.  **Step 7:** The system displays a confirmation message and resets the UI. |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Invalid Data Entered** |  | **Step 9:** The system finds invalid data  **Step 10:** The system displays an appropriate error message |
| **Conclusions** | A new member can now be registered and saved in the Members File. | |
| **Post conditions** | New members can be register in the system. | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

### **Amend Member**

This functional requirement will allow the system to change any member’s details that are currently registered on the Pool System.

Member

Manager

**Staff System**

Display UI

Retrieve summary of members with surname

Update member file

Display confirmation message

Validate member details

Confirm to proceed

Change member details

Retrieve full details of selected member

Select Member to Amend

Enter Member Surname

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Amend a member** | |
| **Use Case Id** | 2 | |
| **Priority** | 1 | |
| **Source** | Manager | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** | Member | |
| **Description** | This use case will allow member details to be changed. | |
| **Preconditions** |  | |
| **Trigger** |  | |
| **Typical Scenario** | **Actor Action** | **System Response** |
|  | **Step 1:** The manager will call on the amend member function.  **Step 3:** The manager will enter the member’s surname.  **Step 5:** The manager selects the member to be amended.  **Step 7:** The manger will change the member’s details.  **Step 8:** The mangers confirms the member’s details to be amended. | **Step 2:** The system will display the UI.  **Step 4:** The system retrieves a summary of all members with surname entered from the members file.  **Step 6:** The system retrieves the full details of the selected member from the members file.  **Step 9:** The system checks to see if member’s details are valid.  **Step 10:** The system validates the member’s details.  **Step 11:** The system updates the members file.  **Step 12:** The system displays a confirmation message and resets the UI. |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Invalid Data Entered** |  | **Step 13:** The system finds invalid data  **Step 14:** The system displays an appropriate error message |
| **Conclusions** | All changes to members will be assigned in the Members File. | |
| **Post conditions** | All members can be updated with new details. | |
| **Business Rules** | Only unblocked Members can be added | |
| **Implementation Constraints** |  | |

### **Block Member**

This functional component will allow a member to be blocked from the system due to misbehaviour.

Manager

**Staff**  **System**

**Manager System**

Display’s UI with members details required.

System retrieves details with member’s surname

User will input member’s surname

User selects member to change status

Sets status to ‘Blocked’ in members file

User confirms status to update

Display Confirmation Message

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Block a member** | |
| **Use Case Id** | 3 | |
| **Priority** | 1 | |
| **Source** | Manager | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** |  | |
| **Description** | This use case will allow a members status to be blocked on the system. | |
| **Preconditions** |  | |
| **Trigger** |  | |
| **Typical Scenario** | **Actor Action** | **System Response** |
|  | **Step 1:** The manager will call on the block member function from the members file.  **Step 3:** The manager will enter member’s surname.  **Step 5:** The manager selects the member to be blocked.  **Step 7:** The manager confirms status to be changed. | **Step 2:** The system will display the UI with member’s details.  **Step 4:** The system retrieves a summary of the details regarding the member’s surname from the members file and displays on UI.  **Step6:** the system retrieves the details for the selected member.  **Step 8:** The system displays a confirmation message and sets the status of the member to ‘blocked’ in the members file.  **Step 9:** The system updates the members file.  **Step 10:** The system clears the UI. |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
|  |  |  |
| **Conclusions** | Any change to a member’s status has been created in the members File. | |
| **Post conditions** | Any member with misconduct can be assigned with a block member status | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

### **List All Members**

This component will display a list of all members who are currently registered with their details of membership type, personal details and current status.

Manager

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Members Details** | |
| **Use Case Id** | 4 | |
| **Priority** | 1 | |
| **Source** | Manager | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** |  | |
| **Description** | This use case will allow the system to display a list of members in specified orders. | |
| **Preconditions** |  | |
| **Trigger** |  | |
| **Typical Scenario** | **Actor Action** | **System Response** |
|  | **Step 1:** The manager will call on the list all members function.  **Step 3:** The manager will enter a member’s name.  **Step 5:** The manager selects the member being displayed in grid view.  **Step 7:** The manager can sort the information by either member id or surname. | **Step 2:** The system will load the members file containing member’s details and display the UI.  **Step 4:** The system retrieves the details being asked from the members file and displays in a grid view.  **Step 6:** The system retrieves the details being asked from the members file.  **Step 8:** The system updates and save the members details to the members file.  **Step 9:** The system displays a conformation message.  **Step 10:** The system clears the UI. |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
|  |  |  |
| **Conclusions** | All of the systems member’s details are stored in the Members file. | |
| **Post conditions** | All members currently on file will be listed with their details. | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

## Manage Activities

This functional component will allow for current activities to be listed and changed. The system will also give the opportunity to add in new activities that the company would like to list.

### **Register New Activity**

This component will allow the system to record a new activity that will be taking place in the company.

Manager

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Register New Activity** | |
| **Use Case Id** | 2 | |
| **Priority** | 1 | |
| **Source** | Manager | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** |  | |
| **Description** | This use case will allow the system to register a new activity and accommodate a number of spaces available. | |
| **Preconditions** |  | |
| **Trigger** |  | |
| **Typical Scenario** | **Actor Action** | **System Response** |
|  | **Step 1:** The manager will call on the function to register a new activity.  **Step 3:** The manager enters the activity type details:   * Activity Name * Description * Duration in Weeks * No. of spaces * Cost   **Step 4:** The manager confirms the new activity is to be set. | **Step 2:** The system determines next activity Id and displays UI.  **Step 5:** The system validates the data entered:   * All fields must be entered * Activity must be alphabetic characters only. * Duration to be displayed in date format in weeks * Cost must be numeric and greater than zero * Activity type must not already be registered. * Start date for future activity   **Step 6:** The system checks to see if the available spaces have been assigned to the activity.  **Step 7:** The system saves the new activity in the activity file.  **Step 8:** The system displays a conformation message.  **Step 9:** The system clears the UI. |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
|  |  |  |
| **Conclusions** | Any new activity will be stored in the Activities File. | |
| **Post conditions** | All activities that are new can be listed. | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

### **List all Activities**

This system requirement will allow the user to view all activities that are being instructed in the company.

Manager

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **List all Activities** | |
| **Use Case Id** | 2 | |
| **Priority** | 1 | |
| **Source** | Manager | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** |  | |
| **Description** | This use case will allow the system to display a list all of all the activities specified by the manager. | |
| **Preconditions** |  | |
| **Trigger** |  | |
| **Typical Scenario** | **Actor Action** | **System Response** |
|  | **Step 1:** The manager will call on the function to list all activities.  **Step 3:** The manager may specify which activity to list.  **Step 5:** The manager views the activity that is being listed. | **Step 2:** The system will retrieve the details of all current and future activities from the activity file and display the UI.  **Step 4:** The system will retrieve the details of the activity that the manager has selected to view.  **Step 6:** The system displays a conformation message.  **Step 7:** The system clears the UI. |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
|  |  |  |
| **Conclusions** | All activities will be listed and stored in the Activities File. | |
| **Post conditions** | All activities that are new can be listed. | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

### **Amend Activity**

This will allow the manager to make any changes needed to the activity files on the system.

Manager

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Amend Activity** | |
| **Use Case Id** | 2 | |
| **Priority** | 2 | |
| **Source** | Manager | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** |  | |
| **Description** | This use case will allow the system to amend an activity. | |
| **Preconditions** |  | |
| **Trigger** |  | |
| **Typical Scenario** | **Actor Action** | **System Response** |
|  | **Step 1:** The manager will call on the function to amend an activity.  **Step 3:** The manager will specify the name of the activity.  **Step 5:** The manager selects the activity to be amended.  **Step 6:** The manger changes the activity details that are listed.  **Step 7:** The manager confirms change. | **Step 2:** The system retrieves the details of the activity from the activity file and display’s the UI.  **Step 4:** The system retrieves the information regarding the name of the activity.  **Step 8:** The system validates the list of activities details.  **Step 9:** The system updates the activity file.  **Step 10:** The system displays a confirmation message and clears the UI. |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
|  |  |  |
| **Conclusions** | Any changes made to an activity will be stored in the Activities File. | |
| **Post conditions** | Any activity can be amended. | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

## Perform Registrations

This system requirement will allow for a list of all members currently registered with an activity, display the activity rota showing all activities that take place and a revenue of the intake from activities that take place.

### **Activity Registration**

For this case the system will allow the manager to call on a function to register a member to an activity.

Member

Manager

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Activity Registration** | |
| **Use Case Id** | 3 | |
| **Priority** | 1 | |
| **Source** | Manager | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** | Member | |
| **Description** | This use case will allow the system to register a member for a specific activity. | |
| **Preconditions** |  | |
| **Trigger** |  | |
| **Typical Scenario** | **Actor Action** | **System Response** |
|  | **Step 1:** The manager will call on the function to register an activity.  **Step 3:** The manager will enter in an activity name.  **Step 5:** The manager selects the activity from valid members and fills in the members details:   * Member id * Members name * Members phone No. | **Step 2:** The system will display the UI.  **Step 4:** The system retrieves the details of the activity from the Activity file.  **Step 6:** The system validates the details entered.  **Step 7:** The system updates and save the registration in the registration file.  **Step 8:** The system displays a conformation message.  **Step 9:** The system clears the UI. |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
|  |  |  |
| **Conclusions** | All member registrations for activities will be stored in registrations File. | |
| **Post conditions** | Registrations for activities will be listed. | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

### **Show Activity Rota**

This will allow the manager to view a detailed rota displaying the current activities that will be taking place each semester.

Manager

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Activity Rota** | |
| **Use Case Id** | 3 | |
| **Priority** | 2 | |
| **Source** | Manager | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** |  | |
| **Description** | This use case will allow the system to display a rota for a specific activity currently taking place. | |
| **Preconditions** |  | |
| **Trigger** |  | |
| **Typical Scenario** | **Actor Action** | **System Response** |
|  | **Step 1:** The manager will call on the function to display the rota for activities.  **Step 3:** The manager enters in an activity name.  **Step 5:** The manager confirms the activity to be viewed. | **Step 2:** The system retrieves details of all current and future activities from the activity file and displays on the UI.  **Step 4:** The system retrieves all activities with the name supplied from the Activity file.  **Step 6:** The system displays the details of the activity selected to view.  **Step 7:** The system displays a conformation message.  **Step 8:** The system clears the UI. |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
|  |  |  |
| **Conclusions** | All information regarding the activity rota will be stored in registrations File. | |
| **Post conditions** | The rota for activities will be listed. | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

### **Analyse Activity Revenue**

The system will display the total revenue for each activity that had been completed. The manager can then analyse the intake taken for the activities and review the gross income.

Manager

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Activity Revenue** | |
| **Use Case Id** | 3 | |
| **Priority** | 3 | |
| **Source** | Manager | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** |  | |
| **Description** | This use case will allow the system to analyse the revenue for the current activities. | |
| **Preconditions** |  | |
| **Trigger** |  | |
| **Typical Scenario** | **Actor Action** | **System Response** |
|  | **Step 1:** The manager will call on the function to analyse the revenue for activities.  **Step 3:** The manager selects for activities to be listed.  **Step 5:** The manager selects the activities to be analysed.  **Step 6:** The manager confirms the data. | **Step 2:** The system will display the UI.  **Step 4:** The system retrieves the activities details from the registrations file.  **Step 7:** The system displays a conformation message.  **Step 8:** The system clears the UI. |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
|  |  |  |
| **Conclusions** | All income from activities will be analysed and displayed. | |
| **Post conditions** | Analysis of revenue from activities will be displayed. | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |



# System Model

The following dataflow diagrams have been produced for the system:

**DFD Elements:**

**External Entities:**

Member

**Data Stores:**

D1 Members File

D2 Activity File

D3 Registration File

**Processes:**

**P1 Manage Members**

P1.1 New Member

P1.2 Amend Member

P1.3 Block Members

P1.4 List Members

**P2 Manage Activities**

P2.1 Register New Activity

P2.2 View all Activities

P2.3 Amend Activity

**P3 Perform Registrations**

P3.1 Member Activity Registration

P3.2 Show Activity Rota

P3.3 Analyse Activity Revenue

## Level-0 DFD

Member

Registration

Pool

SYS

Member ID

## Level-1 DFD

P2

Manage Activities

D2

Activity File

Activity Details

Activity Details

Member Details

P1

Manage Members

Registration details

D3

Registration File

D1

Members File

P3

Perform Registrations

Member Details

Member Details

Member

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

Registration details

Member Details

Member Details

Member

Member Details

P1.2

Amend a Member

Member Details

Change Member Details

P1.3

Block a Member

P1.1

New Member

Member Details

P1.4

List all Members

Member Status

D1

Members File

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

Activity Details

Activity Details

Register Activity

Activity Details

Activity Details

D2

Activity File

D2

Registration File

P2.3

Amend Activity

P2.2

View Activity

P2.1

Register New Activity

## Level-2 DFD (Process P3: Perform Registrations)

Members Details

D1

Members File

D3

Registration File

Activity Registration

Registration Details

Activity Details

Activity Details

Activity Details

D2

Activity File

Member

P3.1

Member Activity Registration

P3.3

Analyse Activity Revenue

P3.2

Show Activity Rota

# Data Model (Class Diagram)

|  |
| --- |
| **Member** |
| Member\_id (pk) |
| Surname |
| Forename |
| Address |
| Phone |
| Email |
| Status  1 |

|  |
| --- |
| **Activity** |
| Activity\_No (pk) |
| Name |
| Start\_Date |
| Duration |
| Rate |
| No\_Spaces |
| No\_Avaliable |

Is for

Is for

0….\*

|  |
| --- |
| **Registrations** |
| Registration\_No (pk) |
| Registration\_Date |
| Member\_id |
| Activity\_id |
| Amount |

0….20

1

# Database Design

**7.1. Relational Schema**

**Member (**Member\_ID, Surname, Forename, Address, Phone, Email, Status**)**

**Registration (**Registration\_No, Registration\_Date, Member\_ID, Activity\_No, Account**)**

**Activities (**Activity\_No, Name, Start\_Date, Finish\_Date, Rate, No\_Spaces, No\_Avaliable**)**

**7.2. Database Schema**

**Relation:** Members

Attributes:

Member\_ID numeric(4)

Surname char (25)

Forename char(35)

HouseNo numeric(3)

Street char(25)

Town char(20)

County char(25)

Phone char(14)

Email char (35)

Status char(1)

Primary Key: member\_ID

**Relation** Registration

Attributes

Registration\_No numeric (4)

Registration\_Date date

Account decimal (5,2)

Member\_Id numeric (4)

Activity\_No numeric(4)

PrimaryKey: Registration\_No

Foreign Key: Member\_Id References Members

Foreign Key: Activity\_No References Activities

**Relation** Activities

Attributes:

Activity\_No numeric (4)

Activity\_Name char(30)

Start\_Date date

Duration numeric(20)

Rate Decimal(6,2)

No\_Of\_Spaces numeric(25)

Availability numeric(25)

Primary Key: Activity\_No.

# Conclusion

During the time of this project, I have learned a lot of information in analysing and building a functional system. With going through changes being made throughout the process of this project I was able to see how each change made a difference in the making of the system. As I found out the documentation and specification for each functional component was very important for the whole design of the system.

I soon became to value what each process of this project was in ways that I wanted to start coding the prototype before the use cases had been completed. With knowing that this would not be a one person job in the work force, I began to acknowledge the hard work that goes in to make a system work and the group effort and communication that is needed for it to be successful.

Overall with finishing the project and prototype I feel very pleased with what I have produced so far and hopefully getting the chance next year to learn more coding so to make the prototype go even further is very much exciting to hear.

# Appendices

## Appendix A – Member Listing

The member listing should be as follows:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **28/02/2012** | **Current Member Listing** | | | | |  |  |
|  |  | |  | |  |  |  |
| **Member Id** | **Surname** | | **Forename** | | **Phone** | **Email** | **Status** |
| 0123 | Smith | | Joe | | 066 754879 | Joe.smith@yahoo.com |  |
| 0236 | Cahill | | Mary | | 085 987621 | [M.C@gmail.com](mailto:M.C@gmail.com) | Blocked |
| 0069 | Doe | | Jane | | 061 456891 | [Jane.Doe@yahoo.co.uk](mailto:Jane.Doe@yahoo.co.uk) |  |
| 0083 | McCarthy | | Ian | | 087 699943 | IanM@123.com | Blocked |
| 0176 | Fisher | | Elaine | | 086 336541 | ElaineFish@gmail.com |  |
|  |  | |  | |  |  |  |
| **Summary Detail** | | | | | | |  |
| Total Members | 5 |  | |  | |  |  |
| Members Blocked | 2 |  | |  | |  |  |