Requirements Engineering

GolfMemSYS

Submitted By: Michael Crowley McDevitt (T00200043)

Computing with Software Development

Date Submitted: 25/August/2020

**Table of Contents**

[1. Introduction/overview 3](#_Toc49257635)

[2. Functional Components 4](#_Toc49257636)

[3. User Requirements 5](#_Toc49257637)

[3.1. GolfMemSYS will manage Member Administration Tasks such as: 5](#_Toc49257638)

[3.2. GolfMemSYS will manage Renewals Administration Tasks such as: 5](#_Toc49257639)

[3.3. GolfMemSYS will create Business Reports such as: 5](#_Toc49257640)

[4. System Requirements 6](#_Toc49257641)

[4.1. System Level Use Case Diagram 7](#_Toc49257642)

[4.2. Manage Members 8](#_Toc49257643)

[4.2.1. Add Membership Type 8](#_Toc49257644)

[4.2.2. Add Member 10](#_Toc49257645)

[4.2.3. View Members 14](#_Toc49257646)

[4.2.4. Update Member 16](#_Toc49257647)

[4.3. Manage Renewals 21](#_Toc49257648)

[4.3.1. Issue Notices 21](#_Toc49257649)

[4.3.2. Record Payments 22](#_Toc49257650)

[4.4. Analyse Finance 25](#_Toc49257651)

[4.4.1. Create Yearly Revenue Report 25](#_Toc49257652)

[4.4.2. Carry out Membership Type Analysis 25](#_Toc49257653)

[5. System Model 27](#_Toc49257654)

[5.1. Level-0 DFD 27](#_Toc49257655)

[5.2. Level-1 DFD 28](#_Toc49257656)

[5.3. Level-2 DFD (Process P1: Manage Members) 29](#_Toc49257657)

[5.4. Level-2 DFD (Process P2: Send Renewal Notice) 30](#_Toc49257658)

[6. Data Model (Class Diagram) This section outlines the data stored by the system and gives graphical representations of this data and how it is stored. 31](#_Toc49257659)

[6.1. Class Diagram 31](#_Toc49257660)

[6.2. Relational Schema 32](#_Toc49257661)

[6.3. Database Schema 33](#_Toc49257662)

[7. Conclusion 35](#_Toc49257663)

# Introduction/overview

This document contains details to create a golf membership system that provides functions for member management, member type management, finance management and renewal tracking. The system will have three core processes; manage members, finance management and renewal management. These processes will consist of their own unique functions which are highlighted below.

Manage Members will have four functions: Add Member, Update Member, View Member and Remove Member. Add Member will allow the user to add a new member to the system. Update Members will allow the user to update an existing member. Remove Member will set the status of an existing member to ‘removed’ so that they no longer count towards the renewal process.

Manage Finance will have two functions: Revenue Analysis and Record Payment. Yearly Revenue Analysis will show the income of the golf course on a specified range of time. Record Payment will allow the system to store the payment paid by each member.

Manage Renewals will have two functions: Issue Renewal Notice and Membership Type Analysis. Issue Renewal Notice allows the system to send out a notice to each member one month prior to the renewal date to give them time and awareness for the renewal to happen. Membership Type Analysis allows the system to show the current number of members and the amount of money the selected Member Type brings in per year based on current member counts signed up to that type. This is useful for finance purposes if demand can change the price of membership according to the number of members of each type.

# Functional Components

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

There are 3 main components in the proposed system.

# User Requirements

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

## GolfMemSYS will manage Member Administration Tasks such as:

* + 1. GolfMemSYS will add a Membership Type
    2. GolfMemSYS will add a member
    3. GolfMemSYS will update a member
    4. GolfMemSYS will remove a member
    5. GolfMemSYS will view all members

## GolfMemSYS will manage Renewals Administration Tasks such as:

* + 1. GolfMemSYS will issue renewal notices on and after the due date.
    2. GolfMemSYS will record payments

## GolfMemSYS will create Business Reports such as:

* + 1. GolfMemSYS will display a yearly revenue report
    2. GolfMemSYS will perform a membership type analysis

# System Requirements

GolfMemSYS consists of three main modules.

The Members module provides functions to add a Member, update Member’s details, view members in the system and add a new membership type for new members to sign up to.

The Renewals module provides functions to issue renewal notices and to record payments.

The Finances module provides functions to display a revenue report and to show a membership type analysis.

## System Level Use Case Diagram

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

Golf Member System

Manager

Member

## Manage Members

This module processes all things related to golf club members such as adding new Members or editing information for existing members.

### Add Membership Type

Manager

<<Includes>>

Validate Information

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Add Membership Type** | |
| **Use Case Id** | 1 | |
| **Priority** | 1 | |
| **Source** | Manager | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** |  | |
| **Description** | This function adds a Membership type to the GolfMemSYS system. | |
| **Preconditions** | None | |
| **Trigger** | None | |
| **Expected Scenario** | **Manager** | **System** |
|  | **Step 1:** The Manager invokes the Add Membership Type function  **Step 3:** Manager enters required data:   * TypeID (NVARCHAR2 2) * TypeName (NVARCHAR2 50) * TypePrice (NUMBER 5,2)   **Step 4:** Manager confirms that the Membership Type is to be added. | **Step 2:** Display UI  **Step 5:** The system validates the data entered:   * All fields must be entered * TypeID cannot already exist * TypeName must not be numeric * The TypePrice must be numeric and GREATER THAN zero   **Step 8:** The System saves the data in the Members files:   * TypeID * TypeName * TypePrice   **Step 8:** Show confirmation Message.  **Step 9:** Reset UI |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
|  |  | **Step 5:** Invalid Information is found  **Step 6:** Show suitable error message  **Step 7:** Return to Step 3: |
| **Conclusions** | A new Member Type will be added to the system. | |
| **Post conditions** | None. | |
| **Business Rules** | Any member can now be assigned this new type. | |
| **Implementation Constraints** | None. | |

### Add Member

This function adds a new member to the system. Each member is identified by a unique MemberID. When a member is added, the member status is set to “Active”.

Manager

Member

<<Includes>>

Validate Information

SYSTEM RESPONSE

MANAGER INPUT

ACTIVITY DIAGRAM: ADD MEMBER.

Start Add Member Function

Assign Member ID

Display UI

Enter Member Details

Validate Member Details

Error Message

No

Valid?

Yes

Set Member Status to “A” (Active)

Set SIGNUPDATE to current Date

Save Member Details to File

Set SIGNUPDATE to current Date

Show Confirmation Message

Clear UI

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Add Member** | |
| **Use Case Id** | 2 | |
| **Priority** | 1 | |
| **Source** | Manager | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** | Member | |
| **Description** | This function adds a Member to the GolfMemSYS system. | |
| **Preconditions** | None | |
| **Trigger** | None | |
| **Expected Scenario** | **Manager** | **System** |
|  | **Step 1:** The Manager invokes the Add Member function  **Step 6:** Manager enters required data:   * MemberID * FirstName * LastName * Street * Town * County * Mobile * Email * MemberStatus * SignUpDate * TypeID * Paid   **Step 7:** Manager confirms that the Member is to be added. | **Step 2:** System shows the next MemberID in the UI.  **Step 3:** The system retrieves details of Membership Types from the Member Types file and loads on UI  **Step 4:** Retrieve counties from Counties file and load on UI  **Step 5:** Display UI  **Step 8:** The system validates the data entered:   * All fields must be entered * Member Name must not contain digits. * Email must be valid format   **Step 9:** System assigns SignUpDate to current system date.  **Step 10:** system assigns a default Member Status “Active” and default “Paid” status to n (no).    **Step 11:** The System saves the data in the Members files:   * MemberID * FirstName * LastName * Street * Town * County * Mobile * Email * MemberStatus * SignUpDate * TypeID * Paid   **Step 12:** Show confirmation Message.  **Step 13:** Reset UI |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Invalid Data Entered** |  | Step 8: Invalid Data entered  Step 9: Display an appropriate error message  Step 10: Return to step 6 |
| **Conclusions** | The member is added to the system | |
| **Post conditions** | This member can now pay/renew fees | |
| **Business Rules** | A default payment of 0 is created for the newly added member. To allow for the record payment to know they have not paid. | |
| **Implementation Constraints** | None. | |

### View Members

**Description:**

This function lets the user view all members in the system, and all their information.

**Use Case Diagram:**

Manager

**Use Case Narrative:**

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **View Members** | |
| **Use Case Id** | 3 | |
| **Priority** | 1 | |
| **Source** | Manager | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** |  | |
| **Description** | This function shows all members in the GolfMemSYS system. | |
| **Preconditions** | The system needs to load all current members | |
| **Trigger** |  | |
| **Expected Scenario** | **Manager** | **System** |
|  | **Step 1:** The Manager triggers the View Members function  **Step 4:** Manager can view all members in the system.  **Step 5:** Manager exits the UI | **Step 2:** Retrieves the summary details of current members from the members file and the Member Types file  **Step 3:** Display UI  **Step 6:** The system closes the UI. |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
|  |  |  |
| **Conclusions** | The manager can view members in the system. | |
| **Post conditions** |  | |
| **Business Rules** | Members with removed status are also displayed. | |
| **Implementation Constraints** |  | |

### Update Member

**Description**:

This function will allow the manager to update information for an existing user.  
You can select a member from the system and then edit any information that is related to them.

It is validated, then the new information is updated in the system.

**Use Case Diagram**

Member

Manager

**Use Case Narrative:**

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Update Member** | |
| **Use Case Id** | 4 | |
| **Priority** | 1 | |
| **Source** | Manager | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** | Member | |
| **Description** | This function updates an existing Member’s details in the GolfMemSYS system. | |
| **Preconditions** | The system needs to load all current members. | |
| **Trigger** |  | |
| **Expected Scenario** | **Manager** | **System** |
|  | **Step 1:** The Manager triggers the Update Member function  **Step 4:** Manager can view all members in the system and can select one to update.  **Step 6:** The manager updates the details.  **Step 7:** The manager confirms the update. | **Step 2:** Retrieves the list of current members from the system.  **Step 3:** Display UI  **Step 5:** The system shows the current member information on the UI form.    **Step 8:** The system validates the information.  **Step 9:** The system saves the information to the Member file.   * MemberID * MemberType * FirstName * LastName * Street * Town * County * Mobile * Email * MemberStatus * SignUpDate * TypeID * Paid   **Step 10:** The system shows a confirmation message  **Step 11:** The system resets the UI |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Invalid Data Entered** |  | **Step 8:** The System fails to validate the data  **Step 9:** Display appropriate error message  **Step 10:** Return to step 6 |
| **Conclusions** | The manager can select a member from the list and update their details. | |
| **Post conditions** | None | |
| **Business Rules** | None | |
| **Implementation Constraints** | None | |

SYSTEM RESPONSE

MANAGER INPUT

ACTIVITY DIAGRAM: UPDATE MEMBER.

Start Update Member Function

Retrieves a list of current members

Display UI

Select a Member to be updated

Fill selected Member details into UI

Enter new Member details

Validate new Member Details

Error Message

No

Valid?

Yes

Save Updated Member Details to File

Show Confirmation Message

Clear UI

## Manage Renewals

### Issue Notices

### Record Payments

Manager

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Record Payment** | |
| **Use Case Id** | 6 | |
| **Priority** | 1 | |
| **Source** | Manager | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** | None. | |
| **Description** | This function allows the manager to record member payments | |
| **Preconditions** | The system needs to load all existing payment records | |
| **Trigger** | None. | |
| **Expected Scenario** | **Manager** | **System** |
|  | **Step 1:** The Manager triggers the Record Payment function  **Step 4:** Manager can view all payments and add a new one for existing members  **Step 5:** Manager selects a MemberID from the list to record the payment for.  **Step 6:** Manager selects appropriate payment amount from a list.  **Step 7:** Manager select the suitable payment for the member type the selected member is part of. | **Step 2:** Retrieves the summary details of past payments from the Renewals file.  **Step 3:** Display UI  **Step 8:** The system assigns the current date to the new payment.  **Step 9:** The system saves the payment details to the renewals file |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
|  |  |  |
| **Conclusions** | The manager can make a payment for any members that have not paid yet or need a renewal. | |
| **Post conditions** | None. | |
| **Business Rules** | None. | |
| **Implementation Constraints** | None. | |

## Analyse Finance

### Create Yearly Revenue Report

### Carry out Membership Type Analysis

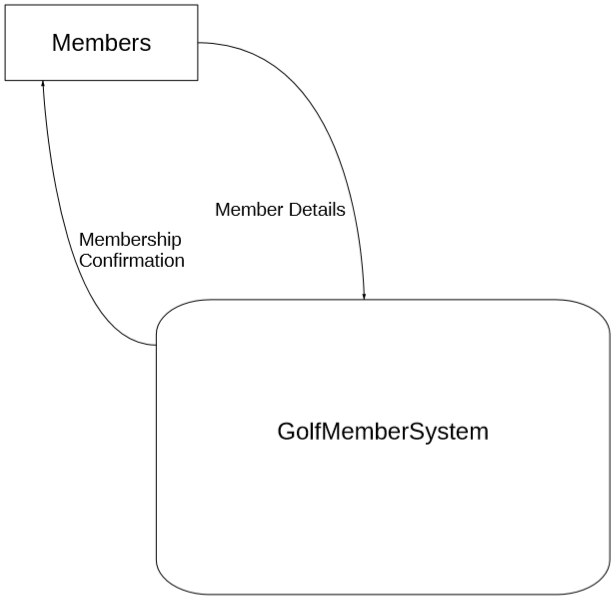
Manager

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Membership Type Analysis** | |
| **Use Case Id** | 8 | |
| **Priority** | 2 | |
| **Source** | Manager | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** | None. | |
| **Description** | This function shows a breakdown of the selected member type and how many members and financial gains it generated. | |
| **Preconditions** | The system needs to load all member types and group the members accordingly. | |
| **Trigger** | None. | |
| **Expected Scenario** | **Manager** | **System** |
|  | **Step 1:** The Manager triggers the Membership Type Analysis function.  **Step 4:** Manager can select a Member Type to view the breakdown of  **Step 5:** Manager exits the UI | **Step 2:** Retrieves the summary details of current member types from the Member Types file  **Step 3:** Display UI  **Step 6:** The system closes the UI. |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
|  |  |  |
| **Conclusions** | The manager can view a breakdown of the selected member type and how many members and financial gains it generated. | |
| **Post conditions** | None. | |
| **Business Rules** | Members with removed status are not displayed. | |
| **Implementation Constraints** | None. | |

# System Model

The following dataflow diagrams have been produced for the system:

## Level-0 DFD



## Level-1 DFD

A picture containing text, map

Description automatically generated

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

A picture containing screenshot

Description automatically generated

## Level-2 DFD (Process P2: Send Renewal Notice)

A screenshot of a cell phone

Description automatically generated

# Data Model (Class Diagram) This section outlines the data stored by the system and gives graphical representations of this data and how it is stored.

## Class Diagram

GOLFMEMBERSLOG

* MEMBERID PK
* FIRSTNAME
* LASTNAME
* STREET
* TOWN
* COUNTY
* MOBILE
* EMAIL
* MEMBERSTATUS
* SIGNUPDATE
* PAID
* TYPEID

COUNTY

* COUNTY PK

1

Must have

1..\*

Makes A

Renewals

* RenewalID PK
* MemberID
* AmountPaid
* DatePaid

Must have

1

MEMBERTYPES

* TYPEID PK
* TYPENAME
* TYPEPRICE

1

MEMBERSTATUS

* MEMBERSTATUS PK

## Relational Schema

MEMBERTYPES (TYPEID, TYPENAME, TYPEPRICE)

COUNTYLIST (COUNTY)

GOLFMEMBERSLOG (MEMBERID, FIRSTNAME, LASTNAME, STREET, TOWN, COUNTY, MOBILE, EMAIL, MEMBERSTATUS, SIGNUPDATE, TYPEID, PAID)

RENEWALS (RENEWALID, MEMBERID, AMOUNTPAID, DATEPAID)  
  
MEMBERSTATUS(MEMBERSTATUS)

## Database Schema

A definition of the database to be implemented.   
This includes primary key, foreign key and other constraints to be implemented.

**Schema:** GOLFMEMSYS schema

**Relation:** MEMBERTYPES  
**Attributes:**  
TYPEID NVARCHAR2(2) NOT NULL  
TYPENAME NVARCHAR2(50)  
TYPEPRICE NUMBER(5,2)  
**PRIMARY KEY:** TYPEID

**Relation:** COUNTYLIST  
**Attributes:**  
COUNTY NVARCHAR2(50) NOT NULL  
**Primary Key:** COUNTY  
  
  
**Relation:** GOLFMEMBERSLOG  
**Attributes:**  
MEMBERID NVARCHAR(3) NOT NULL  
FIRSTNAME NVARCHAR2(50)  
LASTNAME NVARCHAR2(150)  
STREET NVARCHAR2(30)  
TOWN NVARCHAR2(30)  
COUNTY NVARCHAR2(30)  
MOBILE NVARCHAR2(10)  
EMAIL NVARCHAR2(150)  
MEMBERSTATUS NVARCHAR2(10)  
SIGNUPDATE DATE  
TYPEID NVARCHAR2(2) NOT NULL  
PAID NVARCHAR2(1) NOT NULL  
  
**Primary Key:** MEMBERID  
**Foreign Key:** TYPEID REFERENCES MEMBERTYPES (TYPEID)

**Relation:** RENEWALS  
**Attributes:**  
RENEWALID NUMBER(2) NOT NULL  
MEMBERID NVARCHAR2(3) NOT NULL  
AMOUNTPAID NUMBER(5,2) NOT NULL  
DATEPAID DATE  
**Primary Key:** RENEWALID  
**Foreign Key:** MEMBERID REFERENCES GOLFMEMBERSLOG (MEMBERID)  
  
  
**Relation:** MEMBERSTATUS  
**Attributes:**  
MEMBERSTATUS NVARCHAR2(10) NOT NULL  
**Primary Key:** MEMBERSTATUS

# Conclusion

This document has outlined the design, requirements and functional components for a golf membership auditing software solution. The systems architecture at a high level consists of a C# front end with Object Orientated code layout and an oracle database backend. The functional components of the system allow a manager to create, view, update and delete golf club members from the system and additionally perform ad-hoc queries on business objects (financial recorded data) and current existing members data.

This system components are as follows:  
 1) Member Functions  
 - Add Member Type  
 - Add Member  
 - Update Member  
 - Remove Member  
 - View Members  
  
 2) Renewal Functions  
 - Issue a notice when members need renewals  
 - Record payments for members who are new or needing a renewal  
  
 3) Finance Functions  
 - Member Type Analysis  
 - Yearly Revenue Analysis

This system is robust and provides a user-friendly interface that I intuitive and well uniformed throughout the UI. This system could be adopted by numerous industries such as hotel booking and other sports club memberships.  
It is also able to automatically assign the sign-up date based on the current time and date of the system.

Further developments of this system could be a scoring mechanism for golf members to keep track of their score and handicap and update via running the system and entering the appropriate data. This could also be developed as a portable application.