Solution Specification: Motor Racing Club System

Bespoke Software Consultants (BSC)

Outline

This is the specification for the development of a motor racing club system. This serves as the specification for the assignments (both 01 & 02) for ICT3611. Read the specification carefully, and use the assignment marking rubrics (both rubric 01 and 02) when developing the solution.

Your solution must address all the functionality and requirements as discussed in this specification. It must also address the requirements as provided in the rubric.

Please pay careful attention: assignment 01 should store all data in **files on disk**. Assignment 02 should store data in a **relational database**.

1. Introduction

This document provides a base-line specification for the development of a motor racing club management system for client Gauteng West Motoring Association. A base-line specification is used to put on paper the features, functions, and operational guidelines for the system to be developed. As such, it is imperative that all stakeholders scrutinise the document to ensure that all requirements are understood properly.

2. Scope

The scope of the document covers the functional requirements of the software. System configuration, and hardware requirements are excluded.

3. Glossary

Business object	An entity represented by the system, such as a racing driver, or a motor racing event.
Check digit	A special digit which can be used to see if an error has been made in the event that a membership number has been typed in manually.
Data file	A file used to store data on a business object, or a race result.
Event	A motor racing event such as a local race, or a national race.
Manage	The word management refers to the capability to add, modify, and delete data.
Race result	The position after completing the race.

4. Functionality

The software shall provide features which allow a small motor racing club the ability to:

1. Manage motor racing events that its members may participate in.
2. Manage racing drivers (members) of the club.
3. Recording performance of club members in club's events.

The performance for racing drivers should only include the finishing position in the races.

4.1 Managing motor racing events

Managing a motor racing event includes registering the event in the system (and recording details about the event), modifying an existing even, or deleting an event.

It should not be possible to delete an event that has records of racing drivers' performance. That is, once a user enters performance details (results) for a racing driver in an event, it must not be possible to delete that event from the system.

Events should have unique names – it should not be possible to add two events with the same name. This allows the user of the system to uniquely identify each event in the system.

The following data about a motor racing event should be collected:

1. Title of the event (must be unique).
2. Date of the event.
3. Registration Fee.
4. Location of the event.
5. Number of laps.

4.2 Managing racing drivers

Since racing drivers are the primary members of the club, the system must provide proper management of racing drivers.
The following details on racing drivers must be recorded:

1. Membership number (see 4.2.1),
2. Name and surname,
3. Birth date,
4. Gender,
5. Date on which the club was joined,
6. Membership fee amount outstanding.
7. Races competed in, and positions for each race.

It must be possible to add, edit, and delete racing drivers. However, once race results for a racing driver have been recorded, it should no longer be possible to delete them.

4.2.1 Club membership number

A club membership number is a 14 digit number constructed from information from the member. For example the membership number 15198901050047 is constructed from the numbers:

15 Year of registration	19890105 Birthday	004 Random Number between 000 - 999	7 Check digit

The following table provides more information on each sub-field:

Year of registration	The last two digits of the year in which this member was added to the system
Birthday	The birthday of the member in the format yyyymmdd, where yyyy represents the full year of birth, mm represents the numerical month of birth, and dd represents the numerical day of the month.
Random number	A number between 0 and 999 chosen randomly.
Check digit	See below.

Definition 1 *Calculation of the check digit:*

1. *Sum the first 13 individual digits in the member number,*
2. *divide the result by 10 and take the remainder,*
3. *if the remainder is 0, then the check digit is 0, otherwise subtract the remainder from 10 and use the result as the check digit.*

The proper algorithm for generating the check digit is presented in algorithm 1.

Example 1 *The complete number for '1519890105004' will be calculated as follows:*

1. $1 + 5 + 1 + 9 + 8 + 9 + 0 + 1 + 0 + 5 + 0 + 0 + 4 = 43$
2. *43 divided by 10 is 4, with a remainder of 3.*
3. *Subtract this number from 10 ($10 - 3 = 7$). The check digit is thus* 7*.*
4. *The full membership number is '15198901050047'*

Example 2 *The complete number for '1519890105001' will be calculated as follows:*

1. $1 + 5 + 1 + 9 + 8 + 9 + 0 + 1 + 0 + 5 + 0 + 0 + 1 = 40$
2. *40 divided by 10 is 4, with a remainder of 0.*
3. *The remainder is* 0*, thus the check digit is* 0
4. *The full membership number is thus '15198901050010'*

Whenever a vehicle owner number is entered by a user (when doing a search, for example), the system must validate the number.

Definition 2 *Validating the check digit:*

1. *Sum all the digits in the number (including the check digit)*
2. *Divide the result by 10. If the remainder is 0, then the number is valid, if not, then it is invalid.*

Example 3 *Validate number '15198901050047':*

1. *Sum the individual digits:* $1 + 5 + 1 + 9 + 8 + 9 + 0 + 1 + 0 + 5 + 0 + 0 + 4 + 7 = 50$
2. 50 *modulo*[1] $10 = 0$
3. *Result: this is a valid number.*

Example 4 *Validate number '15198901050010':*

1. *Sum the individual digits:* $1 + 5 + 1 + 9 + 8 + 9 + 0 + 1 + 0 + 5 + 0 + 0 + 1 = 40$
2. 40 *modulo* $10 = 0$

[1] a modulo $b =$ the remainder after dividing a by b

3. *Result: this is a valid number.*

Suppose the user is asked to do a look-up on a member using their membership number and they make a typing mistake. Instead of entering '15198901050010' they enter 15198001050010.

Example 5 *Validate number '15198001050010':*

1. *Sum the individual digits:* $1+5+1+9+8+0+0+1+0+5+0+0+1=31$
2. 31 *modulo* $10=1$
3. *Result: this is* ***not*** *a valid number.*

Whenever the user enters a membership number, the system should perform the validation check to ensure that the number is valid, and display appropriate errors when an invalid number is entered.

4.3 Recording race results

The end user must be able to choose a racing driver (either through the use of a textbox in which the racing drivers membership number is entered, or by selecting an racing drivers name from a drop-down list).

Once the user has selected a racing driver, they must be able to record race results for a particular race for that racing driver. These results form part of the permanent history of the racing driver at the club.

5. Summary of development

1. Racing driver (member) management GUI used to add, edit, and delete racing drivers.
2. Race result recording GUI for recording racing drivers race results.
3. Motor racing event (race) management GUI used to add, edit, and delete, race details.
4. Classes/Structures to represent the business objects.
5. Classes/Modules for data storage.

6. Data storage

Phase 1 of this project must store all data in text files on disc using a simple comma separated file format. This requires the use of three files:

1. Racing driver data file
2. Race results data file
3. Event data file

The user must be able to specify the location of the data files.

Racing driver data	Store all collected data on the racing driver (see section 4.2) except race results.
Race results data	Store the racing driver's membership number, the race name, and the result.
Event data	Store all collected data on the event (see section 4.1)

Phase 2 of the project will move the data storage to a relational database. The client insists on phase 1 being delivered with file storage, and penalties will be incurred should this not happen.

7. Classes/Structures and business objects

To ensure a modular design, ensure that no data storage functions are located in GUI elements (VB forms). Additionally, the system may be expanded in the future, thus all business objects (racing drivers and events) must be represented using classes or structures.

It is perfectly acceptable to store a list of race results in the racing driver object (event name and result).

8. Acceptance

Document accepted as presented in current form by all parties involved. Development to commence as soon as resources are allocated.

A. Algorithms

```
Data: $b$ the member birthdate
Result: $m$ A membership number with a check-digit
$r \leftarrow$ random number between 0 and 999
$c \leftarrow$ two rightmost digits of current year
$m \leftarrow c \,\&\, b \,\&\, r$
$s \leftarrow 0$
for $i \leftarrow 1$ to length of $m$ do
    $s \leftarrow s +$ integer value of $m(i)$
end
$d \leftarrow s \bmod 10$
if $d = 0$ then $q \leftarrow 0$ else $q \leftarrow (10 - d)$
$m \leftarrow m \,\&\, q$
return $m$
```

Algorithm 1: Check digit calculation

```
Data: $m$ the membership number
Result: boolean True if the number is valid, false otherwise
$s \leftarrow 0$
$r \leftarrow$ false
for $i \leftarrow 1$ to length of $m$ do
    $s \leftarrow s +$ integer value of $m(i)$
end
$d \leftarrow s \bmod 10$
if $d = 0$ then
    $r \leftarrow$ true
end
return $r$
```

Algorithm 2: Membership number verification