A2 Software Systems Development Coursework

STUDIO2 Case Study

C O N T E N T S

A N A L Y S I S

Background Information

STUDIO2 is the second gym opened by business partners James King and Mark Sheerin. Both are qualified in fields relating to physical fitness, with James being a fully qualified physiotherapist and Mark, a Judo master with several titles. The centre is made of up of a gym, a yoga studio, a large fitness suite with a boxing ring and four therapy rooms which are all available for rental except for the fitness suite. All staff members have a sports qualification as well as a range of expertise in different areas. These are reviewed regularly to ensure the currency of staff qualifications. Classes in yoga, tai chi, aerobics, Pilates, dance, kick boxing and judo are available throughout the day as well as the fitness suite. Internally ran classes are free to members of STUDIO2 but a slot must be booked in advance as there is limited availability. The classes ran externally by external parties involve a fee. Classes and activities run by external instructors and organisations are available to not only STUDIO2 members but also to the public. All classes must be booked through reception. A number of different memberships are available such as individual membership, off peak and concessionary, all of which vary in price.

The business requires a substantially large database to handle all the different aspects and link them together into a central structure. For this reason, I will be focusing on building a database relating to the process of creating and managing memberships.

Problem Identification

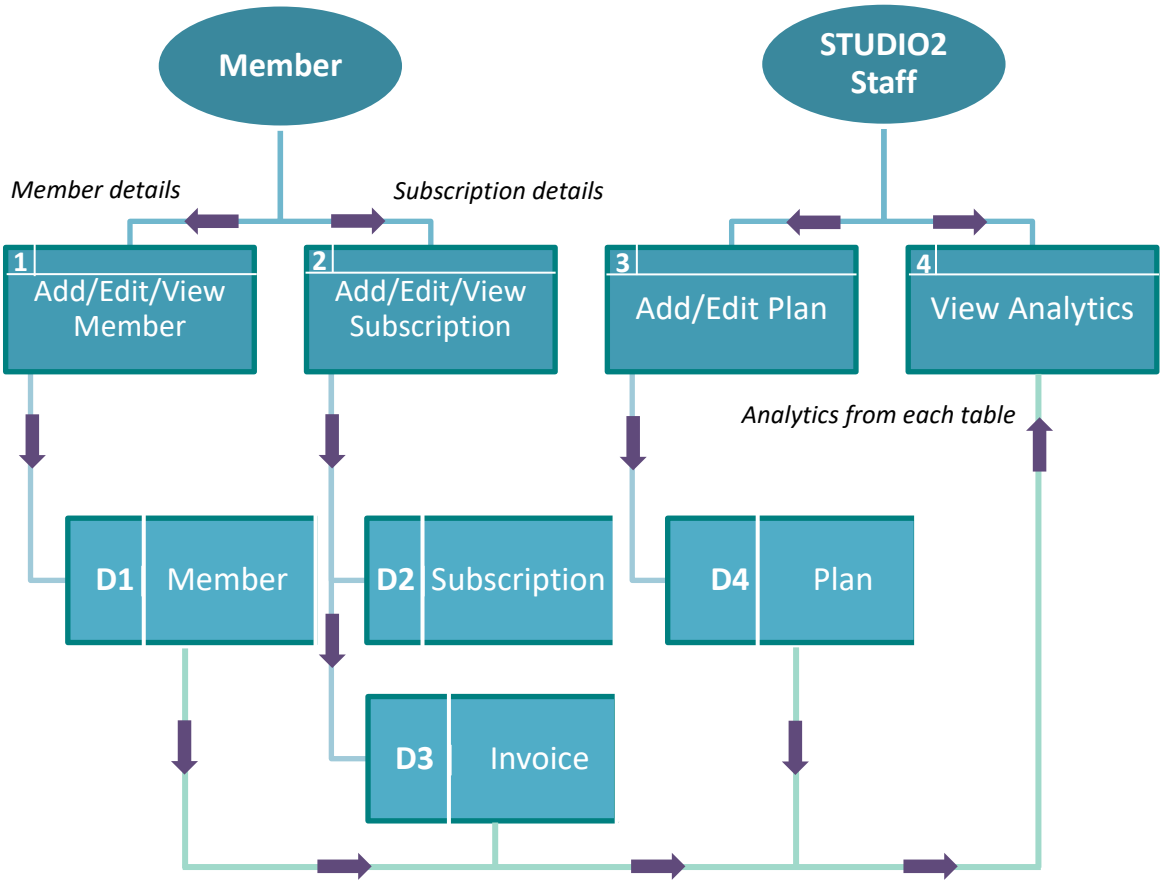
In order to build the subscription database several different factors must be taken into consideration. The first of these is the different subscriptions that are available: individual subscription, off peak, concessionary (over 60), concessionary off peak (over 60) and student. Each of these different subscriptions is price differently from each other. Secondly, subscriptions can be paid either by monthly instalments or by a one off single payment. By choosing a single payment, the overall cost to the customer works out to be 5% cheaper. Thirdly the price of each subscription varies depending on how long the subscription is contracted for. A twelve month contract is £2 cheaper per month than a six month contract for all subscription types. Similarly, the single payment works out cheaper overall for a twelve month contract than a six month contract (when you multiply the price of a six month contract by 2). With such an array of different prices, careful consideration will be needed to ensure that the prices are implemented properly so as to ensure that customers are never overcharged or not charged at all.

It is also vital for the system to be able to monitor when an individual's contract has expired or their monthly payment is due. To achieve this the date on which the subscription was commenced would need to be added to the database. If a subscription is being renewed, the date of renewal would need to replace the date value as if for example a six month contract was being renewed, the system would read the original date and immediately expire the subscription. Introducing a date variable would also allow STUDIO2 to setup an automated system to notify members via email or perhaps text message that they need to make another payment. This would also allow STUDIO2 to identify

any individuals who have failed to keep up with their payments and cancel their subscription if necessary which brings me onto the next point.

In addition, James and Mark mention wanting to have flexibility regarding subscription type and fees. To facilitate for economic fluctuations, the prices for each subscription type would need to be stored in a standalone table so that price changes will take effect throughout the database without requiring each piece of data mentioning price to be altered. In addition to this, there is a possibility that some members may wish to cancel their subscription so this functionality would need to be implemented which in turn, will assist in providing a flexible service.

Data Flow Diagram



System Approach

Waterfall

*'The **waterfall model** is a sequential design process, used in software development processes, in which progress is seen as flowing steadily downwards (like a **waterfall**) through the phases of conception, initiation, analysis, design, construction, testing, production/implementation and maintenance.' – Wikipedia*

The waterfall model is an effective process for the development of a product with a clear, defined set of requirements, which is being developed for a specific individual customer or niche group of customers; waterfall is used for the development of a bespoke package. When software is developed via waterfall, the client will only have a formal input in the early stages when the system requirements are being drawn up. They have minimal input during its development due to the fixed scope typically employed for this model. This makes it unsuitable for a project that involves rapid change in scope. However, by focusing on a fixed scope, the cost of producing the system and the time required to do so will also most likely remain fixed allowing the developer to identify the completion date to relative accuracy. Each stage in waterfall must be completed before the next can begin. If an error is found in one stage, a previous stage may have to be re-worked meaning that the utilisation of a waterfall model is likely not to be the quickest approach to developing a software system. The seven stages which make up waterfall are as follows:

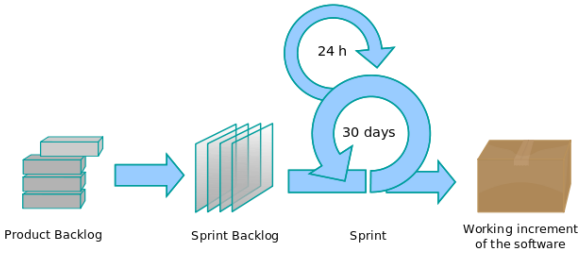
1. **Analysis:** A feasibility study is carried out to determine whether the project is worthwhile. Different methods of fact finding such as interviews, questionnaires, observation and documentation are used to carry out a detailed investigation of existing data flows/processing. The user requirements are also identified at this stage. This is the most crucial stage as any misinterpretation of what the client requires could lead to issues in the foreseeable future.
2. **Design:** Various deliverables based on the user requirements outlined in the previous stage are produced at this stage. These include: system specification, hardware/software specification, user interface/inputs and outputs design, data models, modular structure and the test strategy.
3. **Development:** The code for the system is produced from the design and continually tested by the development team. Technical documentation is also produced.
4. **Testing:** Integration and system testing are carried out by developers to identify any bugs and ensure that all modules that make up the system are operating with each other as intended. Acceptance testing is also carried out by the client.
5. **Implementation:** This stage involves retrieving and installing the necessary software and hardware required to run the system at the client's own site. It may require a suitable changeover method in order to convert existing data so that it is compatible with the new system. End user training may also take place to ensure the client is competent in operating the new system.
6. **Review:** When the system has been in use for an agreed period of time the developers and the client/users reflect on how well the system is performing. Factors to evaluate software include Usability, Effectiveness and Suitability.

7. **Maintenance:** Modifications will be made to the system by the developer after the system has been in use for some time or by feedback received from the review stage. After performing maintenance, the software will have to be retested (regression testing)

Agile Models

- ***Scrum:*** Scrum is one of the most popular Agile methodologies utilised. It involves the carrying out of sprints to incrementally build the system.

  Before a sprint can be performed, planning is carried out during a sprint planning event where the product backlog (which is made up of high-level user requirements or stories) is used to define a sprint backlog. The product backlog comprises item which are chosen by the product owner, an individual representing the client, who also focuses on managing the business side of the project. These items will be ranked in priority depending on the business value they hold i.e. if two different items require the same amount of work to develop but one provides a higher return on investment, it is likely to have a higher priority. However, the development team will also provide input into the time and work that would be required to achieve said items, which in collaboration with the product owner's initial input, will result in the formation of user stories. The product owner also may wish to focus on tasks which could prove to be much more difficult to accomplish, and so wish to have these taken care of earlier on in the project, as they will carry a greater risk.



  The process of choosing the items which are to be included in the sprint backlog is conducted collaboratively by the whole team, not just by the scrum master. High priority items tend to be focused on in early sprints in order to deliver a product of the highest business value in the shortest timeframe. These sprints tend to have duration of 1-4 weeks.

  The technical side of the project is managed by the scrum master. The scrum master is responsible for ensuring that the team are not adversely affected by any obstacles during development which may arise. In addition to this, the scrum master is also tasked with safeguarding the functionality and productivity of the team to ensure that team members are cooperating with each other effectively. This will ensure that the end product produced will be both robust enough to meet the client's requirements and also be built within the allocated time frame.

  One of key component which differentiates Scrum from other Agile methodologies is the daily Scrum. This a short meeting conducted every day in which the scrum master will ask each member of the team 3 questions:

  1. What did you do **yesterday**?
  2. What will you do **today**?
  3. What **problems** have you encountered?

Should it arise during the daily Scrum that there may be a problem which could impede the team's ability to meet the sprint's requirements; the Scrum master will assign someone to resolve these problems.

At the end of a sprint, a sprint review is carried out. At this stage the team presents what has been accomplished during the sprint to stakeholders to reflect on the progress that has been achieved, discuss what went well, problems encountered and the steps taken to rectify any problems. Feedback can then be provided to the product owner who in turn provides the team with change requests which can be implemented in later sprints.

After the sprint review the whole team will participate in a sprint retrospective. This is similar to the sprint review in respect to identifying what went well during the sprint and areas of improvement however in addition to this the information learned will be used to improve the effectiveness of the team in the next sprint and plan how changes should be implemented.

- ***XP (eXtreme Programming):*** XP is another widely used Agile methodology. XP typically takes the common industry practices and brings them to an extreme level as a means of building high quality software but in the shortest time and with the lowest cost. In order to achieve this, only high priority requirements are implemented into the system but in such a manner that the customer is provided with the software they need as they need it, rather than at a distant date in the future, by releasing packages after short iterations. Testing on all levels is also conducted while code is being developed where the test plans are created before the code begins and which are then maintained during development. By doing this, the iteration cycles can be kept short and the customer provided with continual evidence of developments in the system.

  A particular emphasis is also put on having the entire team including the customer, work together closely during development. By constantly communicating with the customer, feedback can be provided as soon as something is added/changed to the system, increasing the speed at which the system can be developed. Another key element found in XP is the use of pair programming. By having 2 programmers work together, one can focus on writing code while the other can review the code by searching for any vulnerabilities or mistakes. The end result of this is that the system may take slightly longer to produce however the overall synergies produced with using this approach will more so than often mean that the number of defects in the code is reduced and the overall quality of the system is improved.

  In general though, XP relies on 5 core values:

  1. **Communication:** Miscommunication among team members can result in specific code or functions being implemented incorrectly or unnecessarily. Because of this time may be wasted by for example focusing on an area that isn't of a high priority. Since time has been wasted, finance too will be wasted as well, which is likely to displease the customer. To ensure that exactly what the customer wants is what is

produced, communication is vital among team members but also so that they can provide each other with guidance and assistance when problems are encountered.

2. **Simplicity:** By focusing on choosing the simplest route to building the system, both cost and time minimisation is maximised. Moving in simple steps will also allow progress to be easily tracked and for any changes to be reverted if necessary without much issue.
3. **Feedback:** This is one of the most important components of any Agile methodology. XP is designed in such a way to allow the customer to provide regular feedback so that changes can be brought in quickly during development through the use of short feedback cycles.
4. **Courage:** It is important that team members have the courage to voice their opinions on any ideas or issues they have with the system. In particular, it is necessary for revealing any programming fatalities as the existences of weaknesses such as this in the system will damage the quality of the software and inevitably displeases the customer when uncovered. Put simply, team members need to be willing to take effective action in the face of fear.
5. **Respect:** All members of the team from developers to management level are valued which is what provides those working with an XP approach with a high motivation to their work. A respectful culture in the workplace also helps to create a positive environment, which encourages greater ingenuity and innovation.

Choosing a methodology

The methodology chosen depends on a number of factors. Since the system being produced is bespoke it may be of greater benefit to choose to undertake a Waterfall approach. At the initial planning stage of the project, a fixed set of user requirements can be created. With the task being clear and defined, involving converting a paper based system to a digital system, the scope of the project can be accurately determined. Because of this, it is likely that after user requirements have been completed, STUDIO2 will not need to make any major changes to the requirements as the core objective remains the same regardless of any external factors. If the system were to be developed for something that is continually changing or which has an unclear set of requirements, such as for a rapidly growing market of some sort, the use of an Agile approach would be the better choice as they are designed to facilitate change during the project's development through the facilitation of constant interaction between developers and customers. However, since this isn't the case for STUDIO2, that makes Waterfall a more favourable choice as by having a clearly defined fixed scope, the cost and the time required for the project can be predicted to reasonable accuracy, which is of importance for STUDIO2 as there is a limited timeframe in which the project can be completed.

On the other hand though, it could be argued that choosing to utilise Scrum or XP may be more beneficial. At the end of a sprint/cycle a functional system is produced. Although it may only meet a select number of requirements, this will continue to grow as more time is spent developing the system. This means that the process of converting paper based files into the digital system can be started during the development of the system. For example, if the member module of the system were completed at an early stage, paper based member records could start being converted. With Waterfall, the next stage of the cycle can only be started after the current stage is completed. Because of this, implementation of data conversion and changeover can only be started after the

finished system is completed, which may result in a lengthier completion period as opposed to an Agile approach.

Overall I believe that for this project, choosing to conduct development through the utilisation of the Waterfall methodology would prove to be the more appropriate decision. The requirements originally outlined by STUDIO2 are easily identified and clearly defined and so it is likely that there will be very few changes needed during development. In addition to this, due to the client having minimal input during development when Waterfall is used, STUDIO2 will be able to focus on its core activities, running its two gyms, rather than having to focus time in the development of the database system. On the other hand, the utilisation of Agile would provide STUDIO2 with an insight into the development of the project due to the working systems produced at the end of each sprint/cycle. In the case of XP, the system could be rapidly built and at a low cost which from a business viewpoint makes it a favourable choice. However since XP involves prioritising requirements so that only those of the highest importance are implemented, there is the risk that some minor features may be missing which in turn may require maintenance to the system sometime in the future, which will also serve to create additional costs in the future.

Time Plan


TASK | WEEK
ANALYSIS
Contents
Background Information
Problem Identification
System Approach
Time Plan
User Requirements
DESIGN
Data Model
Normalisation
Data Dictionary
Storyboard
DEVELOPMENT
Development
TESTING
Test Plan
Testing Evidence
EVALUATION
Evaluation of User Requirements
Evaluation of Approach to Solution
Evaluation of Testing Procedures
Evaluation of Solution
Evaluation of Own Performance
Developer Diary
02/11/2015
09/11/2015
16/11/2015
23/11/2015
30/11/2015
07/12/2015
14/12/2015
21/12/2015
28/12/2015
04/01/2016
11/01/2016
18/01/2016
25/01/2016
01/02/2016
08/02/2016
15/02/2016
22/02/2016
29/02/2016
07/03/2016
14/03/2016
21/03/2016
28/03/2016
04/04/2016
11/04/2016
18/04/2016
25/04/2016

TASKS	RESOURCES		
	Personnel	**Software**	**Hardware**
Contents	Myself	Microsoft Word	School and home computer
Background Information	Myself and STUDIO2 representative	Microsoft Word	School and home computer
Problem Identification	Myself	Microsoft Word	School and home computer
System Approach	Myself	Microsoft Word	School and home computer
Time Plan	Myself	Microsoft Word	School and home computer
User Requirements	Myself and STUDIO2 representative	Microsoft Word	School and home computer
Data Model	Myself	Microsoft Word	School and home computer
Normalisation	Myself and teacher	Microsoft Word	School and home computer
Data Dictionary	Myself	Microsoft Word	School and home computer
Storyboard	Myself	Microsoft Word	School and home computer
Development	Myself	Visual Studio and SQL Server Management Studio	School and home computer
Test Plan	Myself	Microsoft Word and Visual Studio	School and home computer
Test Evidence	Myself	Microsoft Word , Visual Studio and SQL Server Management Studio	School and home computer
Evaluation of User Requirements	Myself	Microsoft Word	School and home computer
Evaluation of Approach to System	Myself	Microsoft Word	School and home computer
Evaluation of Testing Procedures	Myself	Microsoft Word	School and home computer
Evaluation of Solution	Myself	Microsoft Word	School and home computer
Evaluation of Own Performance	Myself	Microsoft Word	School and home computer
Developer Diary	Myself	Microsoft Word	School and home computer

User Requirements

Functional Requirements

- SQL database must contain data for members, subscriptions, subscription plans and invoices
- System must allow user to connect to a locally hosted SQL database
- System must allow members to be created
- System must allow members' details to be updated
- System must display members in a table
- System must allow user to search for member records which match a search query
- System must allow subscriptions to be created
- System must allow subscriptions to be canceled
- System must display subscriptions in a table
- System must allow user to search for subscriptions records which match a search query
- System must allow invoices to be created
- System must display invoices in a table
- System must allow user to search for invoices records which match a search query
- System must display subscription plans in a table
- System must allow prices of subscription plans to be changed
- System must automatically identify when invoices need to be generated and carry out this task
- System must automatically identify invoices which have not been paid and cancel the associated subscription
- System must allow user to print out records from any table
- System must provide detailed analytical data in the form of graphs and statistics

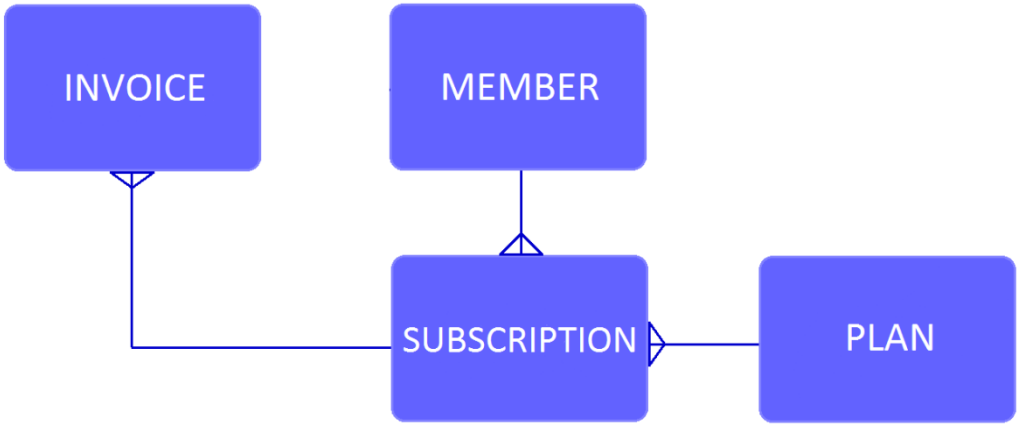
Non-Functional Requirements

- A user friendly Graphical User Interface (GUI) should be created using Windows Forms
- Try Catch blocks should be utilised to catch errors and prevent the application from unexpectedly crashing
- Code should be commented properly to allow for maintenance and maximum reusability
- Application should load and perform operations within a reasonable time
- Appropriate measures should be taken to provide validation

In order to conduct the entirety of the project, all work will be done in the Programming Environment. Visual Studio Express 2013 is required for the development of the user interface and internal code of the system. SQL Server Management Studio is required to create and access the SQL database. To write up documentation, Microsoft Word is needed, along with Microsoft Excel to create the Gantt chart. For deployment of the application, the client will need SQL Server Management Studio installed on their machine(s), a backup of the SQL database, an executable version of the finished application and .NET framework to run the application. In terms of hardware, development and documentation can be done on a relatively low range PC as the software utilised is not particularly resource intensive. Similarly, it is intended that the finished product will be able to function properly on a low spec PC, providing the application with high usability.

D E S I G N

Data Model



A member can create many subscriptions but a subscription can only be linked to one member. Each subscription can create many invoices but an invoice can be linked to only one subscription. In addition, each subscription can only be linked to one plan but each plan can be linked to many subscriptions.

Normalisation

0NF

All of the individual fields present in the database

MEMBER (MemberID, Title, FirstName, Surname, AddressLine, AddressCity, AddressCounty, AddressPostcode, DateOfBirth, EmergencyContactNumber, Gender , Phone, Email, PlanID, Name, Amount, Months, Interval, SubscriptionID, MemberID, PlanID, Recurring, SusbcriptionStartDate, PeriodStartDate, PeriodEndDate, CanceledDate, NextInvoice, SStatus, InvoiceID, IssueDate, SubscriptionID, IStatus)

1NF

A table is in first normal form when repeated groups of fields are removed and placed in a separate table. Since the plan related fields are repeated, they are placed into a separate table.

MEMBER (MemberID, Title, FirstName, Surname, AddressLine, AddressCity, AddressCounty, AddressPostcode, DateOfBirth, EmergencyContactNumber, Gender , Phone, Email, PlanID, SubscriptionID, MemberID, PlanID, Recurring, SusbcriptionStartDate, PeriodStartDate, PeriodEndDate, CanceledDate, NextInvoice, SStatus, InvoiceID, IssueDate, SubscriptionID, IStatus)

PLAN (PlanID, Name, Amount, Months, Interval)

2NF

A table is in second normal form when it is in first normal form and partial key dependencies are removed. No changes are made since no partial key dependencies are present.

MEMBER (MemberID, Title, FirstName, Surname, AddressLine, AddressCity, AddressCounty, AddressPostcode, DateOfBirth, EmergencyContactNumber, Gender , Phone, Email, PlanID, SubscriptionID, MemberID, PlanID, Recurring, SusbcriptionStartDate, PeriodStartDate, PeriodEndDate, CanceledDate, NextInvoice, SStatus, InvoiceID, IssueDate, SubscriptionID, IStatus)

PLAN (PlanID, Name, Amount, Months, Interval)

3NF

A table is in third normal form when it is in second normal form and transitive key dependencies are removed. Both subscription and invoice related fields are removed and placed into corresponding tables to reduce duplication of data.

MEMBER (MemberID, Title, FirstName, Surname, AddressLine, AddressCity, AddressCounty, AddressPostcode, DateOfBirth, EmergencyContactNumber, Gender , Phone, Email)

PLAN (PlanID, Name, Amount, Months, Interval)

SUBSCRIPTION (SubscriptionID, MemberID, PlanID, Recurring, SusbcriptionStartDate, PeriodStartDate, PeriodEndDate, CanceledDate, NextInvoice, SStatus)

INVOICE (InvoiceID, IssueDate, SubscriptionID, IStatus)

Data Dictionary

MEMBER			
Field	**Data Type**	**Size**	**Validation**
MemberID	INT		Primary Key
Title	VARCHAR	30 chars	Not null
FirstName	VARHCAR	30 chars	Not null
Surname	VARCHAR	30 chars	Not null
AddressStreet	VARCHAR	45 chars	Not null
AddressCity	VARCHAR	30 chars	Not null
AddressCounty	VARCHAR	30 chars	Not null
AddressPostcode	VARCHAR	8 chars	Not null
DateOfBirth	DATE		Not null
EmergencyContactNumber	VARCHAR	15 chars	Not null
Gender	VARCHAR	6 chars	Not null
Phone	VARCHAR	15 chars	Not null
Email	VARCHAR	30 chars	Not null

SUBSCRIPTION			
Field	**Data Size**	**Size**	**Validation**
SubscriptionID	INT		Primary key
MemberID	INT		Composite key
PlanID	INT		Composite key
Recurring	BIT		Not null
SubscriptionStartDate	DATE		Not null
PeriodStartDate	DATE		Not null
PeriodEndDate	DATE		Not null
CanceledDate	DATE		
NextInvoice	DATE		
SStatus	VARCHAR	10 chars	Not null

INVOICE			
Field	**Data Size**	**Size**	**Validation**
InvoiceID	INT		Primary key
IssueDate	DATE		Not null
SubscriptionID	INT		Foreign key
IStatus	VARCHAR	10 chars	Not null

PLAN			
Field	**Data Size**	**Size**	**Validation**
PlanID	INT		Primary key
Name	VARCHAR	50 chars	Not null
Amount	FLOAT		Not null
Months	INT		Not null
Interval	VARCHAR	7 chars	Not null

SQL Pseudo Code

SQL CREATE QUERYS

```
CREATE TABLE Member (
	MemberID INT IDENTITY(1,1) PRIMARY KEY,
	Title VARCHAR(30) NOT NULL,
	FirstName VARCHAR(30) NOT NULL,
	Surname VARCHAR(30) NOT NULL,
	AddressLine VARCHAR(45) NOT NULL,
	AddressCity VARCHAR(30) NOT NULL,
	AddressCounty VARCHAR(30) NOT NULL,
	AddressPostcode VARCHAR(8) NOT NULL,
	DateOfBirth DATE NOT NULL,
	EmergencyContactNumber VARCHAR(15) NOT NULL,
	Gender VARCHAR(6) NOT NULL,
	Phone VARCHAR(15) NOT NULL,
	Email VARCHAR(30) NOT NULL
);

CREATE TABLE [Plan] (
	PlanID INT IDENTITY(1,1) PRIMARY KEY,
	Name VARCHAR(50) NOT NULL,
	Amount FLOAT NOT NULL,
	Months INT NOT NULL,
	Interval VARCHAR(7) NOT NULL
);

CREATE TABLE Subscription (
	SubscriptionID INT IDENTITY(1,1) PRIMARY KEY,
	MemberID INT FOREIGN KEY (MemberID) REFERENCES Member(MemberID),
	PlanID INT FOREIGN KEY (PlanID) REFERENCES [Plan](PlanID),
	Recurring BIT NOT NULL,
	SubscriptionStartDate DATE NOT NULL,
	PeriodStartDate DATE NOT NULL,
	PeriodEndDate DATE NOT NULL,
	CanceledDate DATE,
	NextInvoice DATE,
	SStatus VARCHAR(10) NOT NULL
);

CREATE TABLE Invoice (
	InvoiceID INT IDENTITY(1,1) PRIMARY KEY,
	IssueDate DATE NOT NULL,
	SubscriptionID INT FOREIGN KEY (SubscriptionID) REFERENCES
Subscription(SubscriptionID),
	IStatus VARCHAR(10)
);
```

SQL SELECT QUERYS

```
SELECT * FROM Member WHERE FirstName LIKE '%Mary%';

SELECT * FROM Invoice WHERE IStatus = 'Pending';

SELECT COUNT(*) FROM Subscription Where PlanID BETWEEN 13 AND 16;

SELECT AVG(DATEDIFF(year,DateOfBirth,GETDATE())) FROM Member;
```

SQL INSERT QUERYS

```
INSERT INTO Member VALUES ('Mr','Bob','Smith','28 Park
Street','Dungannon','Tyrone','BT72
4JN','1988/04/02','072837192','Male','077375182','bob@email.com');

INSERT INTO Subscription VALUES (2, 1, 1,'2016-04-27','2016-04-27','2017-04-27',
NULL,'2016-05-27','Active'); SELECT SCOPE_IDENTITY();

INSERT INTO Invoice VALUES ('2016-04-27', 5,'Pending');
```

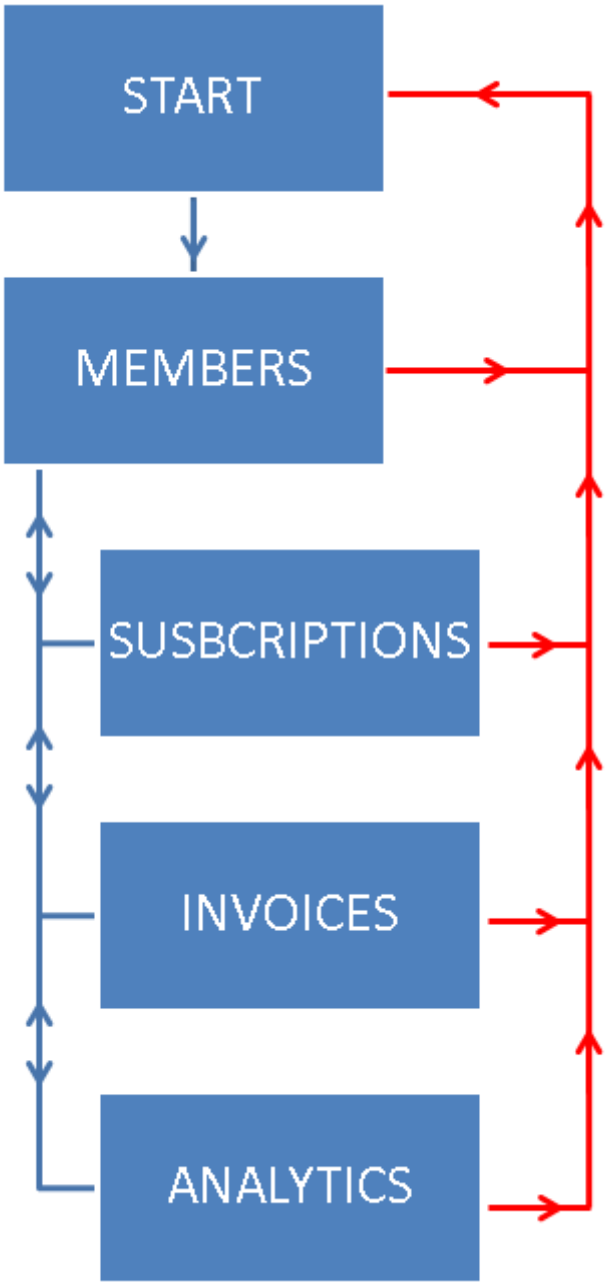
SQL UPDATE QUERYS

```
UPDATE Member SET Title='Ms', FirstName='Mary', Surname='Walsh', AddressLine='28
Hillview', AddressCity='Donaghmore', AddressCounty='Tyrone', AddressPostcode='BT77
64N', DateOfBirth='1990-08-02', EmergencyContactNumber='07762719311', Gender='Female',
Phone='07712839911', Email='mmdmore@gmail.com' WHERE MemberID = 2;

UPDATE Subscription SET Recurring = 'False' WHERE SubscriptionID = 5;

UPDATE Subscription SET SStatus = 'Canceled' WHERE SubscriptionID = 5; UPDATE
Subscription SET CanceledDate = '27/04/2016' WHERE SubscriptionID = 5;

UPDATE Invoice SET IStatus = 'Canceled' WHERE SubscriptionID = 5 AND IStatus =
'Pending';
```

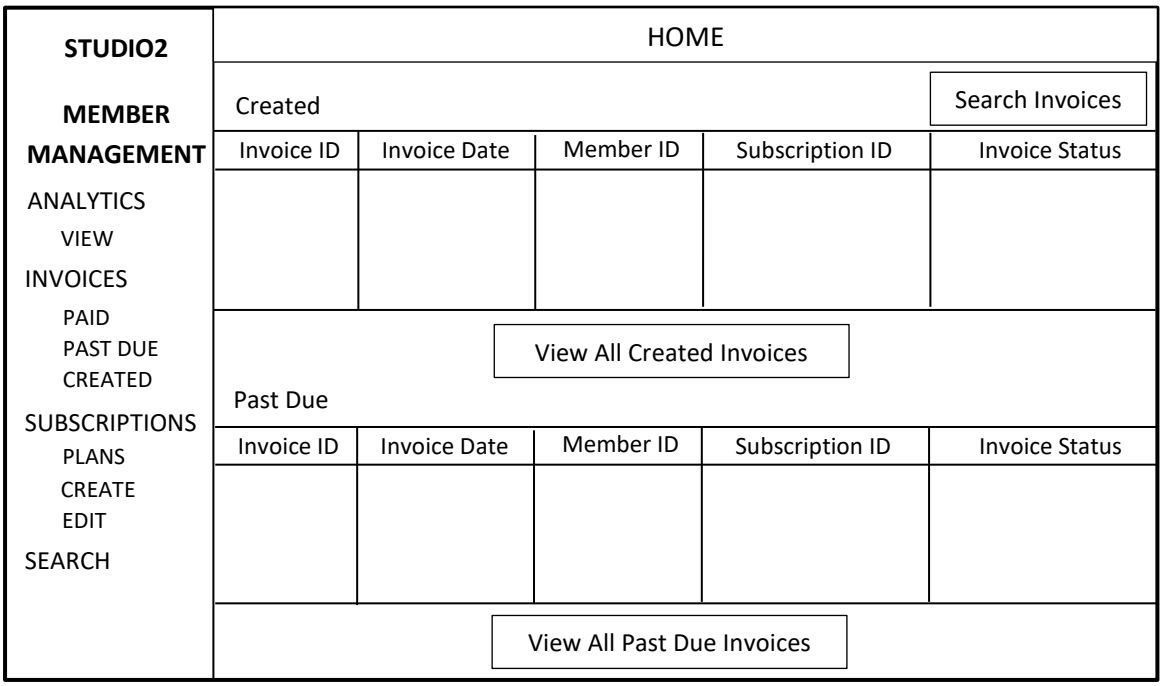
SQL DELETE QUERYS

```
DELETE FROM Invoice WHERE SubscriptionID = 5;

DELETE FROM Subscription WHERE MemberID = 2;

DELETE FROM Member WHERE MemberID = 2;
```

Form Navigation Map



Storyboards

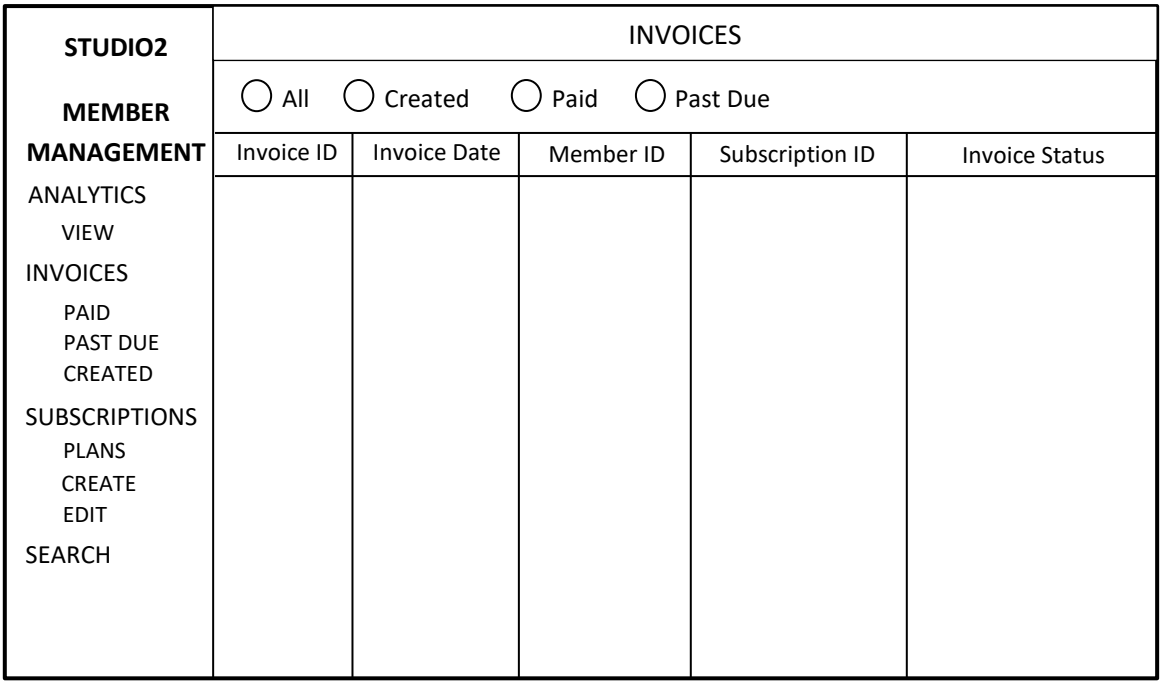


Home

- Click event raised on 'View All Invoices' Button – Displays 'Invoices' form and shows created invoices
- Click event raised on 'Home' Button – Does nothing as 'Home' form is already open

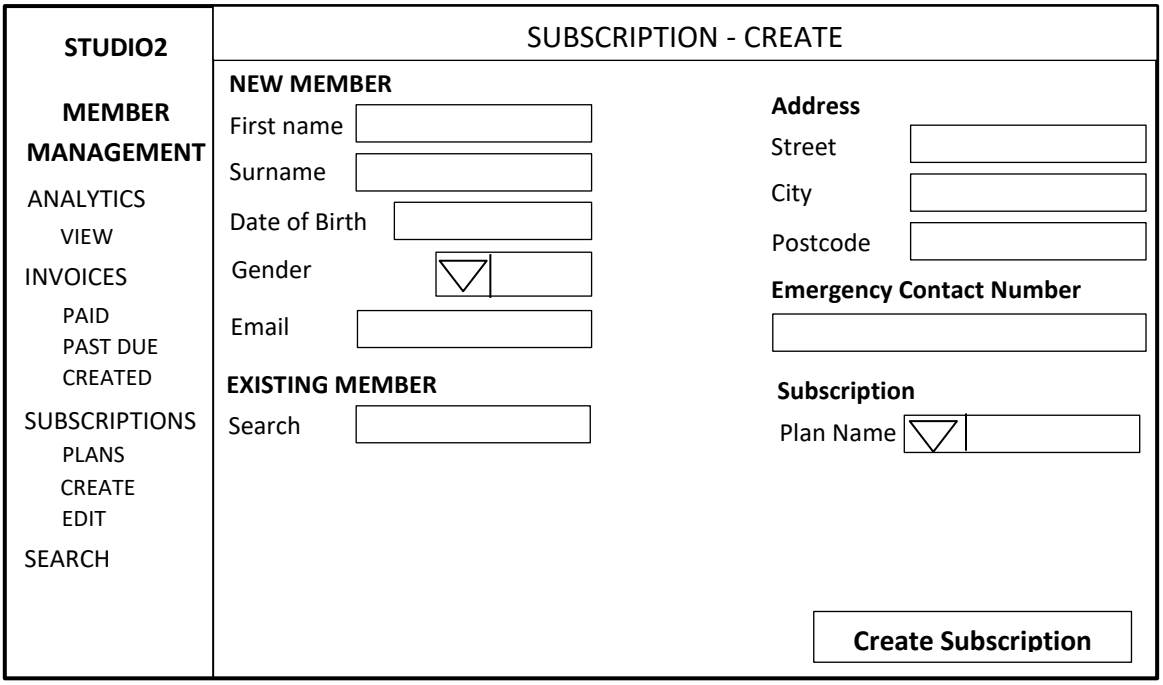
For all Forms

- Click event raised on 'View' Button – Displays 'Analytics' form
- Click event raised on 'Paid' Button – Displays 'Invoices' form and shows paid invoices
- Click event raised on 'Past Due' Button – Displays 'Invoices' form and shows past due invoices
- Click event raised on 'Created' Button – Displays 'Invoices' form and shows created invoices
- Click event raised on 'Plans' Button – Displays 'Subscriptions' form and shows plans
- Click event raised on 'Create' Button – Displays 'Subscriptions' form and shows create member GroupBox
- Click event raised on 'Edit' Button – Displays 'Subscriptions' form and shows edit member GroupBox



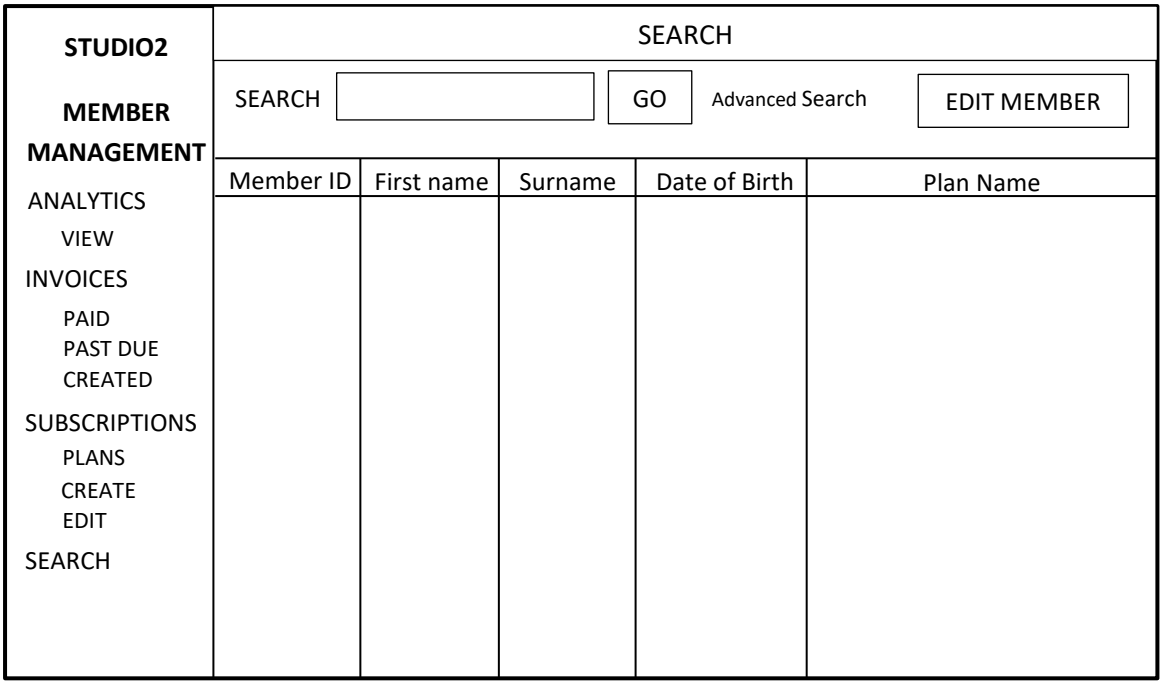
Invoices – View All Invoices Clicked

- On form load, all invoices are displayed in table. They are ordered by 'Invoice ID'
- Click event raised on 'All' RadioButton – All invoices displayed in table
- Click event raised on 'Created' RadioButton – Invoices with 'Created' status are displayed in table
- Click event raised on 'Paid' RadioButton – Invoice with 'Paid' status are displayed in table
- Click event raised on 'Past Due' RadioButton – Invoices with 'Past Due' status are displayed in table



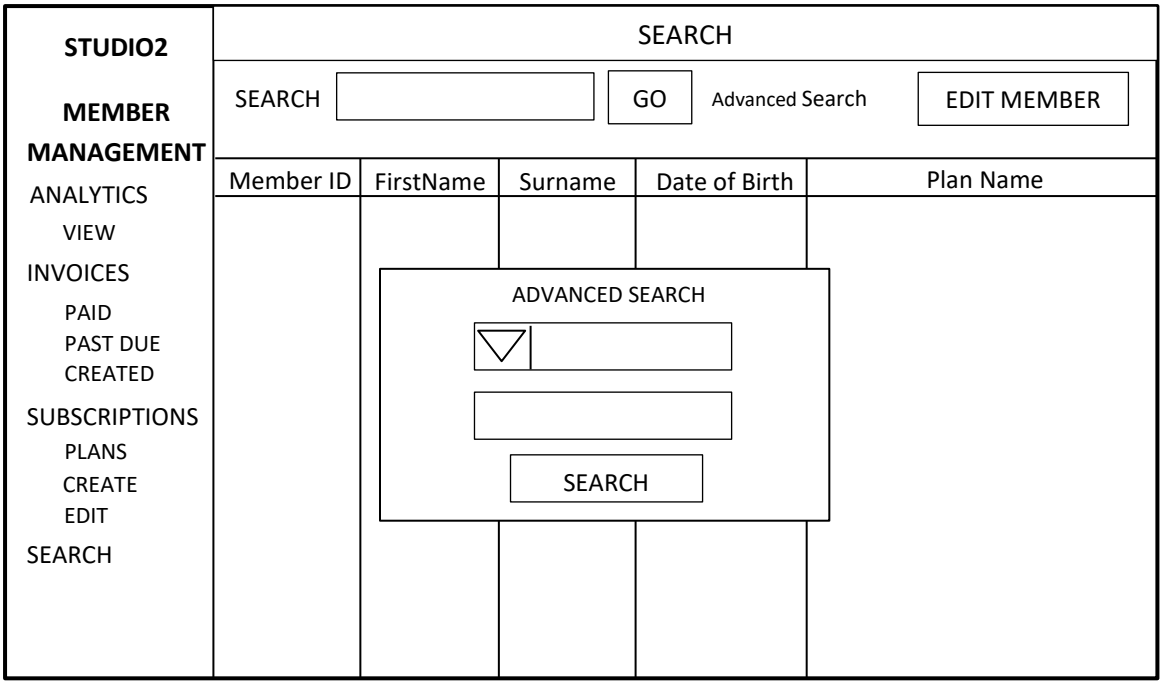
Subscription - Create

- Details entered into TextBoxes
- Click event raised on 'Date of Birth' DateTimePicker – Displays calendar from which date of birth can be selected
- Click event raised on 'Gender' ComboBox - Displays 'Male' or 'Female'
- Click event raised on 'Type' ComboBox - Displays 'Standard', 'Off Peak', 'Concessionary', 'Concessionary Off Peak' or 'Student'
- Click event raised on 'Method' ComboBox - Displays 'Single Payment' or 'Monthly'
- Click event raised on 'Create Membership' Button – Checks if all TextBoxes have been filled.



Search

- On Form load, table displays all members
- Search value entered into 'Search' TextBox
- Click event raised on 'Go' button – Creates an SQL query to show corresponding data for any members which match the Text of the 'Search' TextBox. If any matching data is found, results are shown in table
- Click event raised on 'Advanced Search' Label clicked – 'Advanced Search' GroupBox Visible property set to true (see Search – Advanced Search storyboard)
- Click event raised on 'Edit Member' Button – 'Edit Member' GroupBox Visible property set to true (see Search – Edit Member)

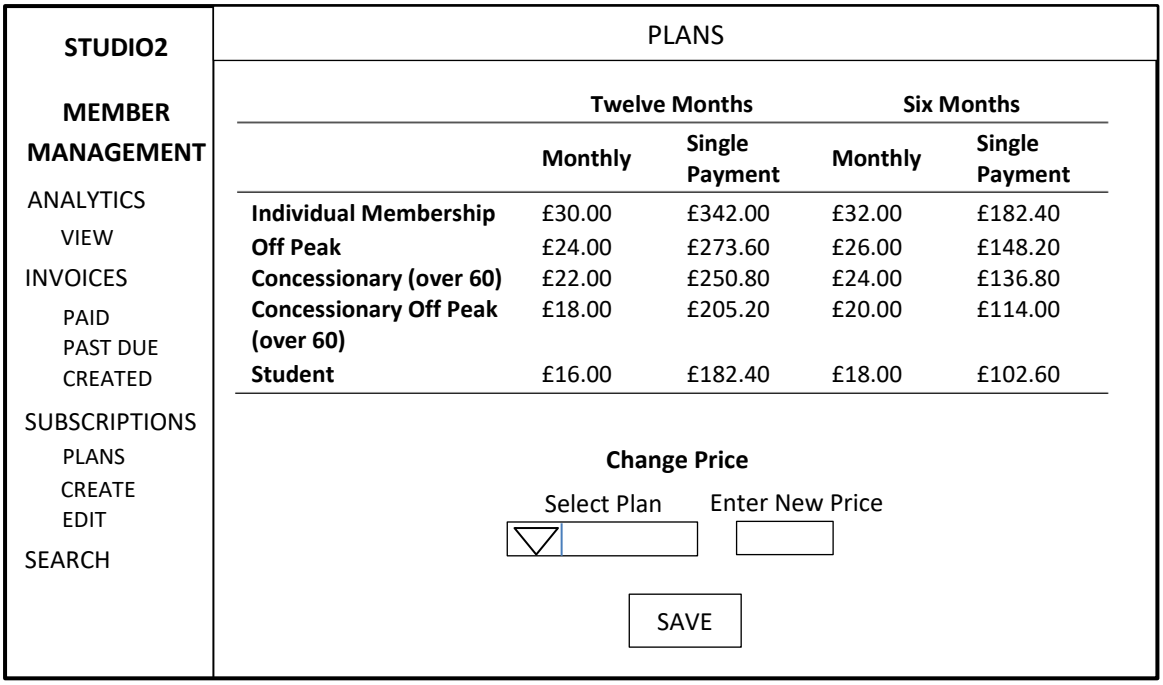


Search – Advanced Search

- Search value entered into 'Advanced Search' TextBox
- Click event raised on 'Advanced Search' ComboBox – Displays other fields which can be used to find results i.e. forename, surname, date of birth etc.
- Click event raised on 'Search' Button – Creates an SQL query to show corresponding data for any members who match the search parameters set using the combo box and Text property of the 'Advanced Search' TextBox. If any matching data is found, results are shown in table

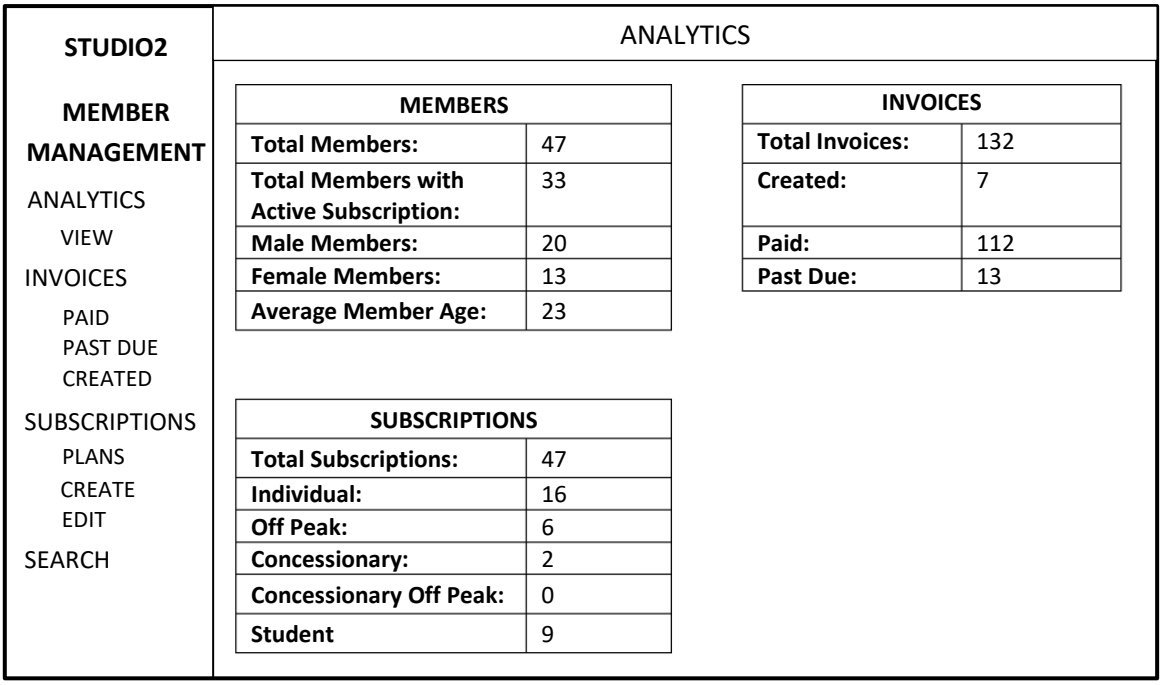
Subscriptions - Edit

- On Load – Data retrieved from database using Member ID as primary key. Data loaded into corresponding TextBoxes
- Shaded TextBoxes Enabled property set to False. This data cannot be directly altered by the user
- Click event raised on 'Cancel Subscription' Button – A warning yes/no message box displays with Text 'Are you sure you wish to cancel this subscription?'. If user selects 'Yes' the current subscription will be cancelled, by setting the subscription status to 'Cancelled'
- Click event raised on 'Save Changes' Button – Current data for selected member present in the database is overwritten with the data in the text boxes



EDIT PLANS

- On form load - Data for prices retrieved from database
- Click event raised on 'Select Plan' ComboBox – All types of plan displayed for example individual-12mnth-monthly
- Click event raised on 'Save' Button – SQL update used to change plan currently selected in 'Select Plan' ComboBox. The new price is retrieved from the 'Enter New Price' text box. The data table is then updated to display the new price. If the plan selected in the 'Select Plan' ComboBox is valid but an invalid price is present in 'Enter New Price', a message box displays saying 'Invalid value for new price field'. If an invalid plan is selected but a valid price is present, a message box displays saying 'Invalid value for plan field'



Analytics

- On Form load – Analytics data retrieved with various SQL SELECT statements

T E S T I N G

Test Plan

The importance of testing cannot be understated. Without testing, the system you produce will be rampant with bugs and vulnerabilities diminishing the quality of the software and making it unattractive to clients or potential users. The stage at which testing occurs during the development varies depending on the system methodology used, however generally some form of testing is always performed early on as this helps to identify any problems early on, saving time and money. There are various types of testing conducted:

1. **Unit Testing:** A small block of code is tested in an isolated environment using both black box and white box testing. A test harness is often used to pass in data rather than implementing the code within a real system. Since this form of testing requires the creation of temporary data in order to test the units, the programmer or a related individual will usually perform the test. As a result of this, this form of testing is frequently done during the development phase.
2. **Integration Testing:** Multiple modules/units are tested together as a single system to determine whether they operate together as intended. Since this type of testing focuses on the flow of data from one module to another it is mainly black box. Who conducts the tests depends on the type of integration testing being carried out. If the testing involves low-level modules being tested together, the developer will design and carry out the tests, however if it is more high-level, it is likely that the analyst who originally designed the system will carry out the tests since they already have knowledge of how the classes should interact with each other.
3. **System Testing:** The complete system is tested. Unlike unit or integration testing, no test harness is used. Instead the system will be tested in a in a simulated environment resembling the client's own environment. These tests are performed by analysts or a group of independent testers. Only black box testing is performed here as functional and non-functional tests are carried out focusing on user requirements and system requirements respectfully.
4. **Acceptance Testing:** The complete system is tested but is tested by the client. This allows the client to determine whether the system meets their requirements. Direct feedback can be provided from the client to determine any amendments needed. Similar to system testing, this is solely black box. There are two distinct types of acceptance testing:
    - **Alpha:** Early stage testing typically conducted either at the developer's own location or at the client's location, under conditions specifically tailored by the developer i.e. using software and hardware provided by the developer. This can incorporate unit, module and system testing.
    - **Beta:** Testing which is conducted by potential users at their own chosen location and under their own chosen conditions. This stage of testing is only begun after alpha testing has been completed. Feedback from this stage is particularly useful for determining the performance of the system in a real working environment with real data.

1. **Black Box Testing:** Testing to determine the resultant outputs produced when using certain inputs. Black box testing can be performed at any level (unit, integration or system). Since it does not involve the internal code unlike white box testing, black box can be conducted by anyone. In addition, the tests are based on requirements produced in the early design stages of development, so test plans can also be created at an early stage. The critical issue with black box is that there is no way of determining that all code paths have actually been tested meaning that some areas may contain unknown issues which may only arise after deployment.
2. **White Box Testing:** Testing to analyse whether the internal code of the system is performing as intended. Since it requires knowledge of the internal workings of the system, the developer is likely to be the one who performs these tests. This form of testing allows the location of any bugs or errors to quickly be identified due to the developers own knowledge of the system.

Test Number	Test Description	Reason for Test	Expected Outcome	Outcome	Corrective Action	Evidence (page no.)
			S T A R T			
1-1	Load Start form	To check form Load event is raised	Start form appears. Text property of txtServer and txtDatabase set to “”	Start form appeared. Text property of txtServer and txtDatabase set to “”	No corrective action required	48
1-2	Enter server and database details (LOCALHOST and STUDIO2 Subscription Manager SQL) then click ‘Connect’ button	To check that the Click event on the ‘Connect’ button is working and that connection to the SQL database is functioning as intended	MessageBox appears stating that the connection was successful. The form then changes to Members form which should retain the same height and width of the Start form	MessageBox appeared stating that the connection was successful. The form then changed to Members form which retained the same height and width of the Start form	The test performed as expected however I encountered an issue in which when the user attempted to connect to the database but on a different machine, the connection would be successful but the FullSystemCheck would fail. However when the user simply restarted the application and tried again it work. Researching into this, I found that the configuration file is cached during runtime and any changes made will only take affect after (due to the caching) the application is	48

					started again. Because of this the only alternative I had was to initiate a new instance of the application when this scenario occurred which I wasn't particularly happy about doing since it seems like a bit of an unprofessional method of doing it.	
1-3	Verify that the FullSystemCheck method is functioning properly by connecting	To check that the FullSystemCheck method which acts as the automatic creator of new invoices, subscriptions and suspender of subscriptions, is functioning properly	MessageBox detailing that the connection to the database was successful also specifies that the FullSystemCheck() was successful	MessageBox detailing that the connection to the database was successful also specified that the FullSystemCheck() was successful	No corrective action required	50
1-4	Run FullSystemCheck when an active subscriptions next invoice is set at a date 7 days from the current date and the NextInvoice value is not equal to the PeriodEndDate of the subscription. Current date is 13/04/2016	To check that FullSystemCheck functions	New invoice is generated and NextInvoice field of the selected subscription record is updated	New invoice was generated and NextInvoice field of the selected subscription record updated	No corrective action required	50
1-5	Run FullSystemCheck when an active subscriptions next invoice is set at a date 7 days from the current date and the NextInvoice value is equal to the PeriodEndDate of the subscription and Recurring value is False. Current date is 13/04/2016	To check that FullSystemCheck functions	Nothing happens	Nothing happened	No corrective action required	50
1-6	Run FullSystemCheck when an active subscriptions next invoice is equal to the current date and the NextInvoice value is equal to the PeriodEndDate of the	To check that FullSystemCheck functions	SStatus of subscription is set to 'Expired'	SStatus of subscription set to 'Expired'	No corrective action required	51

	subscription and Recurring value is False. Current date is 13/04/2016					
1-7	Run FullSystemCheck when an active subscriptions next invoice is equal to the current date and the NextInvoice value is equal to the PeriodEndDate of the subscription and Recurring value is True. Current date is 13/04/2016	To check that FullSystemCheck functions	PeriodStartDate of subscription is set to the current date. PeriodEndDate is set to PeriodStartDate + length of subscription (6 months or 12 months). NextInvoice is set to PeriodStartDate + interval (1 month, 6 months or 12 months)	PeriodStartDate of subscription set to the current date. PeriodEndDate set to PeriodStartDate + length of subscription (6 months or 12 months). NextInvoice set to PeriodStartDate + interval (1 month, 6 months or 12 months)	No corrective action required	51
1-8	Run FullSystemCheck when an active subscription is present which is linked to an invoice with IStatus = 'Pending' and IssueDate + 7 days is after the current date. Current date is 13/04/2016	To check that FullSystemCheck functions	Subscription SStatus is set to 'Suspended' and invoice IStatus is set to 'Not Paid'	Subscription SStatus set to 'Suspended' and invoice IStatus set to 'Not Paid'	No corrective action required	51
1-9	Resize form	To check Resize event functions	pnlConnect Panel X coordinate changes proportionately with form width so as to keep the Panel horizontally centred	pnlConnect Panel X coordinate changed proportionately with form width so as to keep the Panel horizontally centred	No corrective action required	52
			M E M B E R S			
2-1	Load 'Members' form	To check form Load event is raised	DataGridView dgvSQLOutput displays any records retrieved from the Member table of the SQL database. Dimensions and location of form should be the same as the previous form	DataGridView dgvSQLOutput displayed records retrieved from the Member table of the SQL database. Dimensions and location of form were same as the previous form	No corrective action required	53
2-2	Click 'File' ToolStripDropDownButton	To check DropDown Items display	When clicked, the buttons 'Connect to database' and 'Print' should display	When clicked, the items Connect to database and Print displayed	No corrective action required	53
2-3	Click 'Connect to database' ToolStripMenuItem	To check Click event functions	Form changes to 'Start' form. Dimensions and location of form should be the same as the previous form	Form changed to 'Start' form. Dimensions and location of form were same as the previous form	No corrective action required	54
2-4	Click 'Print' ToolStripMenuItem	To check Click event functions	Print settings displays. Once user hits the 'Print' button in print settings, a PrintPreviewDialog	Print settings displayed. Once user hit the 'Print' button in print settings, a PrintPreviewDialog	No corrective action required	54

			shows, containing dgvSQLOutput.To finally print, the print icon is clicked	showed, containing dgvSQLOutput. Document printed when print icon was clicked		
2-5	Click 'Print' ToolStripMenuItem after doing a search query	To check Click event functions	Print settings displays. Once user hits the 'Print' button in print settings, a PrintPreviewDialog shows, containing dgvSQLOutput as well as the search field chosen and the query.To finally print, the print icon is clicked.	Print settings displayed. Once user hit the 'Print' button in print settings, a PrintPreviewDialog showed, containing dgvSQLOutput as well as the search field chosen and the query. Document printed when print icon was clicked	No corrective action required	55
2-6	Click 'Analytics' ToolStripMenuItem	To check Click event functions	Form changes to 'Analytics' form. Dimensions and location of form should be the same as the previous form	Form changed to 'Analytics' form. Dimensions and location of form were same as the previous form	No corrective action required	56
2-7	Click 'Members' ToolStripMenuItem	To check Click event functions	Form changes to 'Members' form. Dimensions and location of form should be the same as the previous form	Form changed to 'Members' form. Dimensions and location of form were same as the previous form	No corrective action required	57
2-8	Click 'Subscriptions' ToolStripMenuItem	To check Click event functions	Form changes to 'Subscriptions' form. Dimensions and location of form should be the same as the previous form	Form changed to 'Subscriptions' form. Dimensions and location of form were same as the previous form	No corrective action required	57
2-9	Click 'Invoices' ToolStripMenuItem	To check Click event functions	Form changes to 'Invoices' form. Dimensions and location of form should be the same as the previous form	Form changed to 'Invoices' form. Dimensions and location of form were same as the previous form	No corrective action required	58
2-10	Click 'Create Member' ToolStripButton	To check Click event functions	grpCreateMember Visible property is set to True, displaying the GroupBox to the user	grpCreateMember Visible property set to true, displaying the GroupBox to the user	No corrective action required	58
2-11	Click cboTitle ComboBox	To check ComboBox Items property contains the correct data	List of all available titles displays	List of all available titles displayed	No corrective action required	59
2-12	Click cboGender ComboBox	To check ComboBox Items property contains the correct data	Male or Female options display	Male or Female options displayed	No corrective action required	59
2-13	Click 'Create Member'	To check Click	MessageBox displays	MessageBox displayed	No corrective action	60

	button when all TextBoxes and ComboBoxes have been filled in. Date of birth chosen in DateTimePicker is also greater than 16 years old	event functions	stating that a member has been successfully created. User is then redirected back to 'Members' form and dgvSQLOutput updates with the new data	stating that a member has been successfully created. User was redirected back to 'Members' form and dgvSQLOutput updated with the new data	required	
2-14	Click 'Create Member' button when all TextBoxes and ComboBoxes have been filled in. Date of birth chosen in DateTimePicker is less than 16 years old	To check Click event functions	MessageBox displays stating that the minimum age requirement is 16 years old	MessageBox displayed stating that the minimum age requirement is 16 years old	No corrective action required	61
2-15	Click 'Create Member' button when all but one TextBox and ComboBox have been filled in. Date of birth chosen in DateTimePicker is greater than 16 years old	To check Click event functions	MessageBox displays stating that all fields must be filled in	MessageBox displayed stating that all fields must be filled in	No corrective action required	61
2-16	Click 'Update Member' ToolStripButton when a valid cell with a member record has been selected in dgvSQLOutput DataGridView	To check Click event functions	grpUpdateMember Visible property is set to True, displaying the groupbox to the user. The data corresponding to the member is filled into each appropriate TextBox, ComboBox and DateTimePicker	grpUpdateMember Visible property set to True, displaying the groupbox to the user. The data corresponding to the member filled into each appropriate TextBox,ComboBox and DateTimePicker	No corrective action required	62
2-17	Click 'Update Member' ToolStripButton when an invalid cell has been selected in dgvSQLOutput DataGridView	To check Click event functions	MessageBox displays stating that a valid cell/row must be selected	MessageBox displayed stating that a valid cell/row must be selected	No corrective action required	62
2-18	Click cboUpTitle ComboBox and cboGender ComboBox	To check ComboBox Items property contains the correct data	List of all available titles displays for cboUpTitle. 'Male' and 'Female' display for cboGender	List of all available titles displayed for cboUpTitle. 'Male' and 'Female' displayed for cboGender	No corrective action required	63
2-19	Click 'Delete Member' Button	To check Click event functions	MessageBox displays asking if user is sure they want to delete member. When user selects 'Yes', MessageBox displays stating that member and all references have been deleted	MessageBox displays asking if user is sure they want to delete member. When user selects 'Yes', MessageBox displays stating that an error was encountered	Upon investigating the DeleteMember() method in Member_DAL I found that there were a number of issues with the logic used. Most notable was that a query was executed to only find any active subscription, thus ignoring the possibility that the	64

					member could have had subscriptions before. Since ExecuteScalar() was used for the exection of the query's command, it also meant then that only 1 value was retrieved so I had to implement a SQLDataAdapter so that multiple values (i.e. SubscriptionID's) could be retrieved and filled into a DataTable. I then utilised a foreach loop on the rows of the data table to execute a delete query on invoices linked to each SubscriptionID retrieved. From here however, it was just a matter of make some minor adjustments to the rest of the code as the remaining delete queries used the MemberID which was already passed in	
2-20	Click 'Update Member' Button when all TextBoxes and ComboBoxes have been filled in. Date of birth chosen in DateTimePicker is also greater than 16 years old. Change surname to ensure update functions.	To check Click event functions	MessageBox displays stating that a member has been successfully updated. User is then redirected back to 'Members' form and dgvSQLOutput updates with the new data	MessageBox displayed stating that a member has been successfully updated. User was redirected back to 'Members' form and dgvSQLOutput updated with the new data. However the DateOfBirth value for the updated member was set to 01/01/1991 even though the chosen date was 08/03/1995	From reviewing the code relating to the updating of a member I found that in Member_DAL.cs , which contains the code for the execution of any SQL query relating to members, that within the UpdateMember method, the value for the date of birth was a string '01/01/1991' instead of the variable dateOfBirth which was passed into the argument.	66

					To rectify this I simply changed the string to dateOfBirth	
2-21	Click 'Update Member' button when all TextBoxes and ComboBoxes have been filled in. Date of birth chosen in DateTimePicker is less than 16 years old	To check Click event functions	MessageBox displays stating that the minimum age requirement is 16 years old	MessageBox displayed stating that the minimum age requirement is 16 years old	No corrective action required	68
2-22	Click 'Update Member' button when all but one TextBox has been filled in. Date of birth chosen in DateTimePicker is greater than 16 years old	To check Click event functions	MessageBox displays stating that all fields must be filled in	MessageBox displayed stating that all fields must be filled in	No corrective action required	68
2-23	Click 'View All Members' ToolStripButton	To check Click event functions	If either grpCreateMember or grpUpdateMember GroupBoxes are visible, their Visible property is set to False. dgvSQLOutput DataGridView is refreshed to show all members	When grpCreateMember was visible, its Visible property was set to False. dgvSQLOutput DataGridView is refreshed to show all members. The same applied when grpUpdateMember was visible	No corrective action required	69
2-24	Click cboSearchField ComboBox	To check ComboBox Items property contains the correct data	All fields relating to 'Member' SQL table display	All fields relating to 'Member' SQL table displayed	No corrective action required	69
2-25	Click 'Search' Button when no text has been entered into txtSearch TextBox	To check Click event functions	MessageBox displays stating that a value must be entered into the search TextBox	MessageBox displayed stating that a value must be entered into the search TextBox	No corrective action required	70
2-26	Click 'Search' Button when '1' has been entered into txtSearch TextBox and cboField text is MemberID	To check Click event functions	Data for the member with MemberID = 1 is displayed in dgvSQLOutput. lblSearchResults text sets to '1 result found'	Data for the member with MemberID = 1 was displayed in dgvSQLOutput. lblSearchResults text set to '1 result found'	No corrective action required	70
2-27	Resize form when dgvSQLOutput is Visible	To check Resize event functions	grpSearch GroupBox width resizes proportionately with form width. dgvSQLOutput width and height resizes proportionately with form width and height	grpSearch GroupBox width resized proportionately with form width. dgvSQLOutput width and height resized proportionately with form width and height	No corrective action required	71
2-28	Resize form when grpCreateMember or grpUpdateMember is Visible	To check Resize event functions	grpCreateMember / grpUpdateMember GroupBoxes width and height resizes proportionately with form width and height	grpCreateMember / grpUpdateMember GroupBoxes width and height resized proportionately with form width and height	No corrective action required	71

SUBSCRIPTIONS						
3-1	Load 'Subscriptions' form	To check form Load event is raised	DataGridView dgvSQLOutput displays any records retrieved from the Subscription table of the SQL database. Dimensions and location of form should be the same as the previous form	DataGridView dgvSQLOutput displayed records retrieved from the Subscription table of the SQL database. Dimensions and location of form were same as the previous form	No corrective action required	72
3-2	Click 'File' ToolStripDropDownButton	To check DropDown Items display	When clicked, the buttons 'Connect to database' and 'Print' should display	When clicked, the items Connect to database and Print displayed	No corrective action required	73
3-3	Click 'Connect to database' ToolStripMenuItem	To check Click event functions	Form changes to 'Start' form. Dimensions and location of form should be the same as the previous form	Form changed to 'Start' form. Dimensions and location of form were same as the previous form	No corrective action required	73
3-4	Click 'Print' ToolStripMenuItem while dgvSQLOutput is visible	To check Click event functions	Print settings displays. Once user hits the 'Print' button in print settings, a PrintPreviewDialog displays, containing dgvSQLOutput. To finally print, the print icon is clicked	Print settings displayed. Once user hit the 'Print' button in print settings, a PrintPreviewDialog displayed, containing dgvSQLOutput. Document printed when print icon was clicked	No corrective action required	74
3-5	Click 'Print' ToolStripMenuItem while dgvSQLOutput is visible after doing a search query	To check Click event functions	Print settings displays. Once user hits the 'Print' button in print settings, a PrintPreviewDialog displays, containing dgvSQLOutput as well as the search field chosen and the query. To finally print, the print icon is clicked.	Print settings displayed. Once user hit the 'Print' button in print settings, a PrintPreviewDialog displayed, containing dgvSQLOutput as well as the search field chosen and the query. Document printed when print icon was clicked	No corrective action required	7
3-6	Click 'Print' ToolStripMenuItem while dgvPlans is visible	To check Click event functions	Print settings displays. Once user hits the 'Print' button in print settings, a PrintPreviewDialog shows, containing dgvPlans. To finally print, the print icon is clicked	Print settings displayed. Once user hit the 'Print' button in print settings, a PrintPreviewDialog showed, containing dgvPlans. Document printed when print icon was clicked	No corrective action required	76
3-7	Click 'Analytics' ToolStripMenuItem	To check Click event functions	Form changes to 'Analytics' form. Dimensions and location of form should	Form changed to 'Analytics' form. Dimensions and location of form were	No corrective action required	77

			be the same as the previous form	same as the previous form		
3-8	Click 'Members' ToolStripMenuItem	To check Click event functions	Form changes to 'Members' form. Dimensions and location of form should be the same as the previous form	Form changed to 'Members' form. Dimensions and location of form were same as the previous form	No corrective action required	77
3-9	Click 'Subscriptions' ToolStripMenuItem	To check Click event functions	Form changes to 'Subscriptions' form. Dimensions and location of form should be the same as the previous form	Form changed to 'Subscriptions' form. Dimensions and location of form were same as the previous form	No corrective action required	78
3-10	Click 'Invoices' ToolStripMenuItem	To check Click event functions	Form changes to 'Invoices' form. Dimensions and location of form should be the same as the previous form	Form changed to 'Invoices' form. Dimensions and location of form were same as the previous form	No corrective action required	78
3-11	Click 'Plans' ToolStripButton	To check Click event functions	grpPlans GroupBox Visible property is set to True. grpCreateSubscription and grpUpdateSubscription GroupBoxes Visible property is set to False. DataGridView dgvPlans displays any records retrieved from the Plan table of the SQL database	grpPlans GroupBox Visible property set to True. grpCreateSubscription and grpUpdateSubscription GroupBoxes Visible property set to False. DataGridView dgvPlans displayed records retrieved from the Plan table of the SQL database	No corrective action required	79
3-12	Click cboPlans ComboBox	To check ComboBox Items property contains the correct data	All PlanName values from Plan table of SQL database display	All PlanName values from Plan table of SQL database displayed	No corrective action required	79
3-13	Click value from cboPlans DropDownList	To check TextChanged event functions	The price corresponding to the selected plan is set as the Text property of txtCurrentPrice	The price corresponding to the selected plan set as the Text property of txtCurrentPrice	No corrective action required	80
3-14	Click 'Save Changes' Button when no value have been entered in txtNewPrice TextBox or cboPlan ComboBox	To check Click event functions	MessageBox displays stating that a plan must be chosen and a valid new price entered	MessageBox displayed stating that a plan must be chosen and a valid new price entered	No corrective action required	80
3-15	Enter "1.1.1" then "30.123" afterwards into txtNewPrice TextBox	To check KeyPress event functions	User is unable to type either number into txtNewPrice	User was able to both "1.1.1" and "30.123" into txtNewPrice	The regular expression (regex) I was using to match the text in txtNewPrice appeared to not be functioning at all. To fix this I chose to	81

					use a different regex which proved to be functional when I tested it on a regex testing website. However when the regex did not match, a char was removed from the end of the txtNewPrice text rather than the current position of typing meaning that invalid values could be passed in. This was due to my use of retrieving a SubString then removing the last character so to rectify this I replaced it with `'e.Handled = e.KeyChar != (char)Keys.Back;'` which proved to be much more effective	
3-16	Click 'Save Changes' Button when a valid value has been entered in txtNewPrice TextBox and cboPlan ComboBox	To check Click event functions	MessageBox displays stating the selected plan and that it has been successfully updated	MessageBox displayed stating the selected plan and that it has been successfully updated	No corrective action required	82
3-17	Click 'Create Subscription' ToolStripButton	To check Click event functions	grpCreateSubscription GroupBox Visible property is set to True. grpUpdateSubscription and grpPlan GroupBoxes Visible property is set to False	grpCreateSubscription GroupBox Visible property set to True. grpUpdateSubscription and grpPlan GroupBoxes Visible property set to False	No corrective action required	83
3-18	Click 'Search' button (in grpCreateSubscription)	To check Click event functions	Form changes to 'Members' form. Dimensions and location of form should be the same as the previous form. 'Select Member' ToolStripButton is the only ToolStripButton present	Form changed to 'Members' form. Dimensions and location of form were be the same as the previous form. 'Select Member' ToolStripButton was the only ToolStripButton present	No corrective action required	83
3-19	Click 'Select Member' ToolStripButton when an invalid cell/row has been selected	To check Click event functions	MessageBox displays stating to ensure that a valid row/cell must be selected	Form changed to 'Subscriptions' form. Dimensions and location of form were the same as the previous form.	Inside the Click event for the 'Select Member' ToolStripButton there was no validation to	84

				grpCreateSubscription Visible property set to True. txtCreateMemberiD Text property set to -1	prevent the user from selecting an invalid cell/row. To fix this, I implemented an if statement to check whether the MemberID value retrieved is less than 0. When this occurs the MessageBox displays as originally intended	
3-20	Click 'Select Member' ToolStripButton when a valid cell/row has been selected	To check Click event functions	Form changes to 'Subscriptions' form. Dimensions and location of form should be the same as the previous form. grpCreateSubscription Visible property is set to True. txtCreateMemberiD Text property is set to the MemberID of the member record chosen in the previous 'Members' form	Form changed to 'Subscriptions' form. Dimensions and location of form were the same as the previous form. grpCreateSubscription Visible property set to True. txtCreateMemberiD Text property set to the MemberID of the member record chosen in the previous 'Members' form	No corrective action required	85
3-21	Click cboCreatePlans ComboBox	To check ComboBox Items property contains the correct data	All PlanName values from Plan table of SQL database display	All PlanName values from Plan table of SQL database displayed	No corrective action required	86
3-22	Click value from cboCreatePlans DropDownList	To check TextChanged event functions	The Amount value corresponding to the plan name selected displays in txtPlanPrice	The Amount value corresponding to the plan name selected displayed in txtPlanPrice	No corrective action required	87
3-23	Click cboCreateRecurring ComboBox	To check ComboBox Items property contains the correct data	The values 'True' and 'False' display	The values 'True' and 'False' displayed	No corrective action required	87
3-24	Click 'Create Subscription' Button when txtCreateMemberID, cboCreatePlans and cboCreateRecurring Text properties all contain valid values ('3', 'individual-monthly-twelve-month', 'True' respectively). Current date is 16/04/2016	To check Click event functions	MessageBox displays stating that a subscription has been successfully created. User is then redirected back to 'Subscriptions' form and dgvSQLOutput updates with the new data	MessageBox displays stating that a subscription has been successfully created. User is then redirected back to 'Subscriptions' form and dgvSQLOutput updates with the new data	No corrective action required	88
3-25	Click 'Create Subscription' Button when	To check Click event functions	MessageBox displays stating to ensure that	MessageBox displayed stating to ensure that	No corrective action required	89

	txtCreateMemberID, cboCreatePlans and cboCreateRecurring Text properties contain no values		all fields must be filled in	all fields must be filled in		
3-26	Click 'Create Subscription' Button when txtCreateMemberID, cboCreatePlans and cboCreateRecurring Text properties all contain valid values ('1', 'individual-monthly-twelve-month', 'True' respectively) but MemberID = 1 is already assigned an active subscription	To check Click event functions	MessageBox displays stating that the selected member already has a subscription and that it must be canceled to start a new one. User is then redirected back to 'Subscriptions' form	MessageBox displayed stating that the selected member already has a subscription and that it must be canceled to start a new one. User was then redirected back to 'Subscriptions' form	No corrective action required	89
3-27	Click 'Update Subscription' ToolStripButton when an invalid row/cell has been selected	To check Click event functions	MessageBox displays stating to ensure that a row/cell must be selected from the subscription	MessageBox displayed stating to ensure that a row/cell must be selected from the subscription	No corrective action required	90
3-28	Click 'Update Subscription' ToolStripButton when a valid row/cell has been selected	To check Click event functions	grpUpdateSubscription GroupBox Visible property is set to True. grpCreateSubscription Visible property is set to False. The data corresponding to the subscription is filled into each appropriate TextBox and ComboBox	grpUpdateSubscription GroupBox Visible property set to True. grpCreateSubscription Visible property set to False. The data corresponding to the subscription filled into each appropriate TextBox and ComboBox	No corrective action required	90
3-29	Click 'Update Subscription' ToolStripButton when 'Create Subscription' GroupBox or 'Update Subscription' GroupBox is Visible	To check Click event functions	MessageBox displays stating that no subscription is selected. User is then redirected back to 'Subscriptions' form	MessageBox displayed stating that no subscription is selected. User was then redirected back to 'Subscriptions' form. However an issue was encountered in which if the user attempted to use the 'Update Subscription' ToolStripButton it would continue returning the same error message regardless of whether or not a valid row/cell was selected	When either 'Create Subscription' or 'Update Subscription' GroupBoxes Visible property is set to true, a class variable Boolean called viewingData. This is used to determine whether the user is viewing the DataGridView as I could not find a way of setting the DataGridView selection to a null value. In this case, viewingData was not set to True when the either 'Create Subscription' or	91

					'Update Subscription' GroupBoxes Visible property was set to False. Fixing this was a minor task that only involved entering a single line of code	
3-30	Click 'Recurring' ComboBox (in grpUpdateSubscription)	To check ComboBox Items property contains the correct data	The values 'True' and 'False' display	The values 'True' and 'False' displayed	No corrective action required	92
3-31	Select the other value in 'Recurring' ComboBox then select original value	To check TextChanged event functions	When the other value is selected (True when the original value is False and vice versa), 'Update Subscription' Button to Enabled is set to True. When the original value is selected, 'Update Subscription' Enabled property is set to False	When the other value was selected (True when the original value is False and vice versa), 'Update Subscription' Button to Enabled property set to True. When the original value was selected, 'Update Subscription' Enabled property set to False	No corrective action required	93
3-32	Change 'Recurring' ComboBox value then click 'Update Subscription' Button	To check Click event functions	MessageBox displays stating that the subscription has been successfully updated. User is then redirected back to 'Subscriptions' form and dgvSQLOutput updates with the new data	MessageBox displayed stating that the subscription has been successfully updated. User was then redirected back to 'Subscriptions' form and dgvSQLOutput updated with the new data	No corrective action required	93
3-33	Click 'Cancel Subscription' Button when an invoice record with IStatus = 'Pending' is linked to the subscription	To check Click event functions	MessageBox displays asking if user is sure they want to cancel the subscription. If user selects 'No', confirmation MessageBox displays and they are redirected back to 'Subscriptions' form. If user selects 'Yes', a MessageBox displays stating that the subscription has been successfully canceled and the pending invoice canceled. User is then redirected to 'Subscriptions' form and dgvSQLOutput	MessageBox displayed asking if user is sure they want to cancel the subscription. When user selected 'No', confirmation MessageBox displayed and they were redirected back to 'Subscriptions' form. When user selected 'Yes', a MessageBox displayed stating that the subscription has been successfully canceled and the pending invoice canceled. User was then redirected to 'Subscriptions' form	No corrective action required	94

			updates with the new data	and dgvSQLOutput updated with the new data		
3-34	Click 'Cancel Subscription' Button when there is no invoice record with IStatus = 'Pending' linked to the subscription	To check Click event functions	MessageBox displays asking if user is sure they want to cancel the subscription. If user selects 'No', confirmation MessageBox displays and they are redirected back to 'Subscriptions' form. If user selects 'Yes', a MessageBox displays stating that the subscription has been successfully canceled. User is then redirected to 'Subscriptions' form and dgvSQLOutput updates with the new data	MessageBox displayed asking if user is sure they want to cancel the subscription. When user selected 'No', confirmation MessageBox displayed and they were redirected back to 'Subscriptions' form. When user selected 'Yes', a MessageBox displayed stating that the subscription has been successfully canceled and pending invoice canceled as well. User was then redirected to 'Subscriptions' form and dgvSQLOutput updated with the new data	Within the Subscription_DAL, the SQL query I used to cancel pending invoices was `"UPDATE Invoice SET IStatus = 'Canceled' WHERE SubscriptionID = " + subID + " ;".` This meant that all invoices related to the specified subID were having their IStatus set to 'Canceled' rather than just the the one with 'Pending' IStatus. In order to fix this I added " `AND IStatus = 'Pending';"` to the end of the query.	97
3-35	Click 'View All Subscriptions' ToolStripButton	To check Click event functions	If grpCreateSubscription , grpUpdateSubscription, grpPlans GroupBoxes are visible, their Visible property is set to False. dgvSQLOutput DataGridView is refreshed to show all subscriptions	When grpCreateSubscription was visible, its Visible property was set to False. dgvSQLOutput did not refresh to show all subscriptions. The same applied when grpUpdateSubscription or grpPlans was visible	A line of code for reseting the data source of dgvSQLOutput was missing. Once added, the problem was fixed.	100
3-36	Click cboSearchField ComboBox	To check ComboBox Items property contains the correct data	All fields relating to 'Subscription' SQL table display	All fields relating to 'Subscription' SQL table displayed	No corrective action required	101
3-37	Click 'Search' Button when no text has been entered into txtSearch TextBox	To check Click event functions	MessageBox displays stating that a value must be entered into the search TextBox	MessageBox displayed stating that a value must be entered into the search TextBox	No corrective action required	101
3-38	Click 'Search' Button when '1' has been entered into txtSearch TextBox and cboField text is 'SubscriptionID'	To check Click event functions	Data for the subscription with SubscriptionID = 1 is displayed in dgvSQLOutput. lblSearchResults text sets to '1 result found'	Data for the subscription with SubscriptionID = 1 was displayed in dgvSQLOutput. lblSearchResults text set to '1 result found'	No corrective action required	102
3-39	Resize form when dgvSQLOutput is Visible	To check Resize event functions	grpSearch GroupBox width resizes	grpSearch GroupBox width resized	No corrective action required	102

			proportionately with form width. dgvSQLOutput width and height resizes proportionately with form width and height	proportionately with form width. dgvSQLOutput width and height resized proportionately with form width and height		
3-40	Resize form when grpPlans GroupBox is Visible	To check Resize event functions	grpPlans GroupBox width and height resizes proportionately with form width and height. dgvPlans DataGridView width and height resizes proportionately with form width and height. grpChangePrice GroupBox width resizes proportionately with form width and Y coordinate also changes proportionately to ensure that grpChangePrice never overlaps with dgvPlans	grpPlans GroupBox width and height resized proportionately with form width and height. dgvPlans DataGridView width and height resized proportionately with form width and height. grpChangePrice GroupBox width resized proportionately with form width and Y coordinate also changed proportionately so that grpChangePrice never overlaped with dgvPlans	No corrective action required	103
3-41	Resize form when grpCreateSubscription or grpUpdateSubscription is Visible	To check Resize event functions	grpCreateSubscription / grpUpdateSubscription GroupBoxes width and height resizes proportionately with form width and height	grpCreateSubscription / grpUpdateSubscription GroupBoxes width and height resized proportionately with form width and height	No corrective action required	103
			I N V O I C E			
4-1	Load 'Invoices' form	To check form Load event is raised	DataGridView dgvSQLOutput displays any records retrieved from the Invoice table of the SQL database. Dimensions and location of form should be the same as the previous form	DataGridView dgvSQLOutput displayed records retrieved from the Invoice table of the SQL database. Dimensions and location of form were same as the previous form	No corrective action required	105
4-2	Click 'File' ToolStripDropDownButton	To check DropDown Items display	When clicked, the buttons 'Connect to database' and 'Print' should display	When clicked, the items Connect to database and Print displayed	No corrective action required	105
4-3	Click 'Connect to database' ToolStripMenuItem	To check Click event functions	Form changes to 'Start' form. Dimensions and location of form should be the same as the previous form	No changes	No click event had been created for the ToolStripMenuItem so I simply created a new click event and utilised the same code used in each	106

					other form	
4-4	Click 'Print' ToolStripMenuItem	To check Click event functions	Print settings displays. Once user hits the 'Print' button in print settings, a PrintPreviewDialog displays, containing dgvSQLOutput. To finally print, the print icon is clicked	Print settings displayed. Once user hit the 'Print' button in print settings, a PrintPreviewDialog displayed, containing dgvSQLOutput. Document printed when print icon was clicked	No corrective action required	106
4-5	Click 'Print' ToolStripMenuItem after doing a search query	To check Click event functions	Print settings displays. Once user hits the 'Print' button in print settings, a PrintPreviewDialog displays, containing dgvSQLOutput as well as the search field chosen and the query. To finally print, the print icon is clicked.	Print settings displayed. Once user hit the 'Print' button in print settings, a PrintPreviewDialog displayed, containing dgvSQLOutput as well as the search field chosen and the query. Document printed when print icon was clicked	No corrective action required	107
4-6	Click 'Analytics' ToolStripMenuItem	To check Click event functions	Form changes to 'Analytics' form. Dimensions and location of form should be the same as the previous form	Form changed to 'Analytics' form. Dimensions and location of form were same as the previous form	No corrective action required	108
4-7	Click 'Members' ToolStripMenuItem	To check Click event functions	Form changes to 'Members' form. Dimensions and location of form should be the same as the previous form	Form changed to 'Members' form. Dimensions and location of form were same as the previous form	No corrective action required	109
4-8	Click 'Subscriptions' ToolStripMenuItem	To check Click event functions	Form changes to 'Subscriptions' form. Dimensions and location of form should be the same as the previous form	Form changed to 'Subscriptions' form. Dimensions and location of form were same as the previous form	No corrective action required	109
4-9	Click 'Invoices' ToolStripMenuItem	To check Click event functions	Form changes to 'Invoices' form. Dimensions and location of form should be the same as the previous form	Form changed to 'Invoices' form. Dimensions and location of form were same as the previous form	No corrective action required	110
4-10	Click 'All Invoices' ToolStripButton	To check Click event functions	dgvSQLOutput DataGridView displays all invoices. lblSearchResults displays the number of matching records	dgvSQLOutput DataGridView displays all invoices. lblSearchResults displayed the number of matching records	No corrective action required	110
4-11	Click 'Paid' ToolStripButton	To check Click event functions	dgvSQLOutput DataGridView displays all invoices with IStatus	dgvSQLOutput DataGridView displayed all invoices	No corrective action required	111

			= 'Paid'. lblSearchResults displays the number of matching records. If grpUpdateInvoice GroupBox Visible property is True, it is set to False	with IStatus = 'Paid'. lblSearchResults displayed the number of matching records. When grpUpdateInvoice GroupBox Visible property was True, it was set to False		
4-12	Click 'Pending' ToolStripButton	To check Click event functions	dgvSQLOutput DataGridView displays all invoices with IStatus = 'Pending'. lblSearchResults displays the number of matching records	dgvSQLOutput DataGridView displayed all invoices with IStatus = 'Pending'. lblSearchResults displayed the number of matching records	No corrective action required	111
4-13	Click 'Not Paid' ToolStripButton	To check Click event functions	dgvSQLOutput DataGridView displays all invoices with IStatus = 'Not Paid'. lblSearchResults displays the number of matching records	dgvSQLOutput DataGridView displayed all invoices with IStatus = 'Not Paid'. lblSearchResults displayed the number of matching records	No corrective action required	112
4-14	Click 'Canceled' ToolStripButton	To check Click event functions	dgvSQLOutput DataGridView displays all invoices with IStatus = 'Canceled'. lblSearchResults displays the number of matching records	dgvSQLOutput DataGridView displayed all invoices with IStatus = 'Canceled'. lblSearchResults displayed the number of matching records	No corrective action required	112
4-15	Click 'Paid' ToolStripButton when grpUpdateInvoice GroupBox is visible	To check Click event functions	dgvSQLOutput DataGridView displays all invoices with IStatus = 'Paid'. lblSearchResults displays the number of matching records. grpUpdateInvoice GroupBox Visible property is set to False	dgvSQLOutput DataGridView displayed all invoices with IStatus = 'Paid'. lblSearchResults displayed the number of matching records. grpUpdateInvoice GroupBox Visible property was set to False	No corrective action required	113
4-16	Click 'Pending' ToolStripButton when grpUpdateInvoice GroupBox is visible	To check Click event functions	dgvSQLOutput DataGridView displays all invoices with IStatus = 'Pending'. lblSearchResults displays the number of matching records. grpUpdateInvoice GroupBox Visible property is set to False	dgvSQLOutput DataGridView displayed all invoices with IStatus = 'Pending'. lblSearchResults displayed the number of matching records. grpUpdateInvoice GroupBox Visible property was set to False	No corrective action required	114
4-17	Click 'Not Paid'	To check Click	dgvSQLOutput	dgvSQLOutput	No corrective action	114

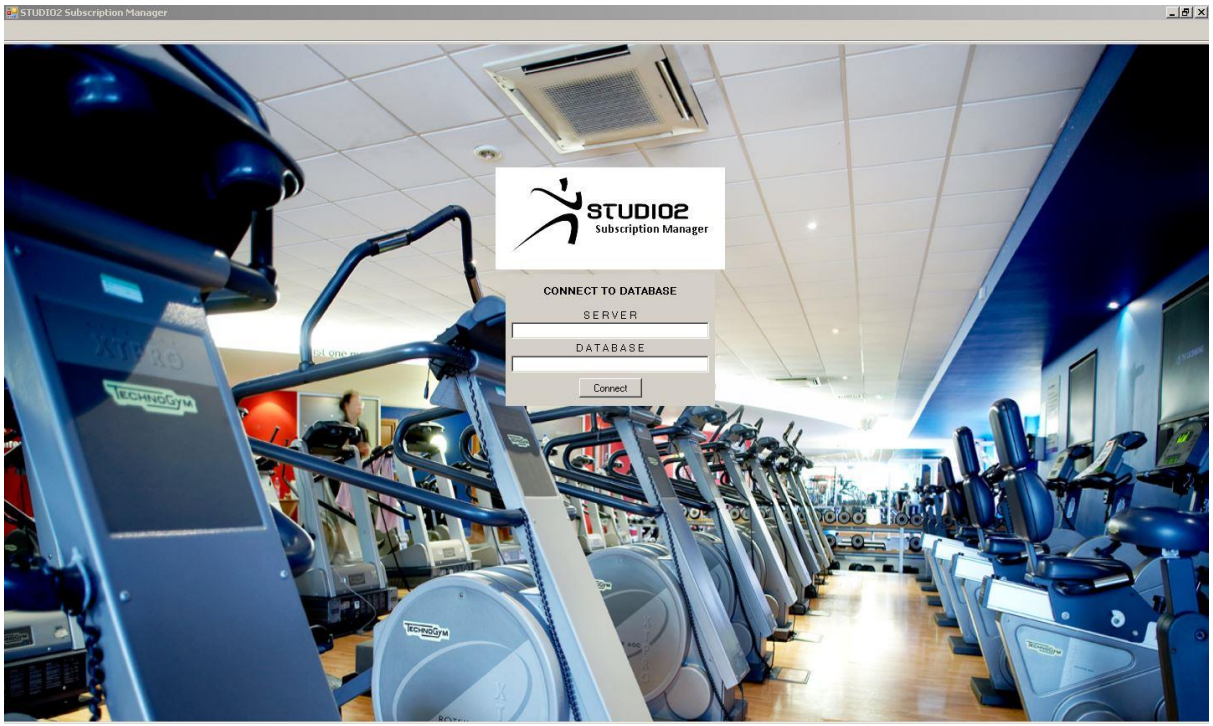
	ToolStripButton when grpUpdateInvoice GroupBox is visible	event functions	DataGridView displays all invoices with IStatus = 'Not Paid'. lblSearchResults displays the number of matching records. grpUpdateInvoice GroupBox Visible property is set to False	DataGridView displayed all invoices with IStatus = 'Not Paid'. lblSearchResults displayed the number of matching records. grpUpdateInvoice GroupBox Visible property was set to False	required	
4-18	Click 'Canceled' ToolStripButton when grpUpdateInvoice GroupBox is visible	To check Click event functions	dgvSQLOutput DataGridView displays all invoices with IStatus = 'Canceled'. lblSearchResults displays the number of matching records. grpUpdateInvoice GroupBox Visible property is set to False	dgvSQLOutput DataGridView displayed all invoices with IStatus = 'Canceled'. lblSearchResults displayed the number of matching records. grpUpdateInvoice GroupBox Visible property was set to False	No corrective action required	115
4-19	Click 'Update Invoice' ToolStripButton when an valid cell/row has been selected	To check Click event functions	MessageBox displays stating that a cell/row of the chosen invoice must be selected	MessageBox displayed stating that a cell/row of the chosen invoice must be selected	No corrective action required	115
4-20	Click 'Update Invoice' ToolStripButton when a valid cell/row has been selected	To check Click event functions	grpUpdateInvoice Visible property is set to True, displaying the GroupBox to the user. The data corresponding to the invoice is filled into each appropriate TextBox	grpUpdateInvoice Visible property set to True, displaying the GroupBox to the user. The data corresponding to the invoice filled into each appropriate TextBox	No corrective action required	116
4-21	Click cboIStatus ComboBox	To check ComboBox Items property contains the correct data	'Paid', 'Pending', 'Not Paid' and 'Canceled' display	'Paid', 'Pending', 'Not Paid' and 'Canceled' displayed	No corrective action required	117
4-22	Click 'Update Invoice' button when cboIStatus Text has not been changed to a different value	To check Click event functions	MessageBox displays stating that no changes have been made. User is then redirected back to 'Invoices'	MessageBox displayed stating that no changes have been made. User was then redirected back to 'Invoices'	No corrective action required	118
4-23	Click 'Update Invoice' button when cboIStatus Text has been changed to a different value	To check Click event functions	MessageBox displays stating that the invoice has been updated. User is then redirected back to 'Invoices' form and dgvSQLOutput updates with the new data	MessageBox displayed stating that the invoice has been updated. User was then redirected back to 'Invoices' form and dgvSQLOutput updated with the new data	No corrective action required	118
4-24	Click cboSearchField	To check	All fields relating to	All fields relating to	No corrective action	119

	ComboBox	ComboBox Items property contains the correct data	'Invoice' SQL table display	'Invoice' SQL table displayed	required	
4-25	Click 'Search' Button when no text has been entered into txtSearch TextBox	To check Click event functions	MessageBox displays stating that a value must be entered into the search TextBox	MessageBox displayed stating that a value must be entered into the search TextBox	No corrective action required	120
4-26	Click 'Search' Button when '1' has been entered into txtSearch TextBox and cboField text is 'SubscriptionID'	To check Click event functions	Data for the subscription with InvoiceID = 1 is displayed in dgvSQLOutput. lblSearchResults text sets to '1 result found'	Data for the subscription with Invoic ID = 1 was displayed in dgvSQLOutput. lblSearchResults text set to '1 result found'	No corrective action required	120
4-27	Resize form when dgvSQLOutput is Visible	To check Resize event functions	grpSearch GroupBox width resizes proportionately with form width. dgvSQLOutput width and height resizes proportionately with form width and height	grpSearch GroupBox width resized proportionately with form width. dgvSQLOutput width and height resized proportionately with form width and height	No corrective action required	121
4-28	Resize form when grpUpdateInvoice is Visible	To check Resize event functions	grpUpdateInvoice GroupBox width and height resizes proportionately with form width and height	grpUpdateInvoice GroupBox width resized proportionately with form width and height	No corrective action required	121
			A N A L Y T I C S			
5-1	Load 'Analytics' form	To check form Load event is raised	ReportView member report displays. Dimensions and location of form should be the same as the previous form. GroupBoxes display data related to various tables of SQL database too.	ReportView member report displays. Dimensions and location of form were same as the previous form. GroupBoxes displayed data related to various tables of SQL database too.	No corrective action required	122
5-2	Click 'File' ToolStripDropDownButton	To check DropDown Items display	When clicked, the buttons 'Connect to database' and 'Print' should display	When clicked, the items Connect to database and Print displayed	No corrective action required	122
5-3	Click 'Connect to database' ToolStripMenuItem	To check Click event functions	Form changes to 'Start' form. Dimensions and location of form should be the same as the previous form	Form changed to 'Start' form. Dimensions and location of form were same as the previous form	No corrective action required	123
5-4	Click 'Analytics' ToolStripMenuItem	To check Click event functions	Form changes to 'Analytics' form. Dimensions and location of form should be the same as the previous form	Form changed to 'Analytics' form. Dimensions and location of form were same as the previous form	No corrective action required	123
5-5	Click 'Members'	To check Click	Form changes to	Form changed to	No corrective action	124

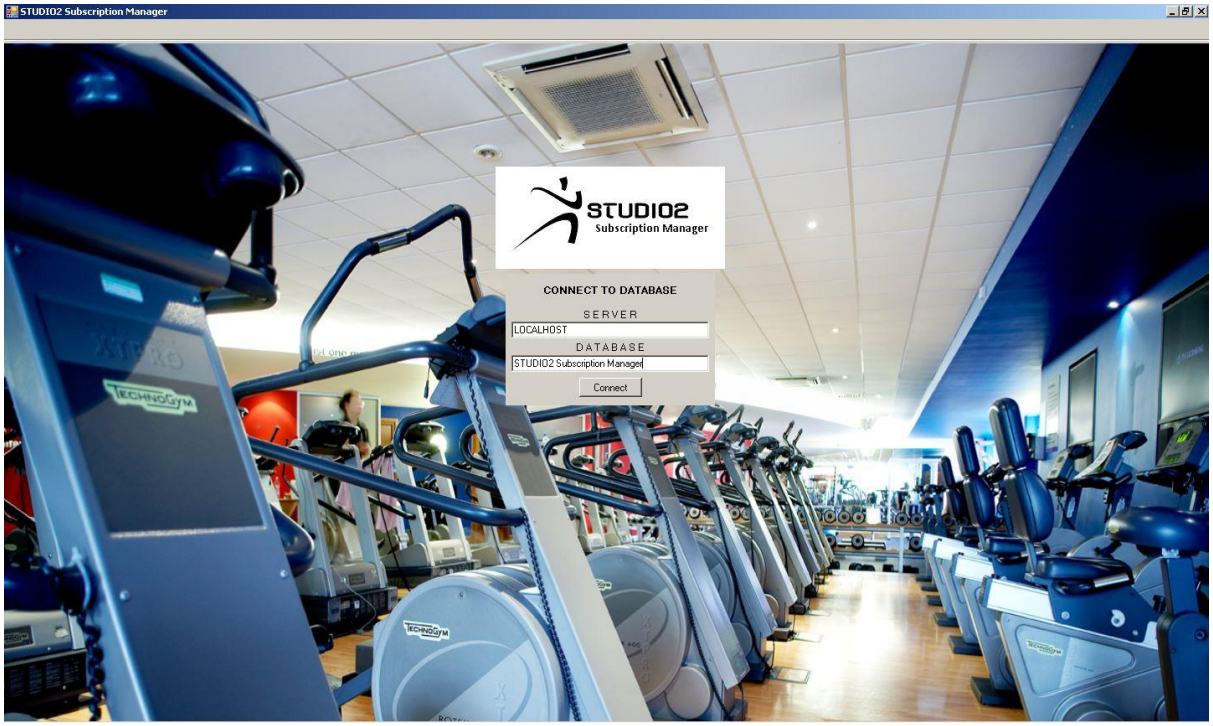
	ToolStripMenuItem	event functions	'Members' form. Dimensions and location of form should be the same as the previous form	'Members' form. Dimensions and location of form were same as the previous form	required	
5-6	Click 'Subscriptions' ToolStripMenuItem	To check Click event functions	Form changes to 'Subscriptions' form. Dimensions and location of form should be the same as the previous form	Form changed to 'Subscriptions' form. Dimensions and location of form were same as the previous form	No corrective action required	124
5-7	Click 'Invoices' ToolStripMenuItem	To check Click event functions	Form changes to 'Invoices' form. Dimensions and location of form should be the same as the previous form	Form changed to 'Invoices' form. Dimensions and location of form were same as the previous form	No corrective action required	125
5-8	Select "ReportSubscriptions.rdlc" in ToolStripMenuItem ComboBox	To check TextChanged event functions	reportViewerMember Visible property is set to false. reportViewer Subscription Visible property is set to true.	reportViewerMember Visible property set to false. reportViewer Subscription Visible property set to true.	No corrective action required	125
5-9	Select "ReportMembers.rdlc" in ToolStripMenuItem ComboBox	To check TextChanged event functions	reportViewerMember Visible property is set to true. reportViewer Subscription Visible property is set to false.	reportViewerMember Visible property set to true. reportViewer Subscription Visible property set to false.	No corrective action required	126
5-10	Create new member in Member Form then move to Analytics Form to check that Member Report Viewer updates with new data	To check Report Viewer updates	Member Report Viewer updates to incorporate new member	Member Report Viewer updated to incorporate new member	No corrective action required	126
5-11	Resize form	To check Resize event functions	Both ReportViewers width and height resizes proportionately with form width and height	Both ReportViewer width and height resized proportionately with form width and height	No corrective action required	128

Testing Evidence

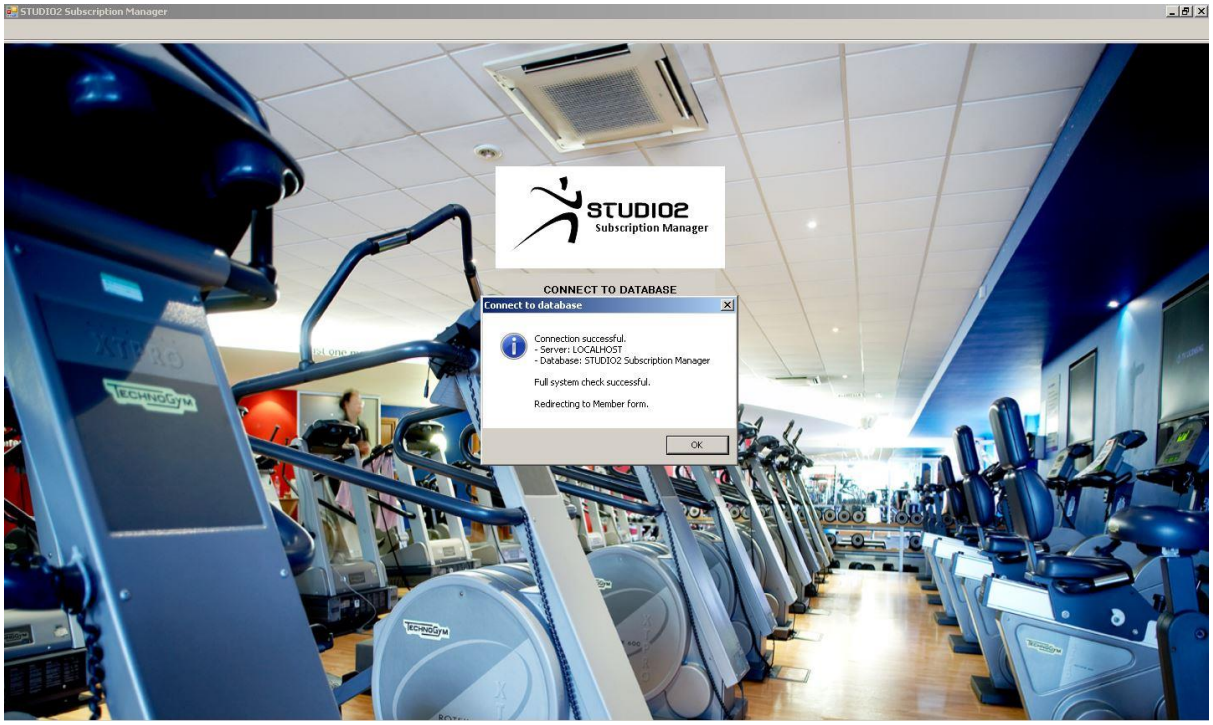
START FORM



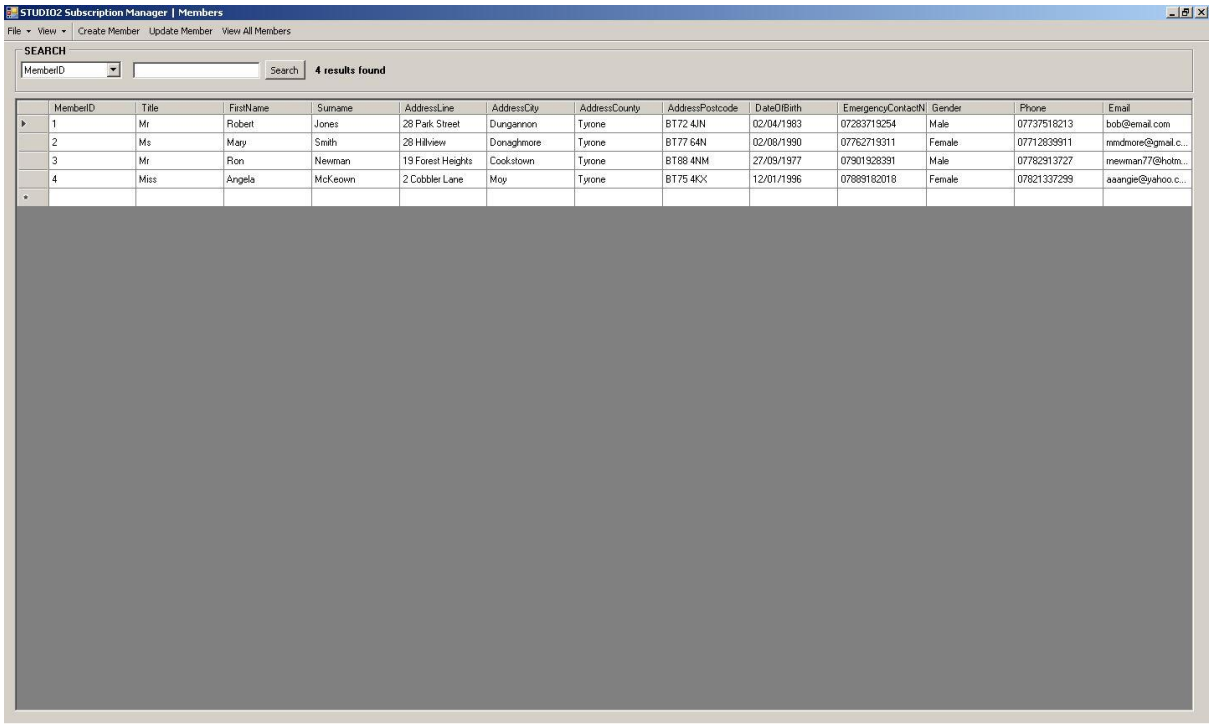
Test 1-1: Load Start form



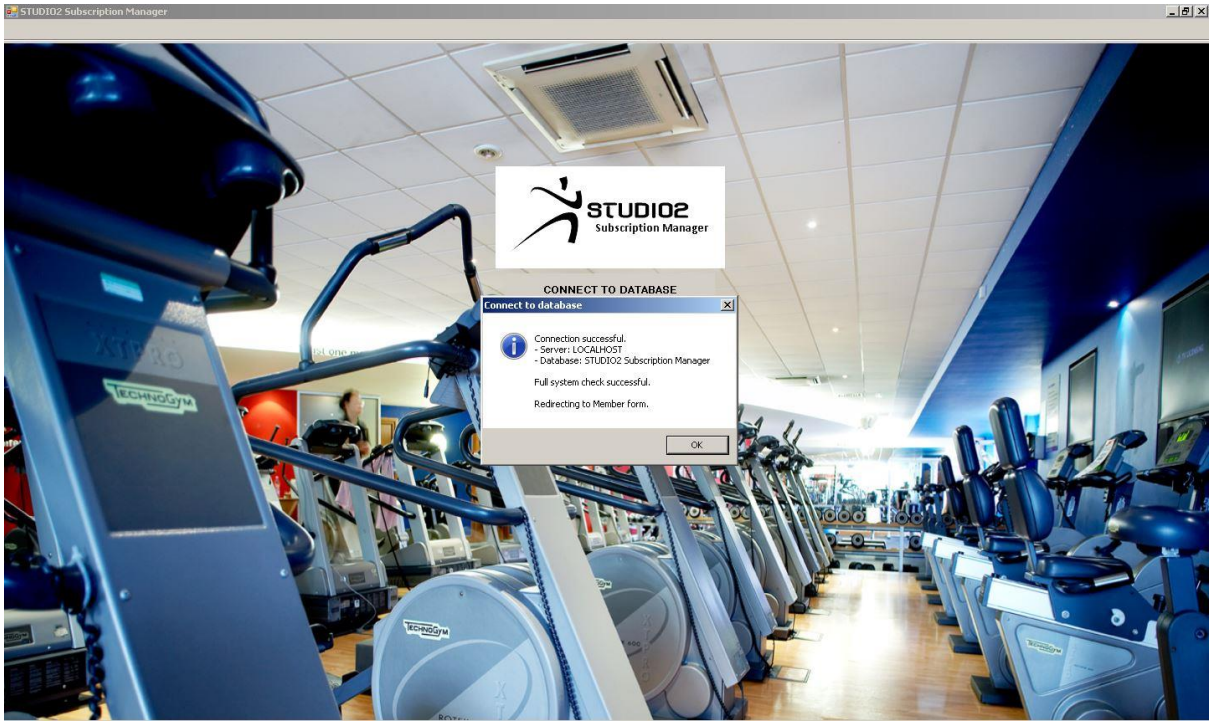
Test 1-2: Enter server and database details (LOCALHOST and STUDIO2 Subscription Manager SQL) then click 'Connect' Button



Test 1-2: 'Connect' Button clicked



Test 1-2: Members Form displays



Test 1-3: Verify that the FullSystemCheck method is functioning properly by connecting (ENSURE

	SubscriptionID	MemberID	PlanID	Recurring	SubscriptionStartDate	PeriodStartDate	PeriodEndDate	CanceledDate	NextInvoice	SStatus
1	1	1	1	0	2016-04-13	2016-03-20	2017-04-20	NULL	2016-04-20	Active

TESTS ARE CORRECT)

Test 1-4: Subscription record

	InvoiceID	IssueDate	SubscriptionID	IStatus
1	1	2016-03-20	1	Paid

Test 1-4: Invoice record

	InvoiceID	IssueDate	SubscriptionID	IStatus
1	1	2016-03-20	1	Paid
2	2	2016-04-13	1	Pending

Test 1-4: FullSystemCheck performed. New invoice created

	SubscriptionID	MemberID	PlanID	Recurring	SubscriptionStartDate	PeriodStartDate	PeriodEndDate	CanceledDate	NextInvoice	SStatus
1	1	1	1	0	2016-03-20	2016-03-20	2017-03-20	NULL	2016-05-20	Active

Test 1-4: In Subscription record, NextInvoice field updated

	SubscriptionID	MemberID	PlanID	Recurring	SubscriptionStartDate	PeriodStartDate	PeriodEndDate	CanceledDate	NextInvoice	SStatus
1	1	1	1	0	2015-04-20	2015-04-20	2016-04-20	NULL	2016-04-20	Active

Test 1-5: Subscription record

	InvoiceID	IssueDate	SubscriptionID	IStatus
1	1	2016-03-13	1	Paid

Test 1-5: Nothing happens

	SubscriptionID	MemberID	PlanID	Recurring	SubscriptionStartDate	PeriodStartDate	PeriodEndDate	CanceledDate	NextInvoice	SStatus
1	1	1	1	0	2015-04-13	2015-04-13	2016-04-13	NULL	2016-04-13	Active

Test 1-6: Subscription record

	SubscriptionID	MemberID	PlanID	Recurring	SubscriptionStartDate	PeriodStartDate	PeriodEndDate	CanceledDate	NextInvoice	SStatus
1	1	1	1	0	2015-04-13	2015-04-13	2016-04-13	NULL	2016-04-13	Expired

Test 1-6: Subscription record after FullSystemCheck

	SubscriptionID	MemberID	PlanID	Recurring	SubscriptionStartDate	PeriodStartDate	PeriodEndDate	CanceledDate	NextInvoice	SStatus
1	1	1	1	1	2015-04-13	2015-04-13	2016-04-13	NULL	2016-04-13	Active

Test 1-7: Subscription record

	SubscriptionID	MemberID	PlanID	Recurring	SubscriptionStartDate	PeriodStartDate	PeriodEndDate	CanceledDate	NextInvoice	SStatus
1	1	1	1	1	2015-04-13	2016-04-13	2017-04-13	NULL	2016-05-13	Active

Test 1-7: Subscription record after FullSystemCheck

	InvoiceID	IssueDate	SubscriptionID	IStatus
1	1	2016-03-13	1	Paid
2	3	2016-04-13	1	Pending

Test 1-7: Pending invoice generated for new subscription cycle (InvoiceID = 3)

	SubscriptionID	MemberID	PlanID	Recurring	SubscriptionStartDate	PeriodStartDate	PeriodEndDate	CanceledDate	NextInvoice	SStatus
1	1	1	1	1	2015-05-12	2015-05-13	2016-05-12	NULL	2016-04-12	Active

Test 1-8: Subscription record

	InvoiceID	IssueDate	SubscriptionID	IStatus
1	1	2016-04-05	1	Pending

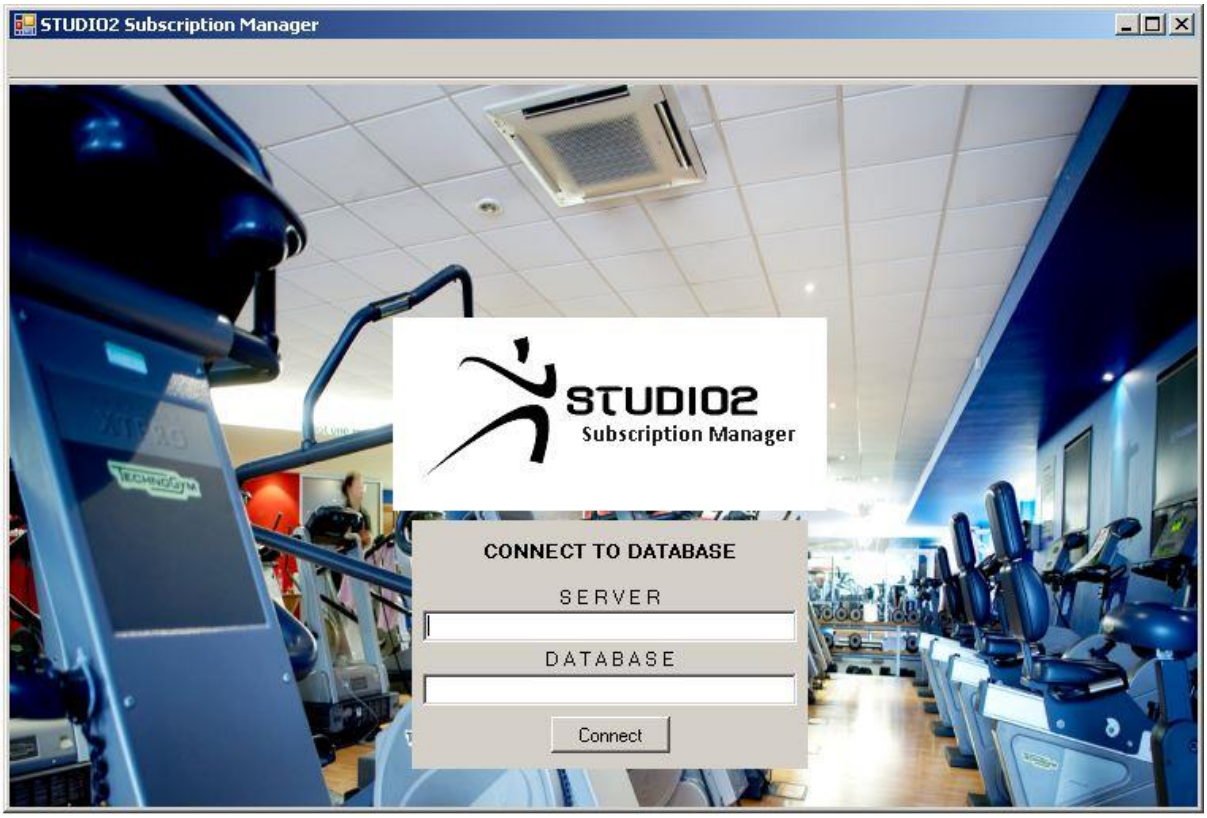
Test 1-8: Invoice record

	SubscriptionID	MemberID	PlanID	Recurring	SubscriptionStartDate	PeriodStartDate	PeriodEndDate	CanceledDate	NextInvoice	SStatus
1	1	1	1	1	2015-05-12	2015-05-13	2016-05-12	NULL	2016-04-12	Suspended

Test 1-8: Subscription record after FullSystemCheck

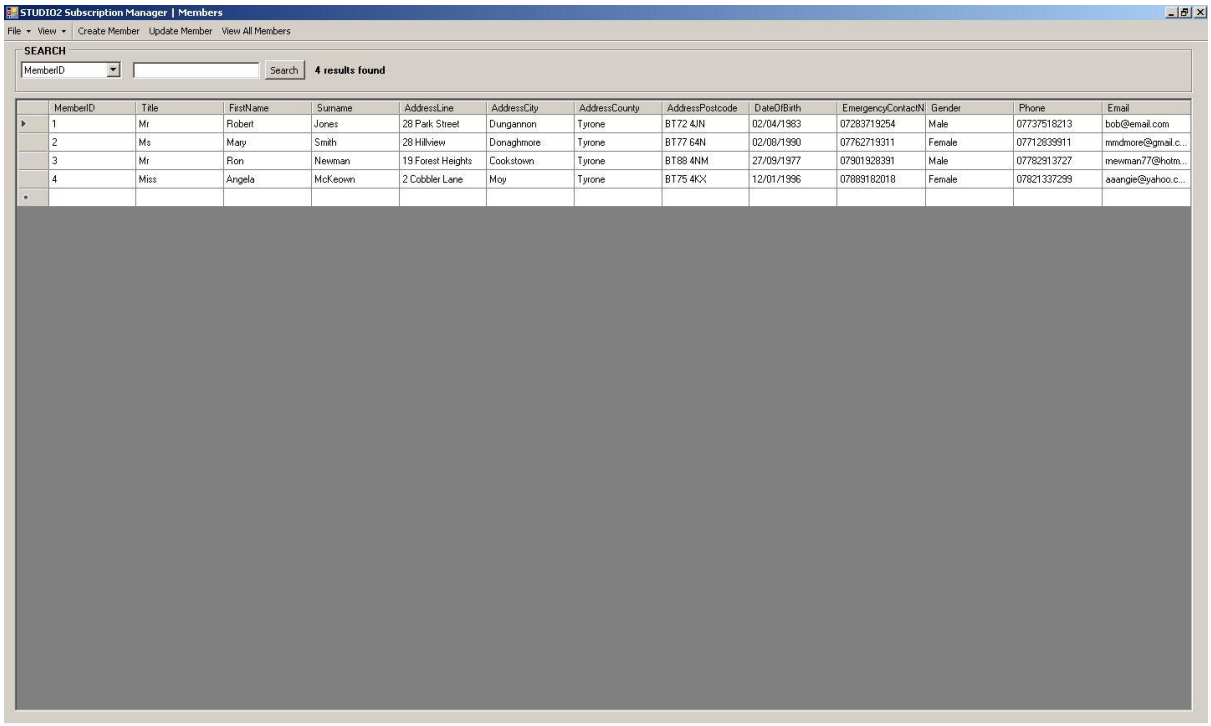
	InvoiceID	IssueDate	SubscriptionID	IStatus
1	1	2016-04-05	1	Not Paid

Test 1-8: Invoice record after FullSystemCheck

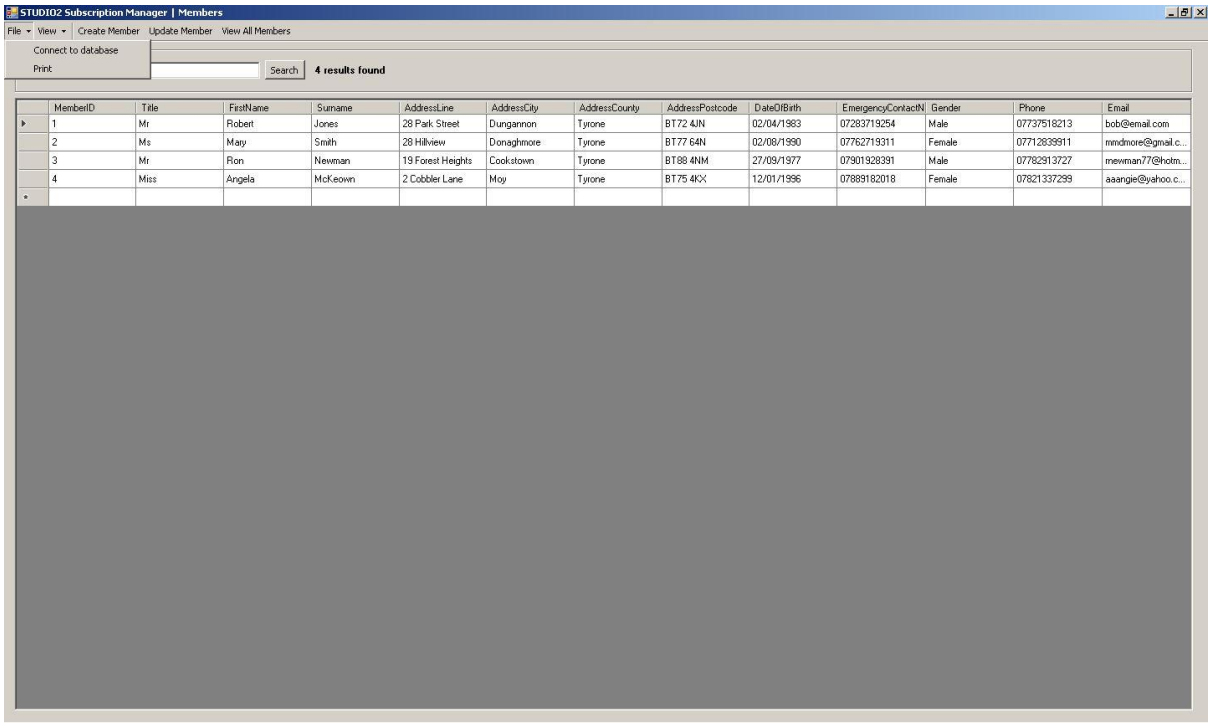


Test 1-9: Resize Form

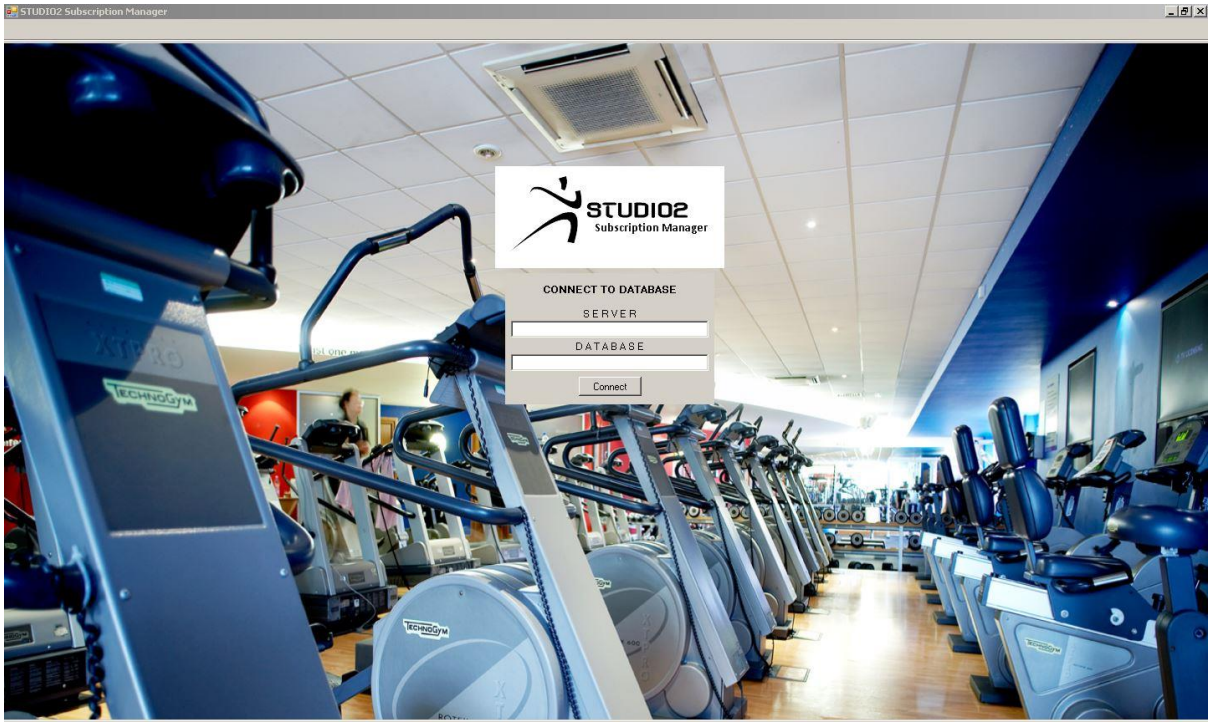
MEMBERS FORM



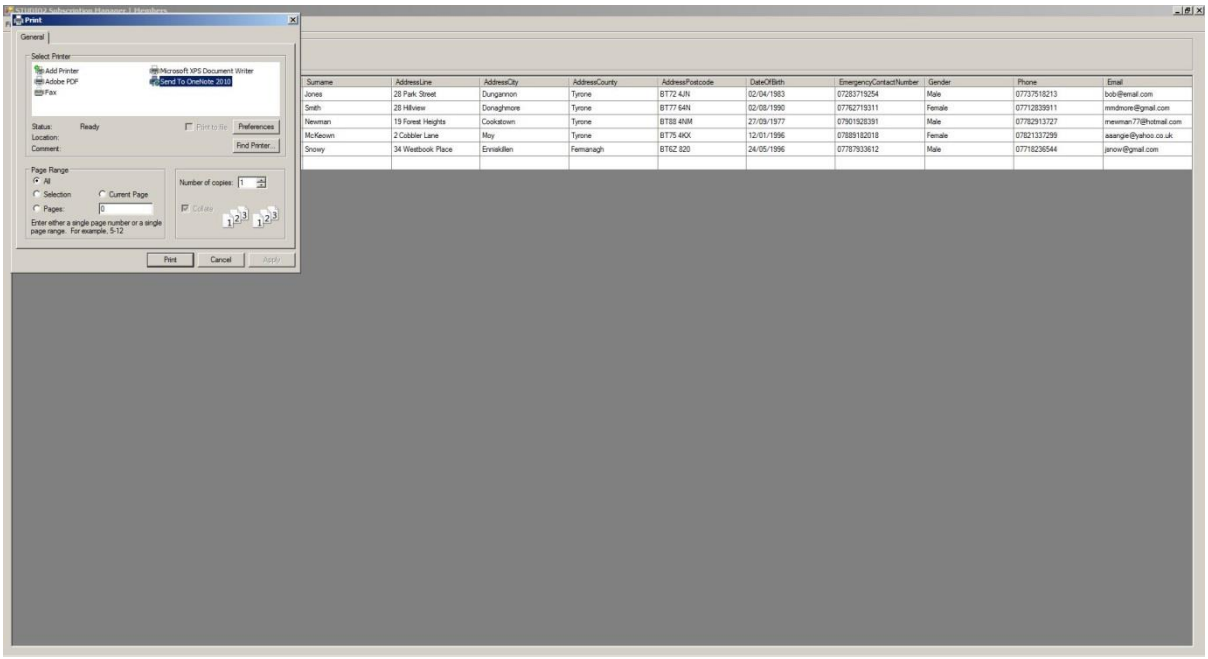
Test 2-1: Load Members Form



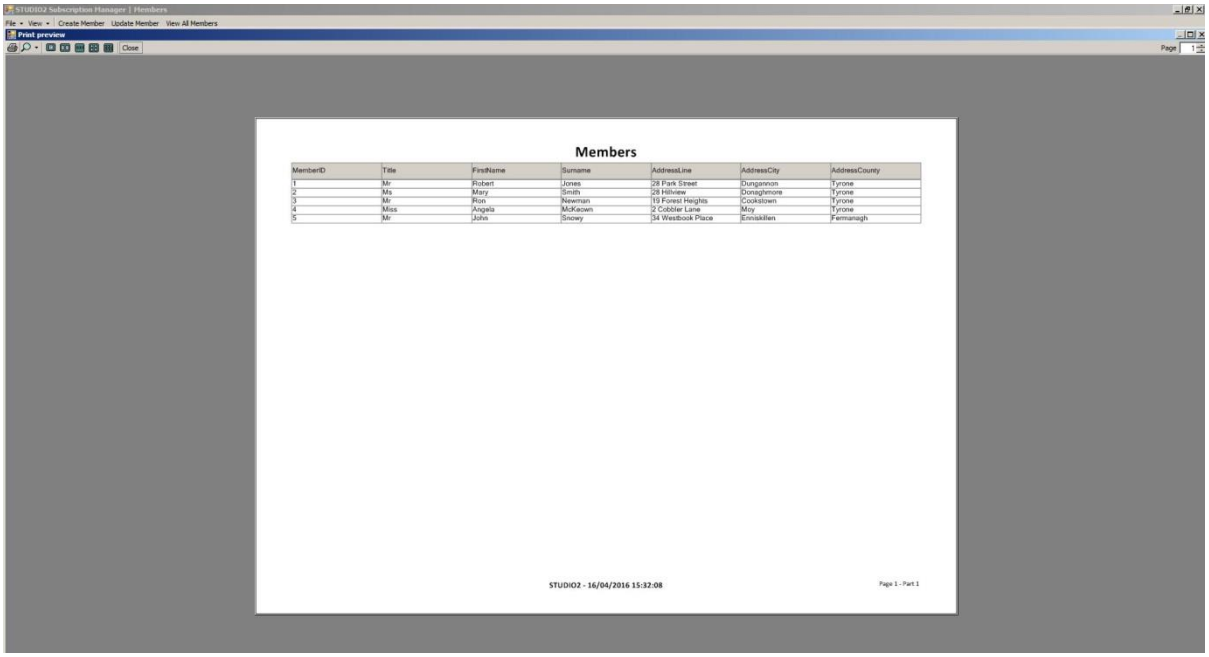
Test 2-1: Click 'Connect to Database' ToolStripMenuItem



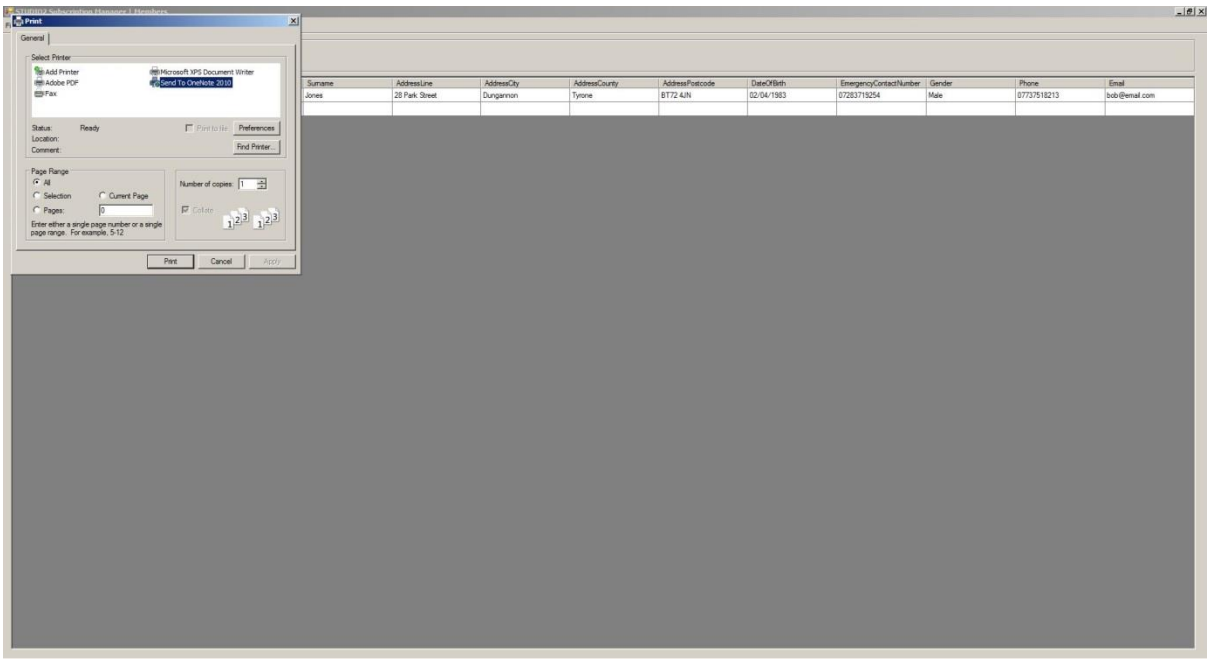
Test 2-3: Clicked 'Connect to Database' ToolStripMenuItem



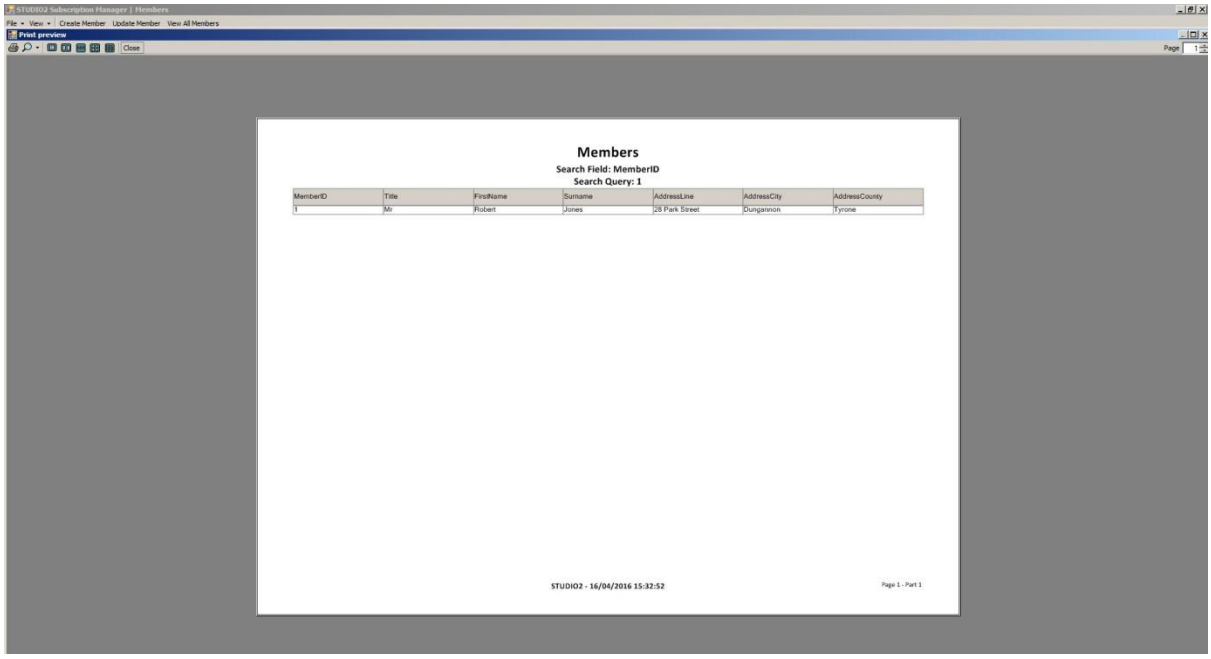
Test 2-4: Clicked 'Print' ToolStripMenuItem



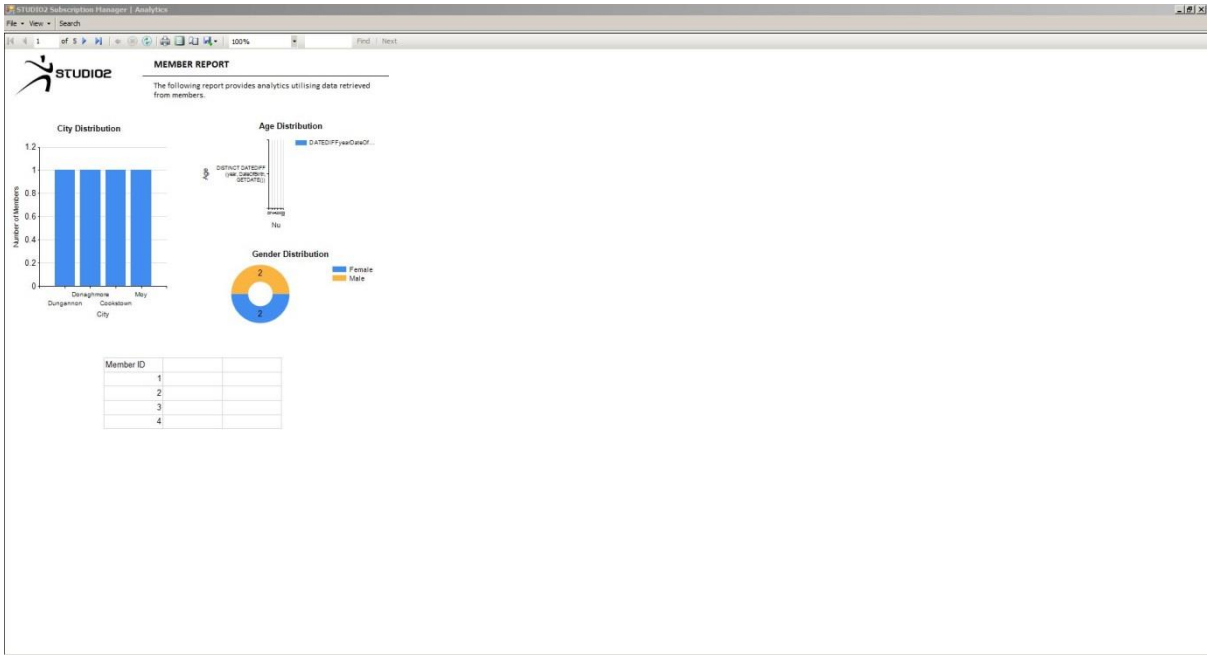
Test 2-4: Clicked 'Print' Button in Print Settings



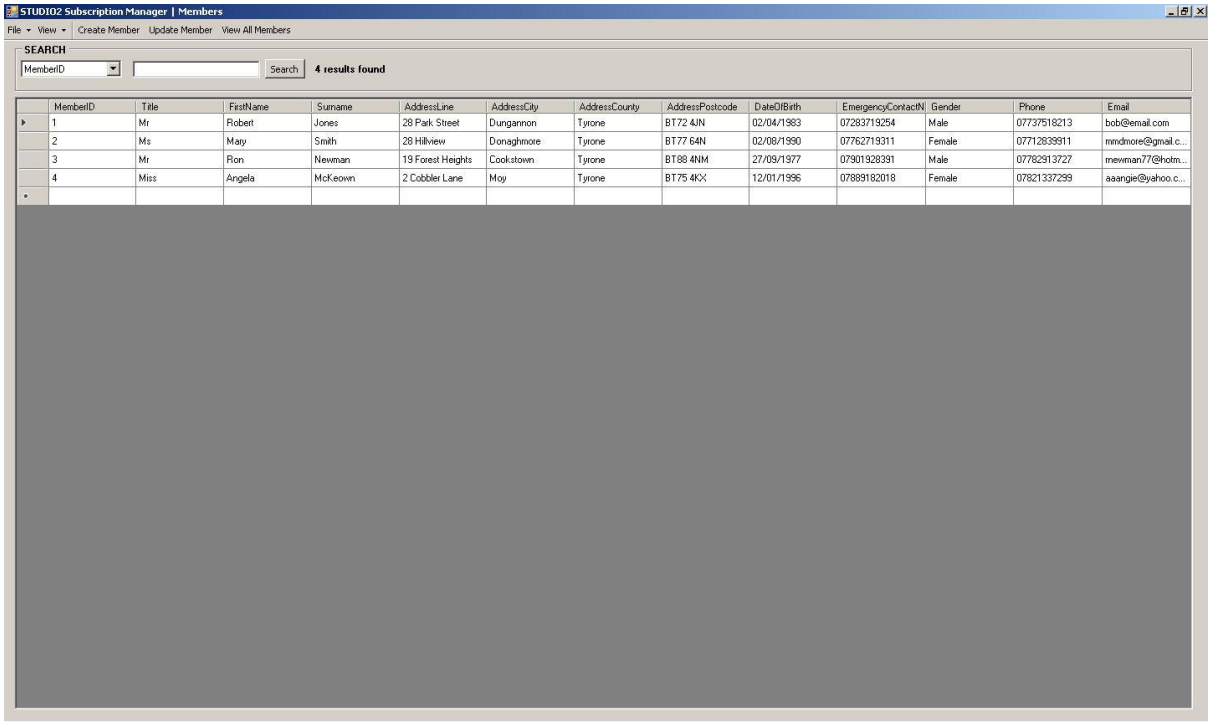
Test 2-5: Clicked 'Print' ToolStripMenuItem



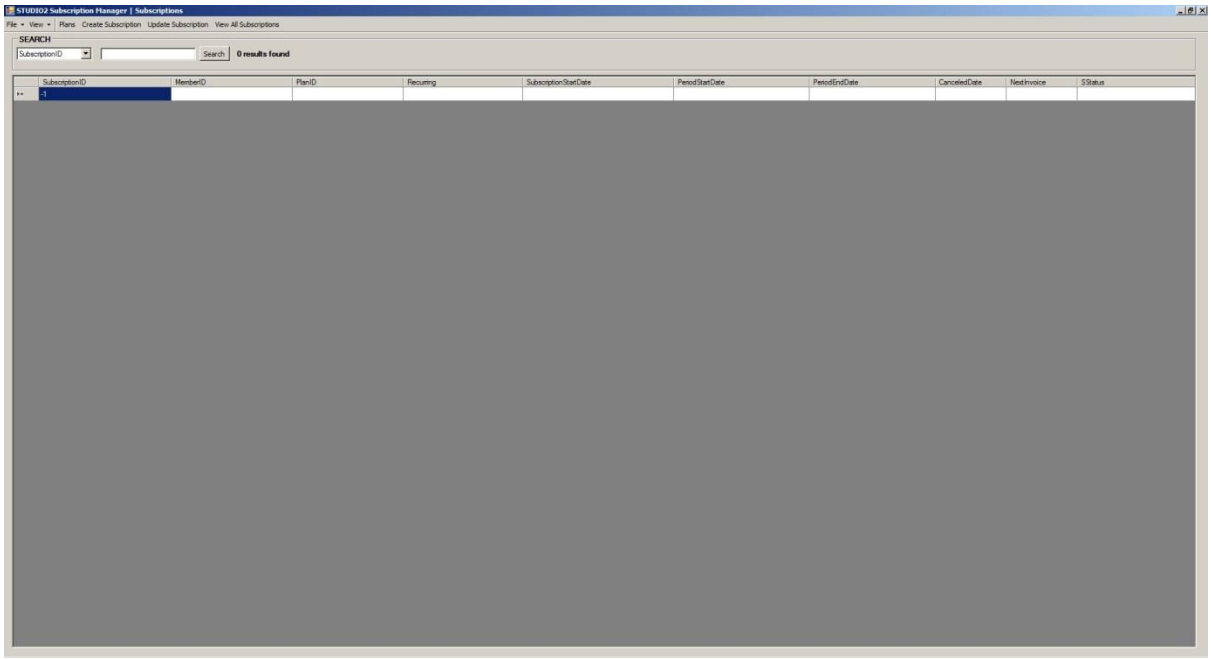
Test 2-5: Clicked 'Print' Button in Print Settings



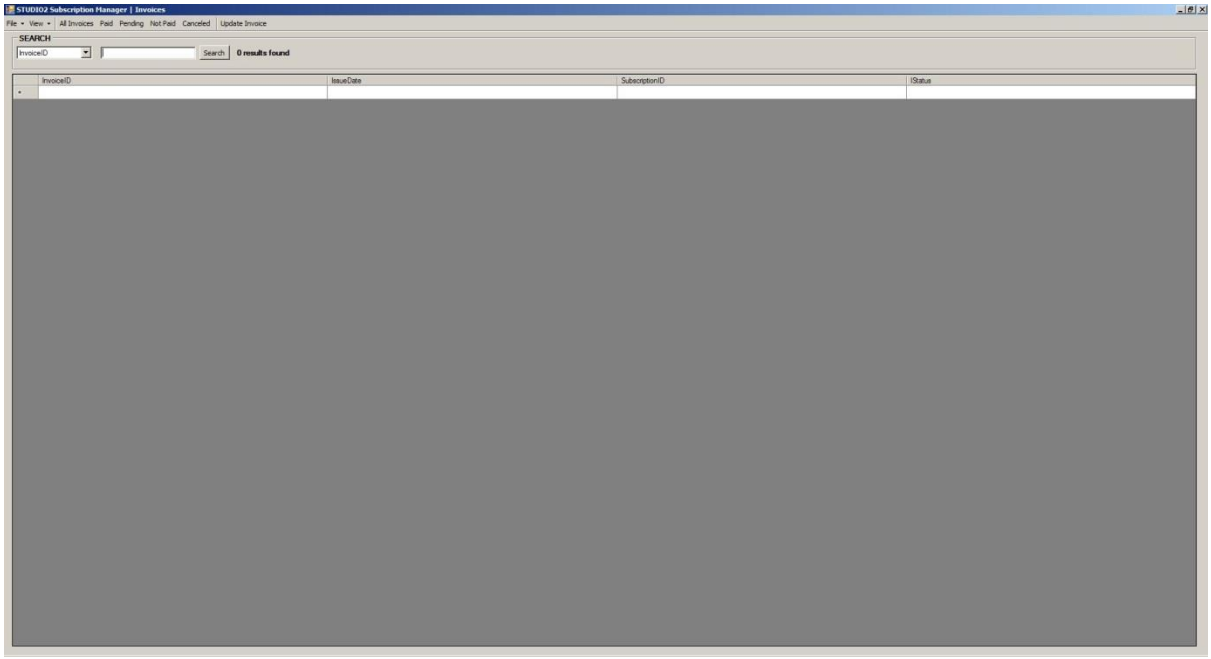
Test 2-6: Clicked 'Analytics' ToolStripMenuItem (FIX IMAGE)



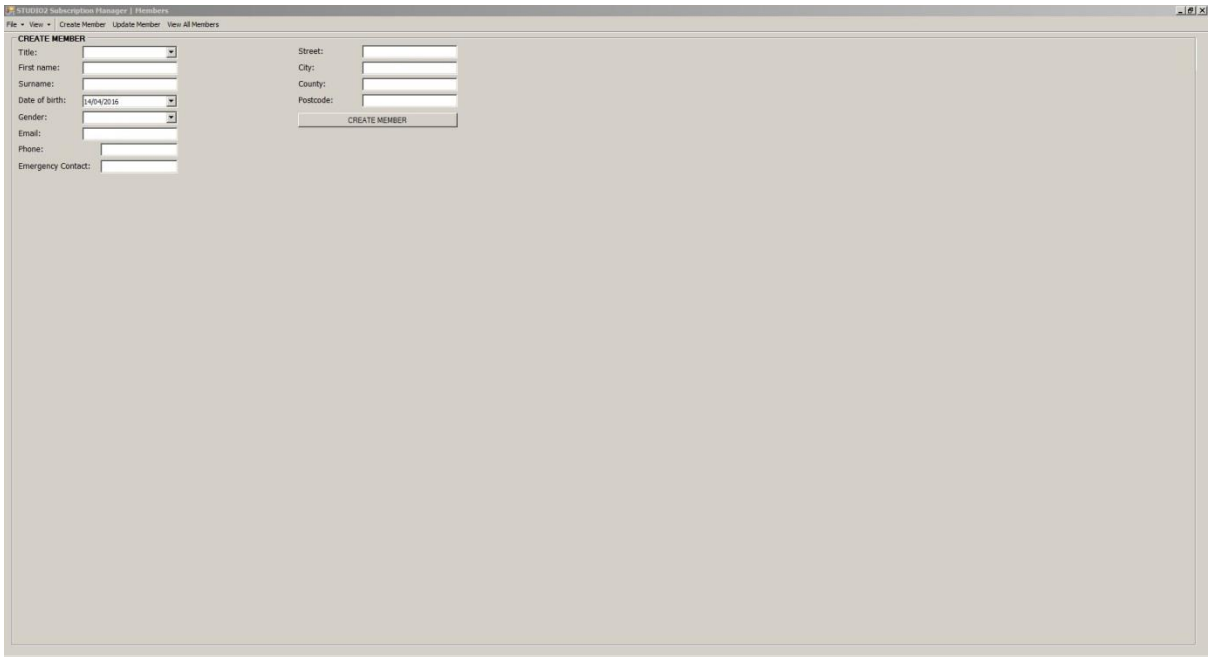
Test 2-7: Clicked 'Members' ToolStripMenuItem



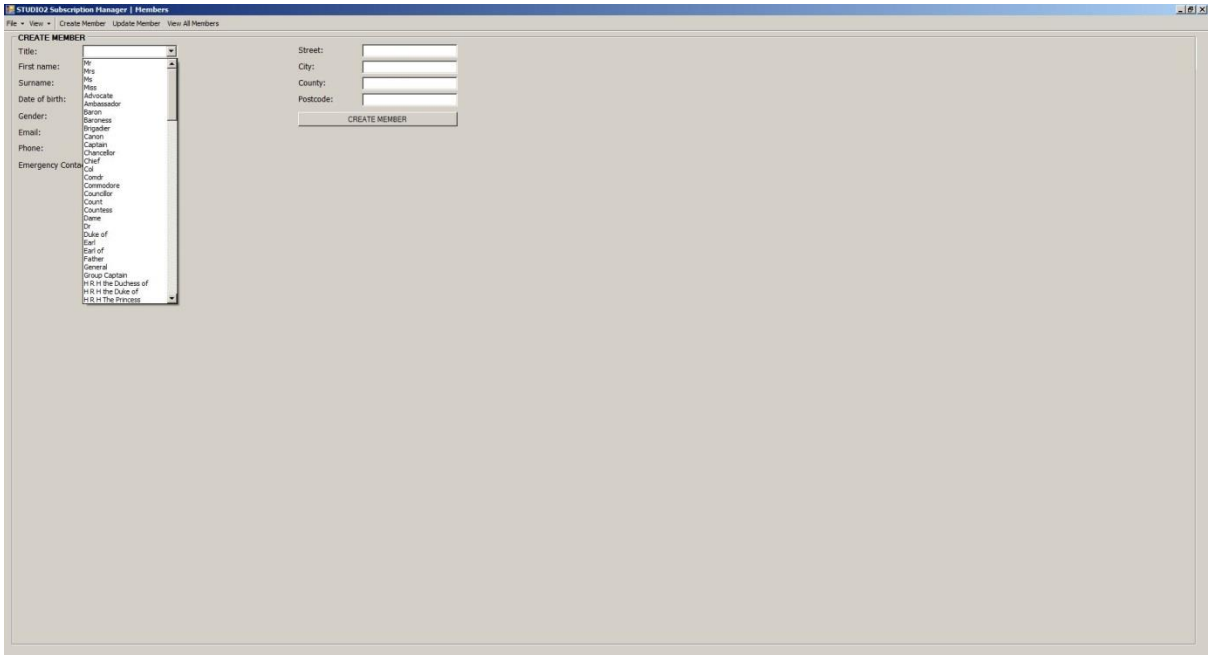
Test 2-8: Clicked 'Subscriptions' ToolStripMenuItem



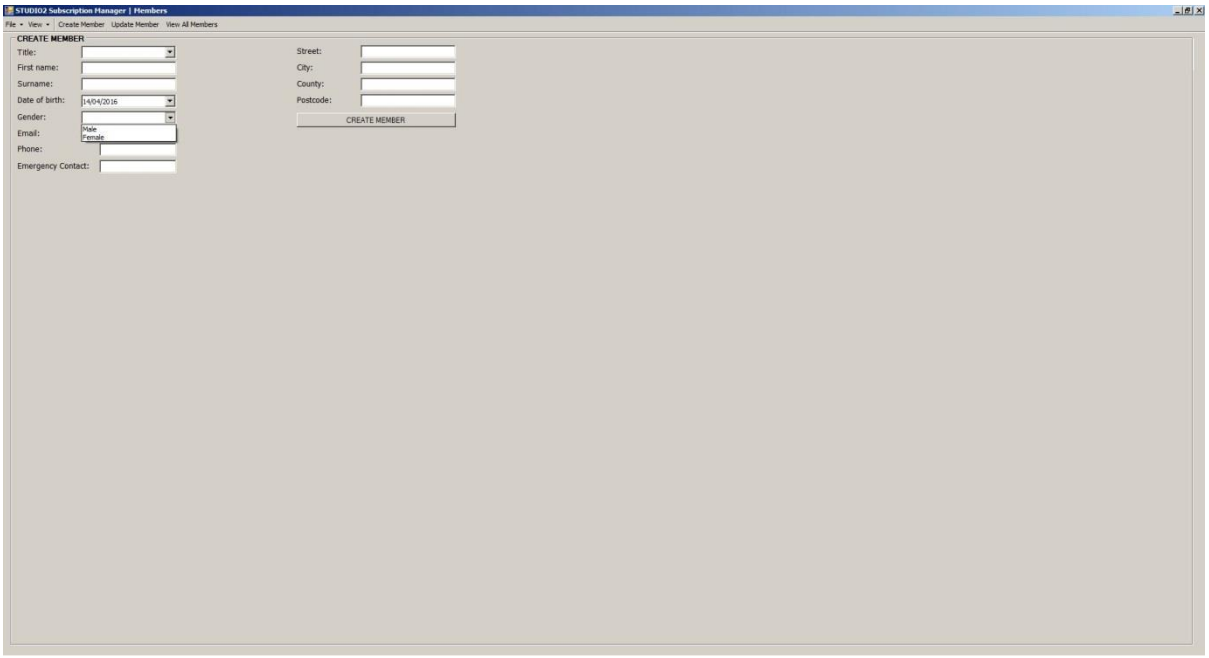
Test 2-9: Clicked 'Invoices' ToolStripMenuItem



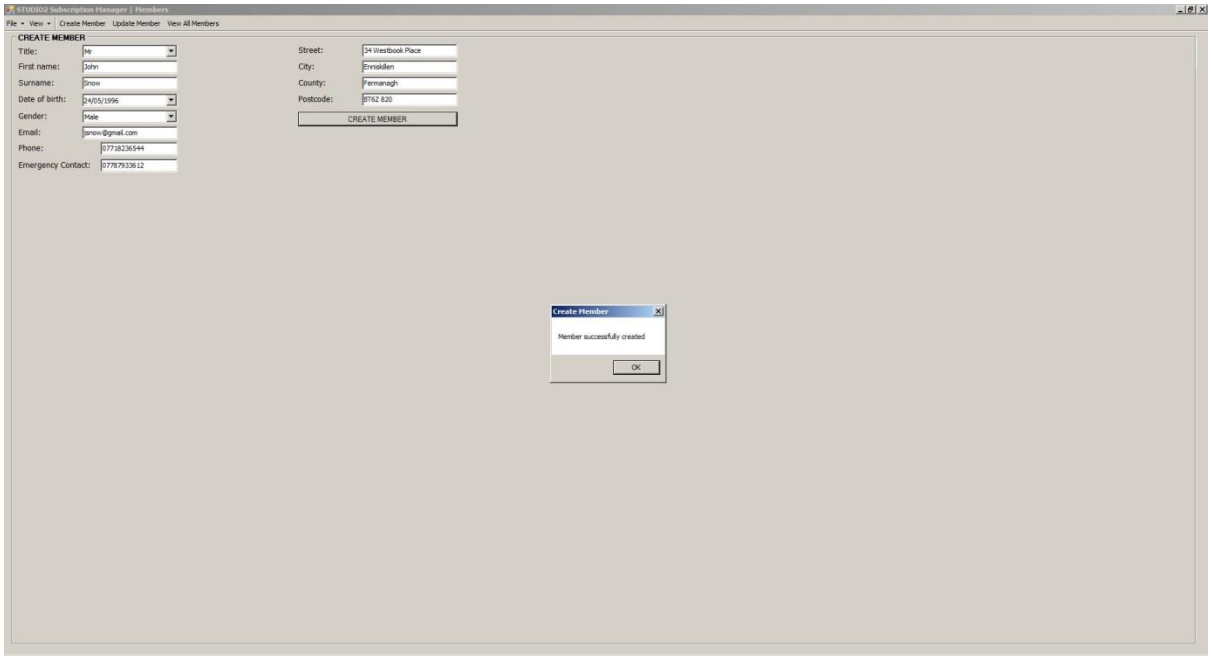
Test 2-10: Click 'Create Member' ToolStripButton



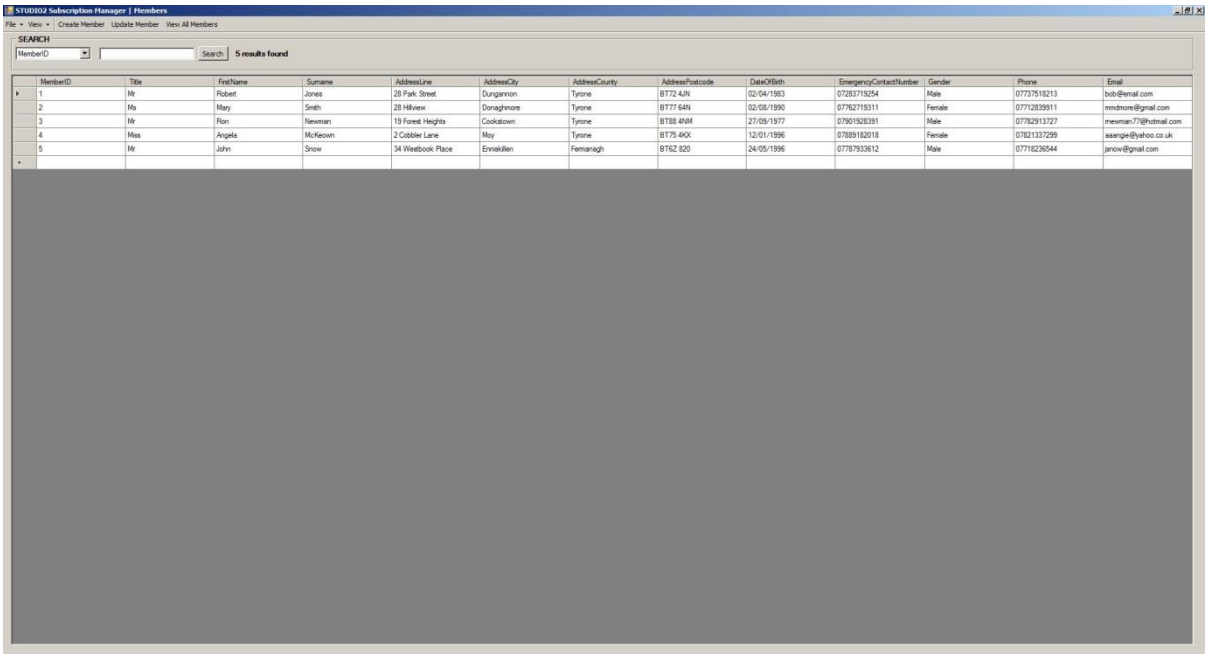
Test 2-11: Click 'Title' ComboBox



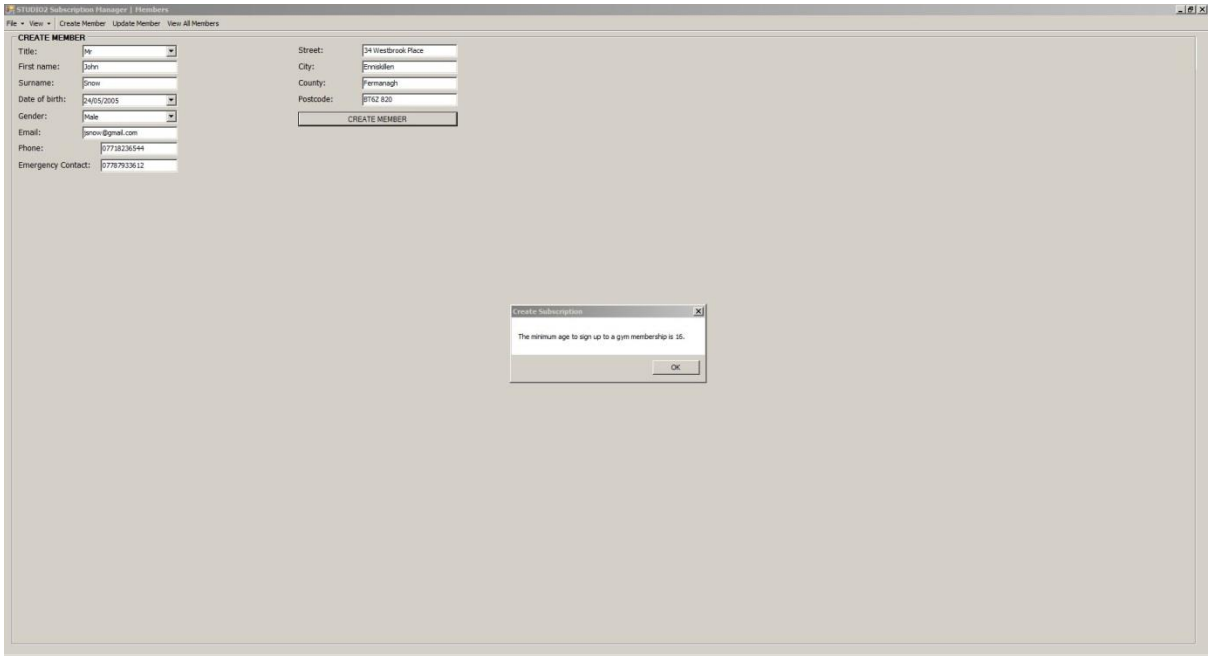
Test 2-12: Click 'Gender' ComboBox



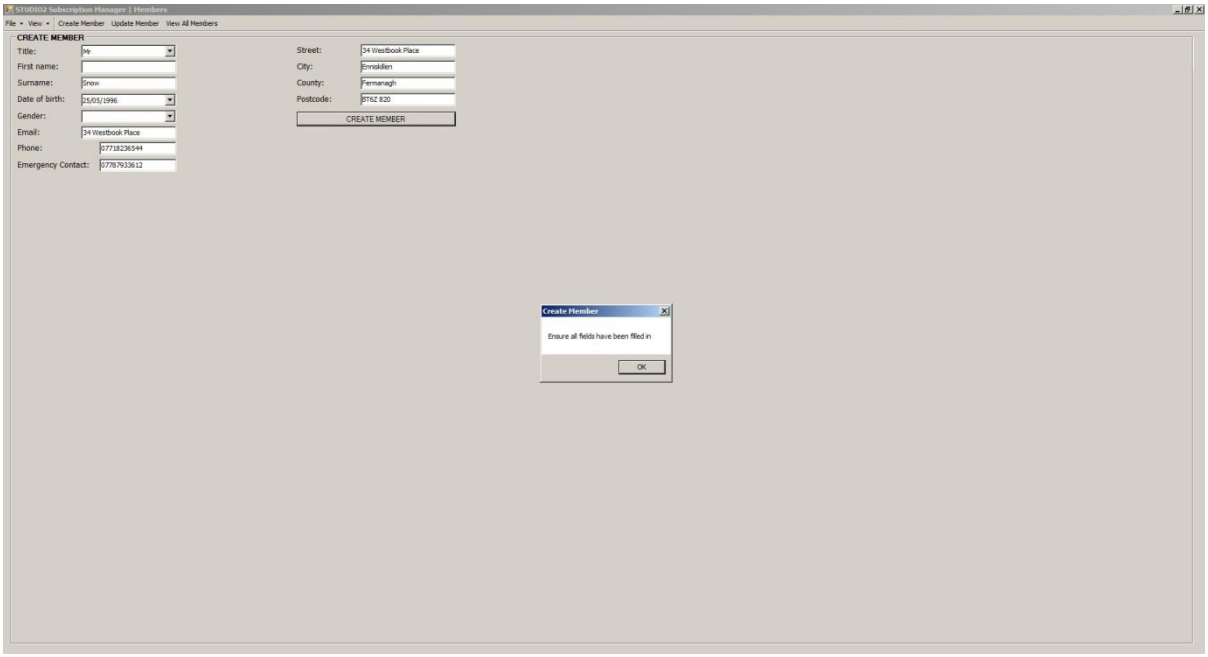
Test 2-13: Fill in all fields with valid values and click 'Create Member' Button



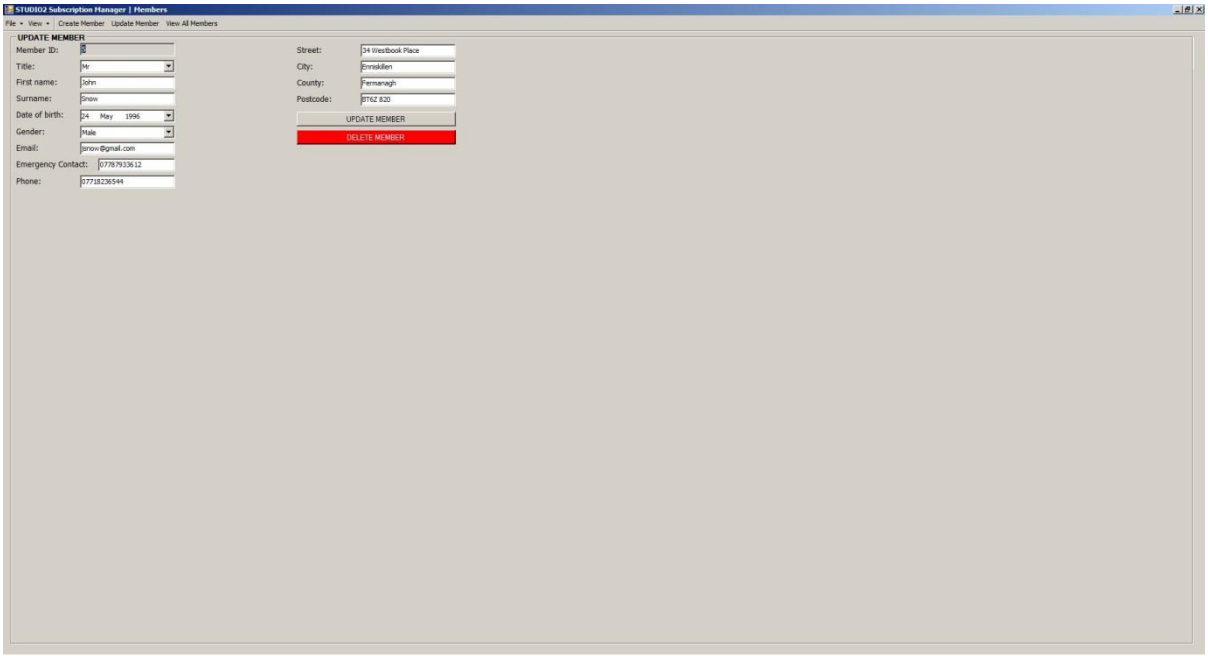
Test 2-13: New member displayed in dgvSQLOutput



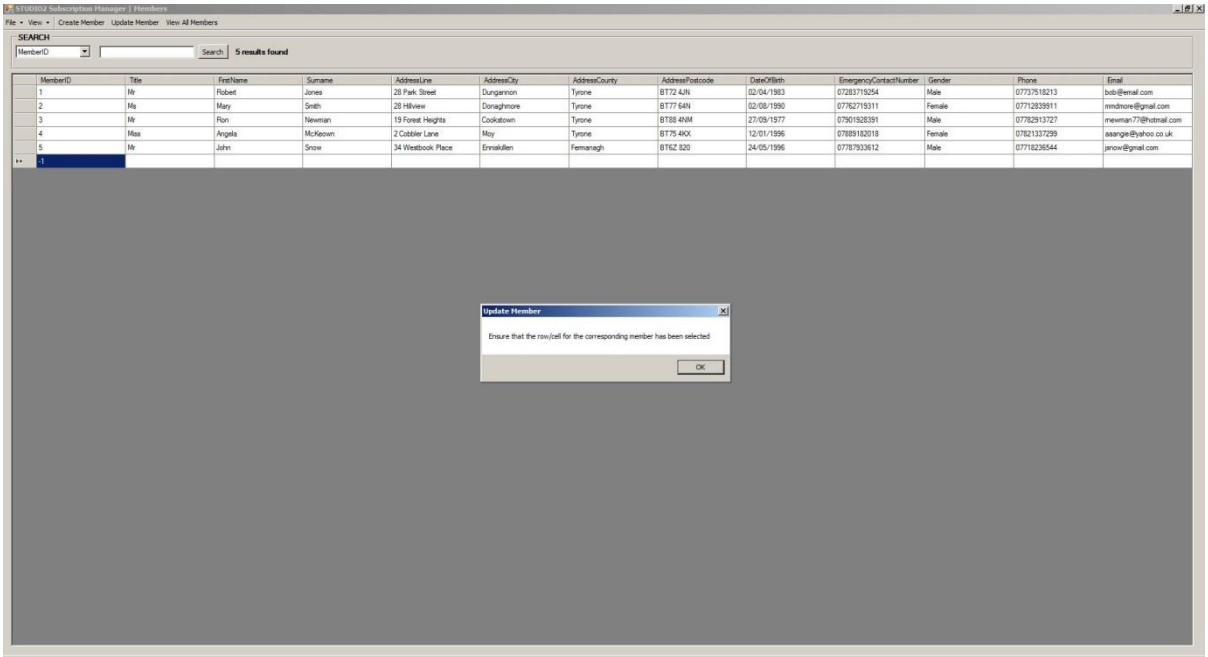
Test 2-14: Fill in all fields with valid values but set Date of Birth less than 16 years ago and click 'Create Member' Button



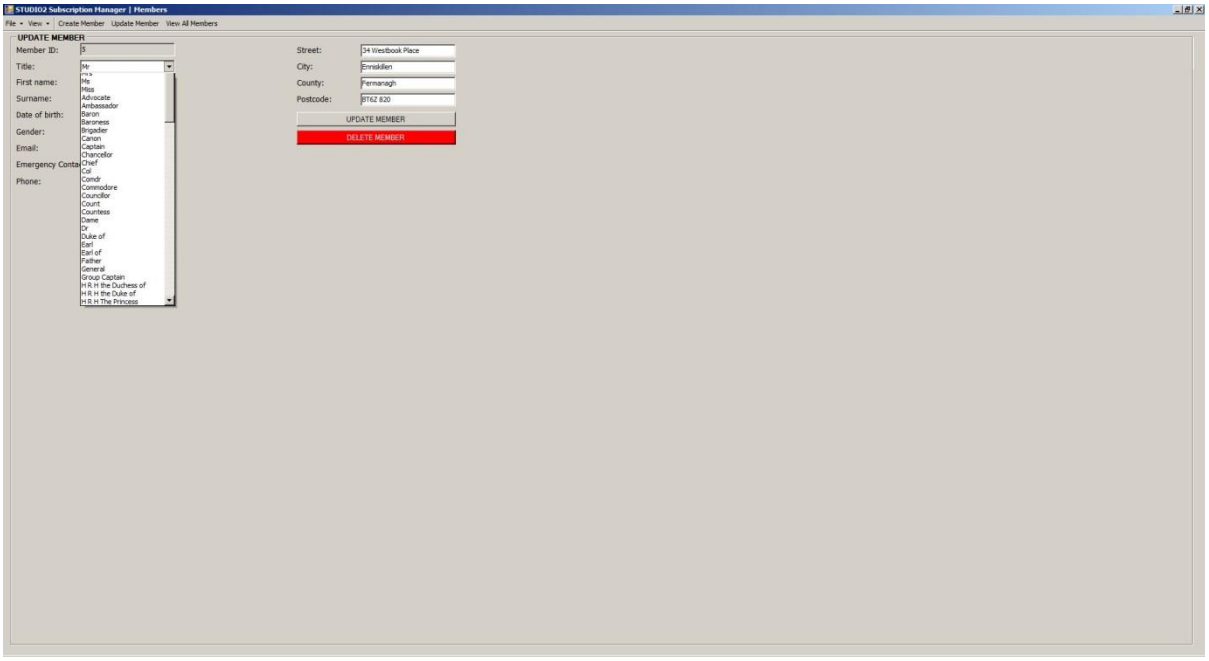
Test 2-15: Fill in all fields with valid values but one ComboBox and TextBox and click 'Create Member' Button



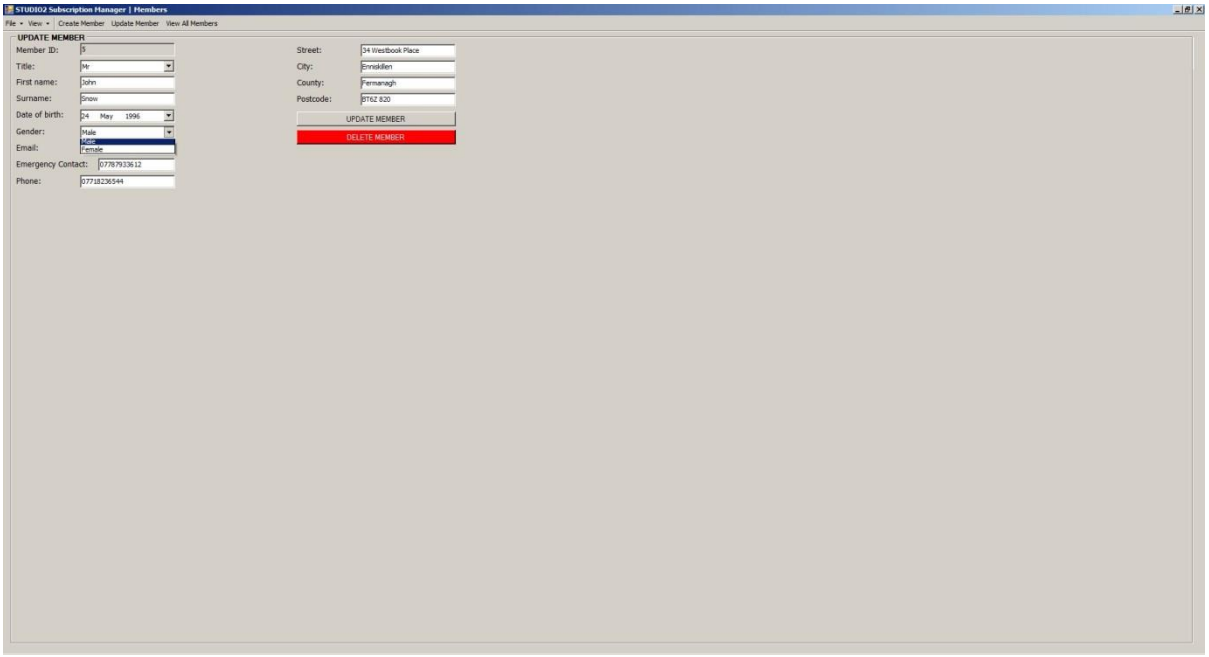
Test 2-16: Click 'Update Member' when a valid cell has been selected in dgvSQLOutput



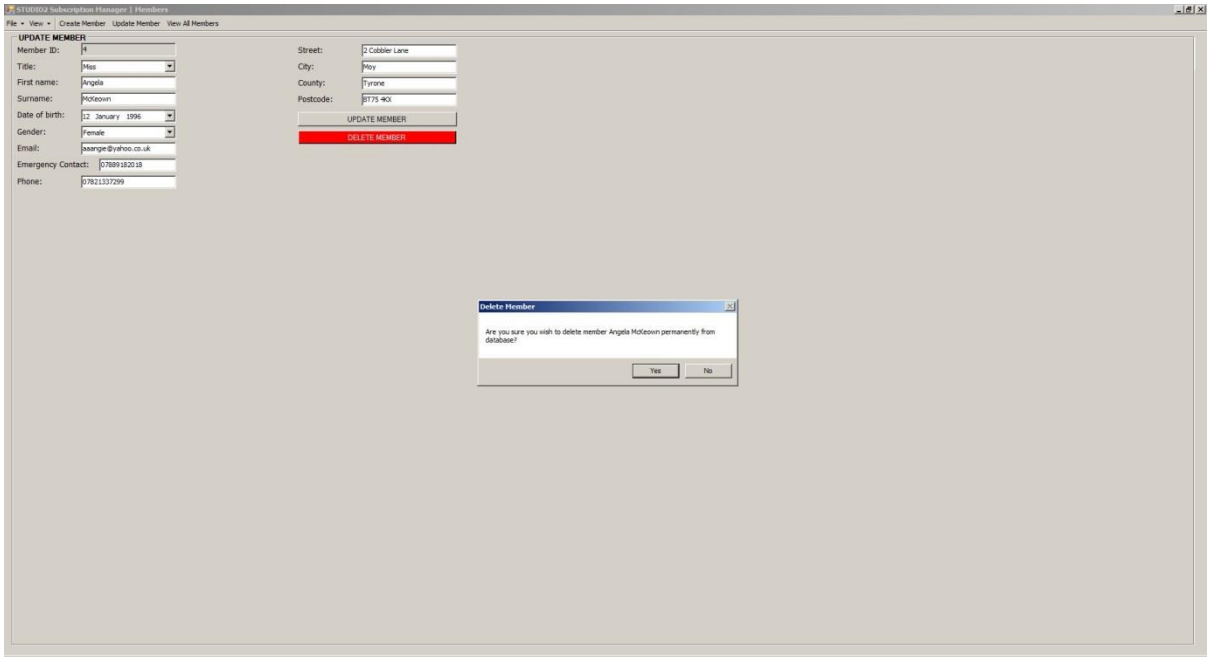
Test 2-17: Click 'Update Member' when an invalid cell has been selected in dgvSQLOutput



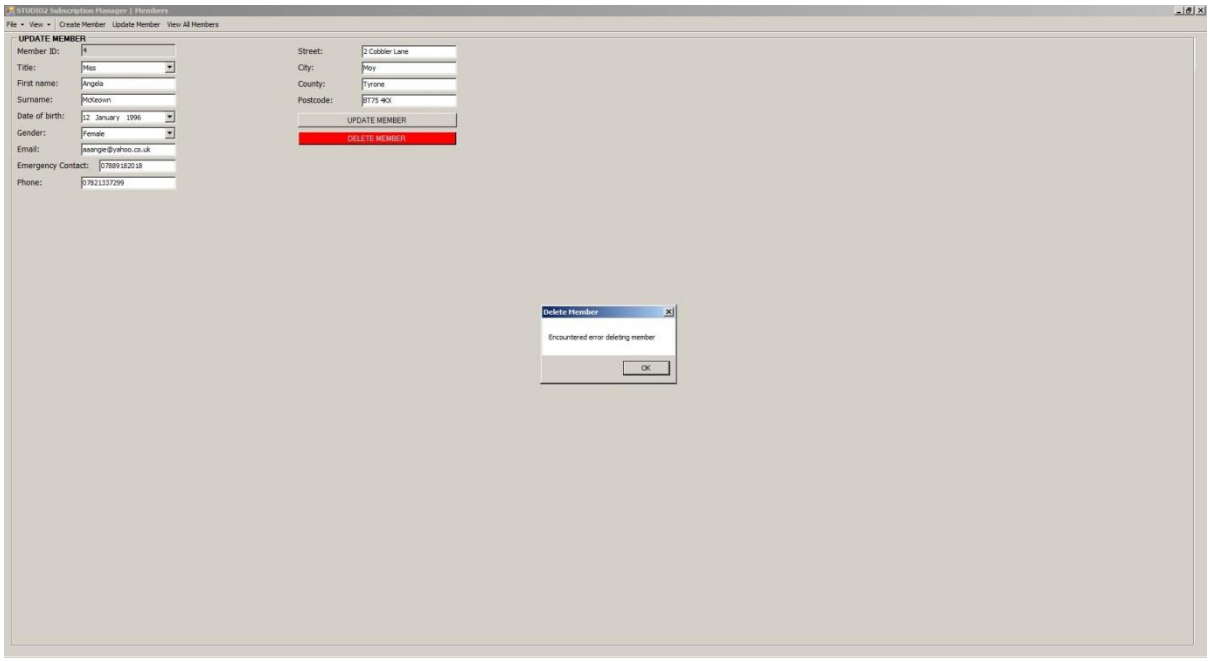
Test 2-18: Click 'Title' ComboBox



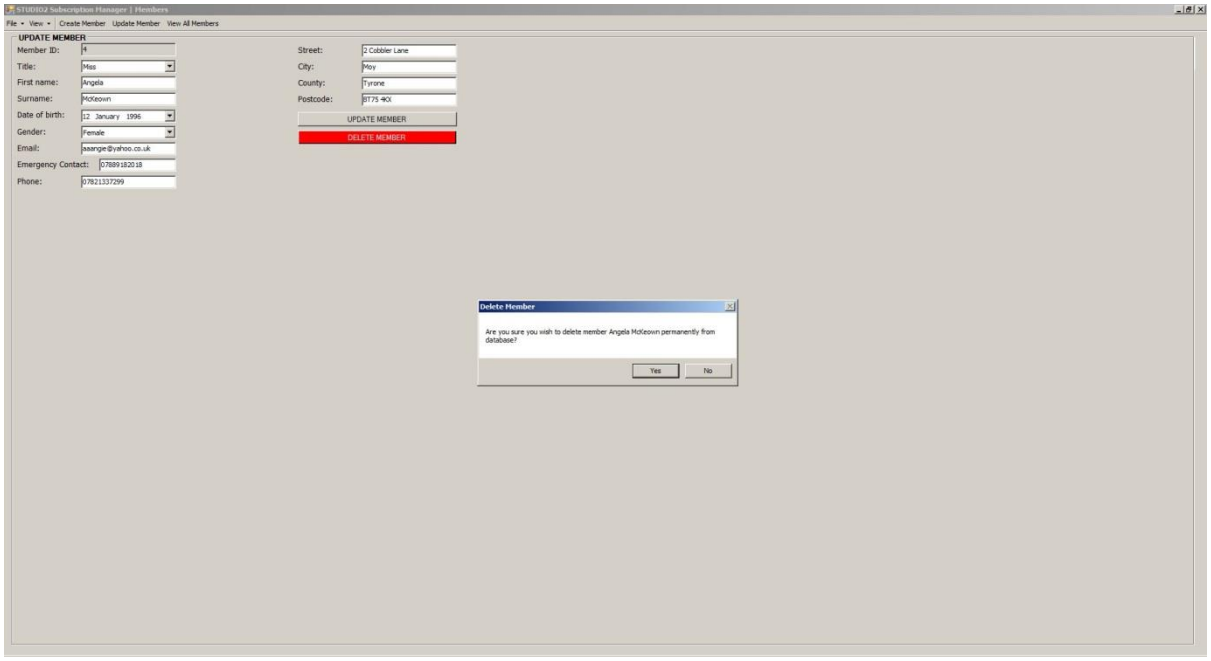
Test 2-18: Click 'Gender' ComboBox



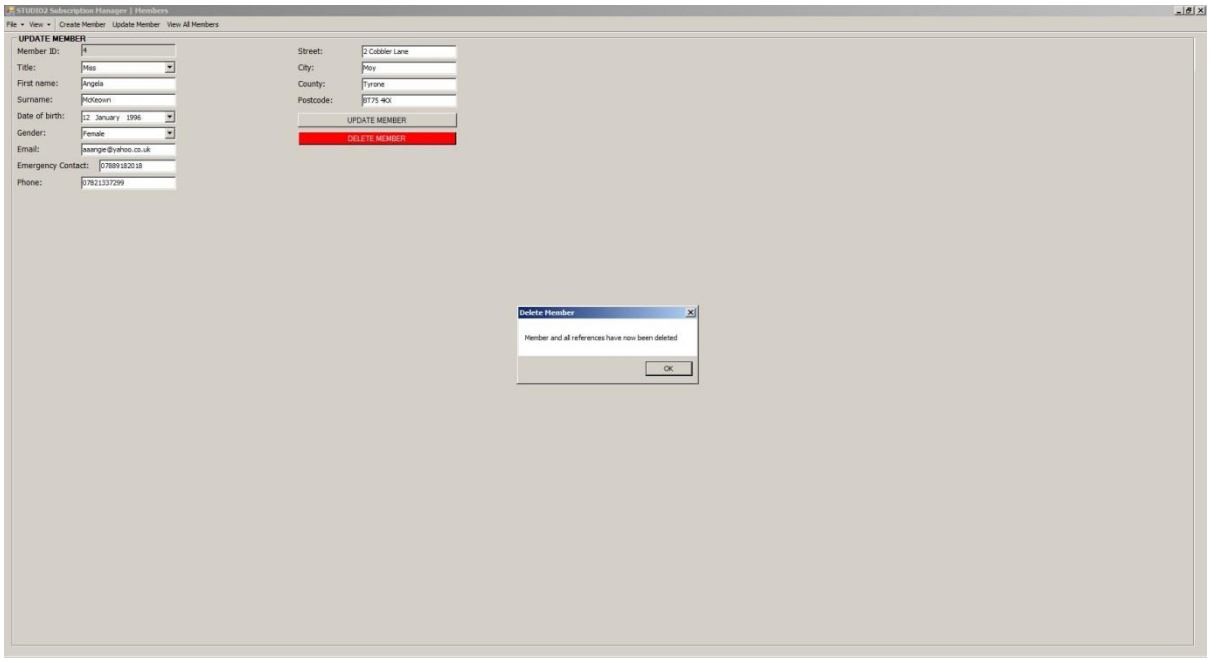
Test 2-19: Click 'Delete Member' Button



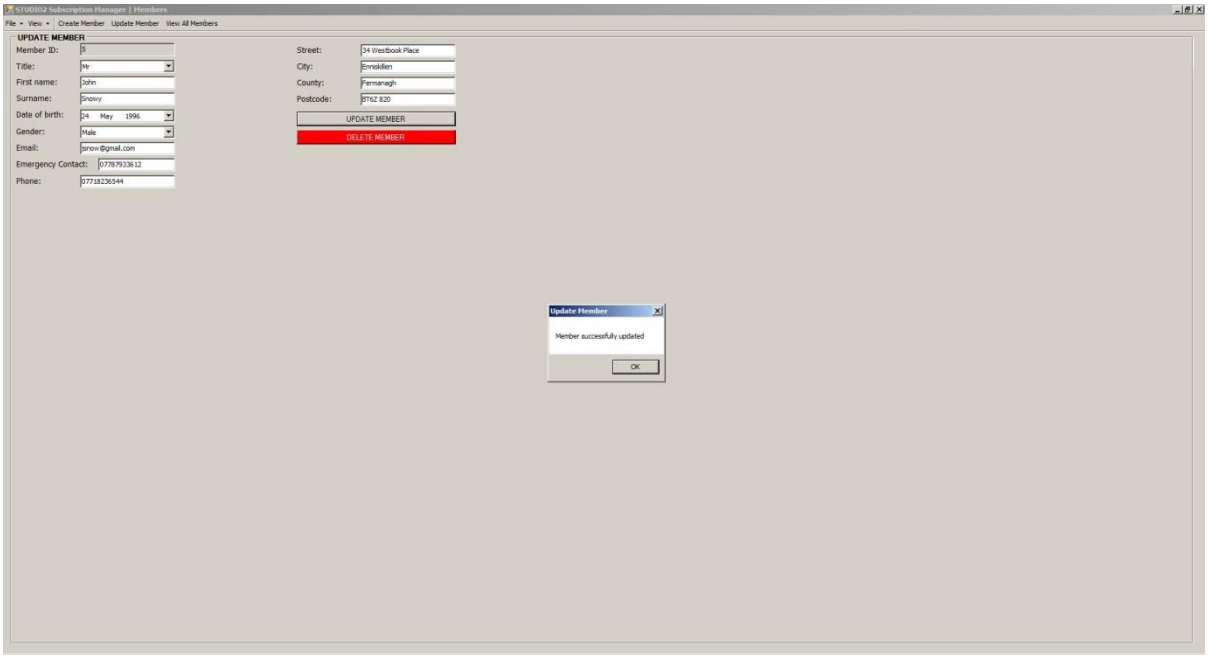
Test 2-19: Error encountered



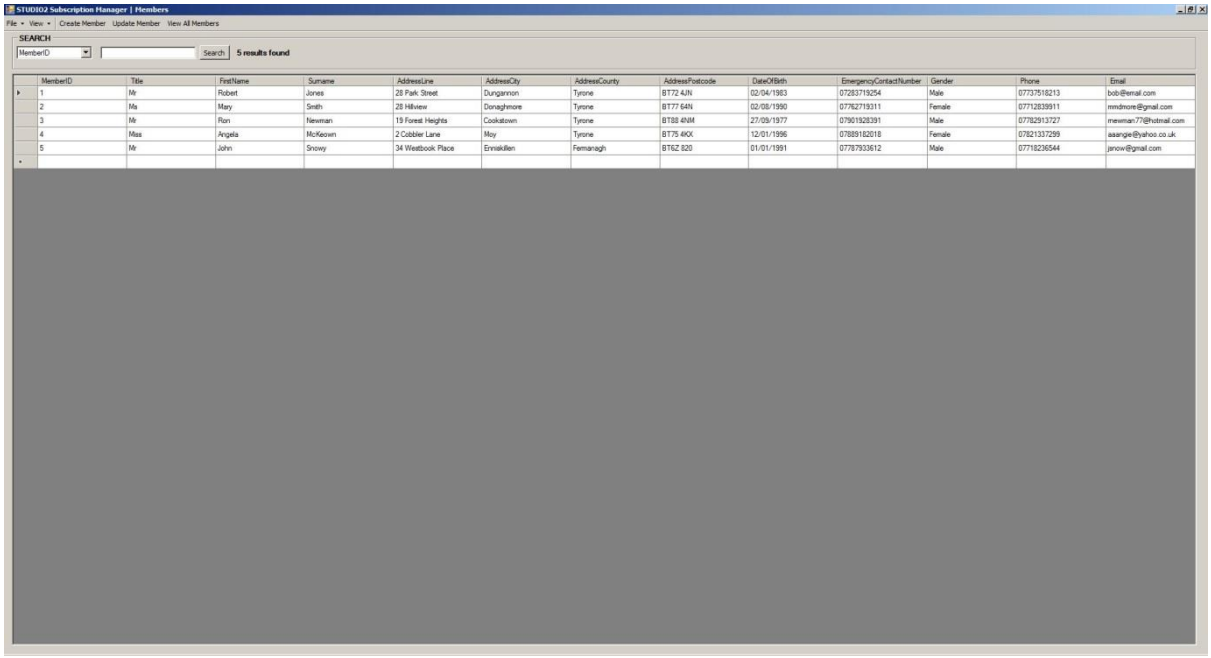
Test 2-19: Corrective action taken. 'Delete Member'



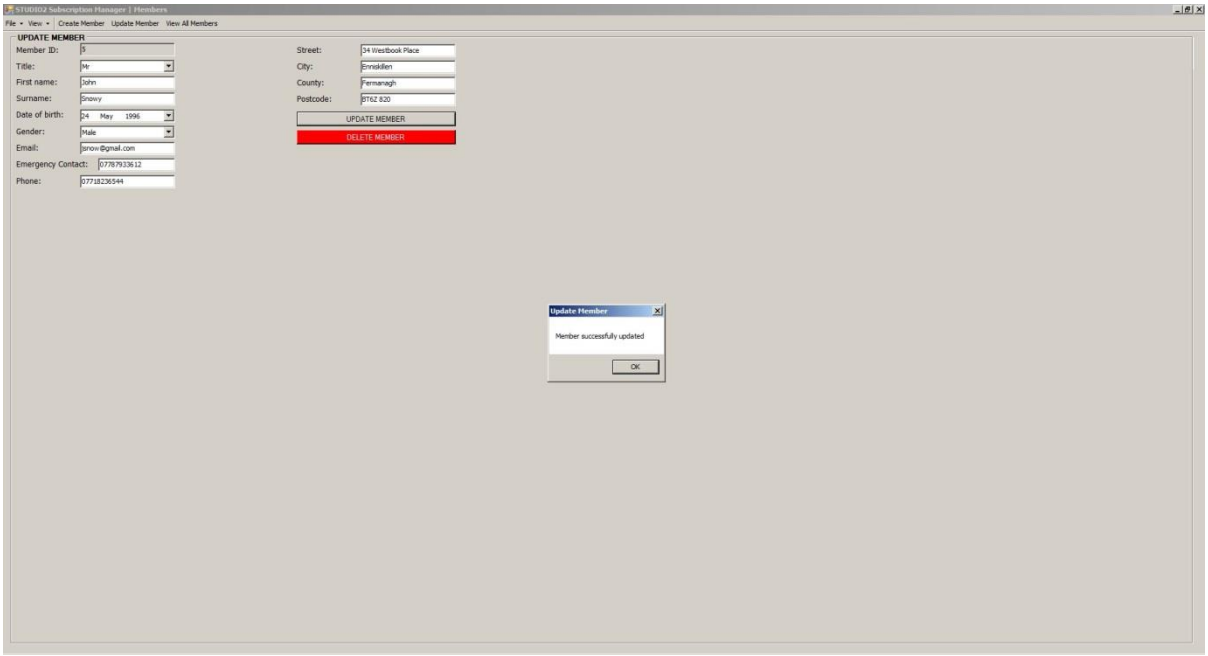
Test 2-19: Member deleted as intended



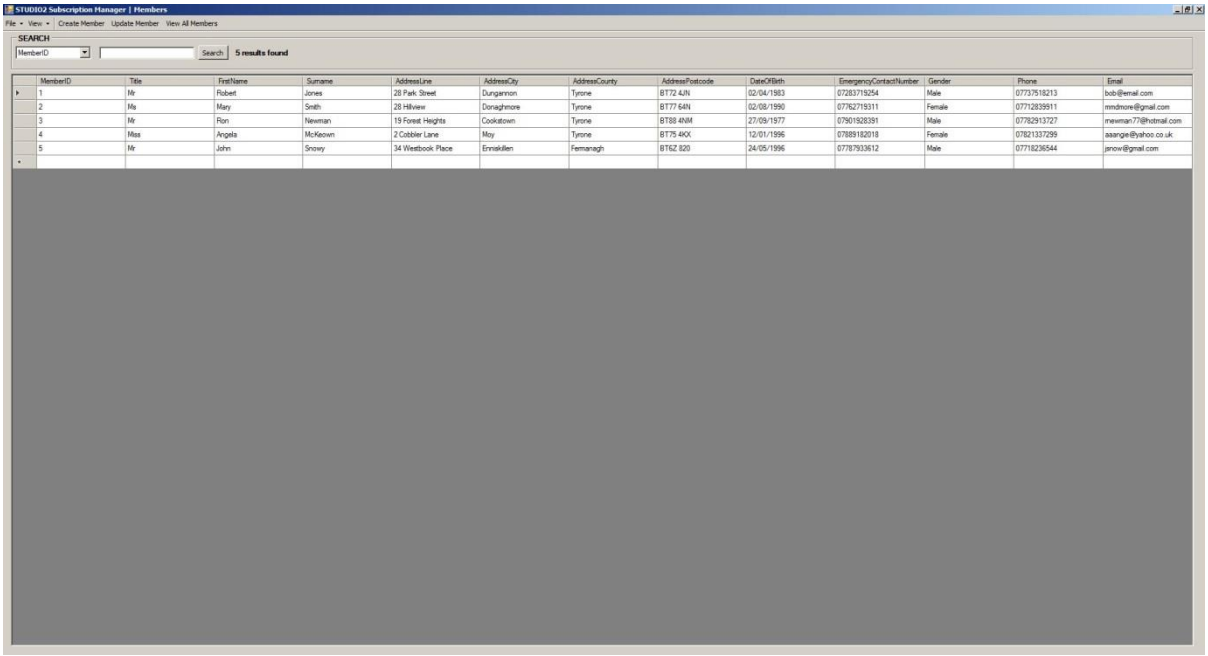
Test 2-20: Click 'Update Member' Button after changing surname



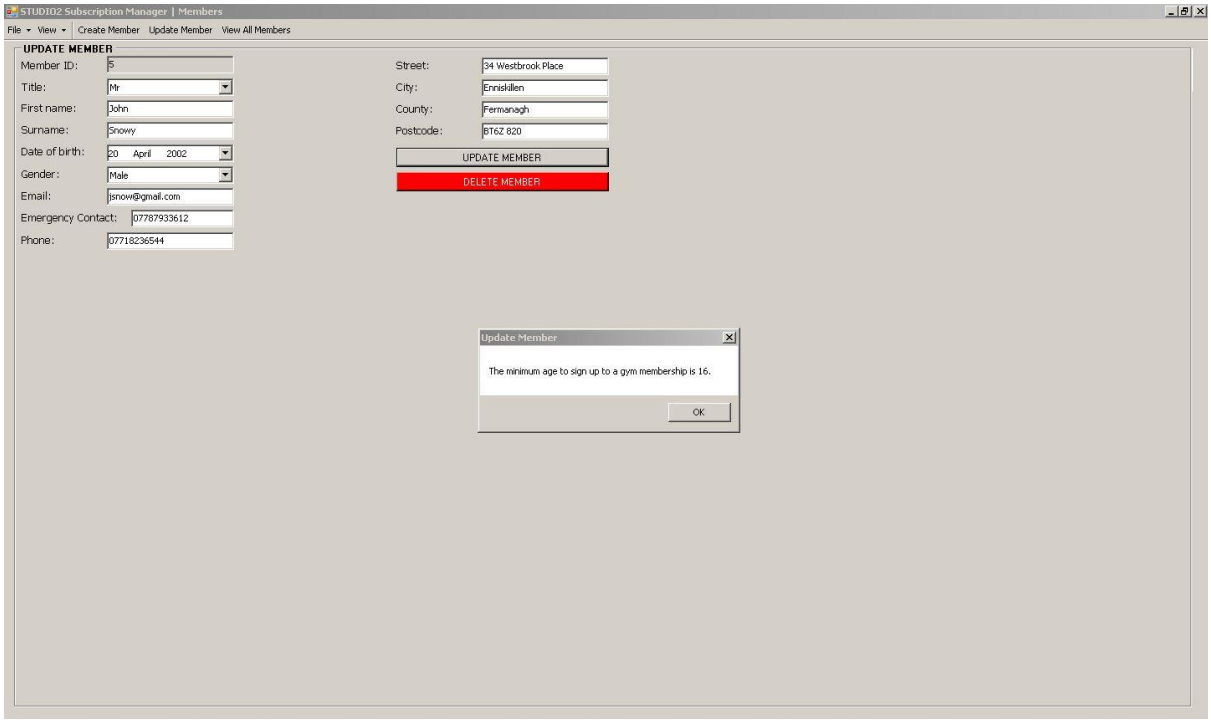
Test 2-20: Surname updated as well as DateOfBirth value



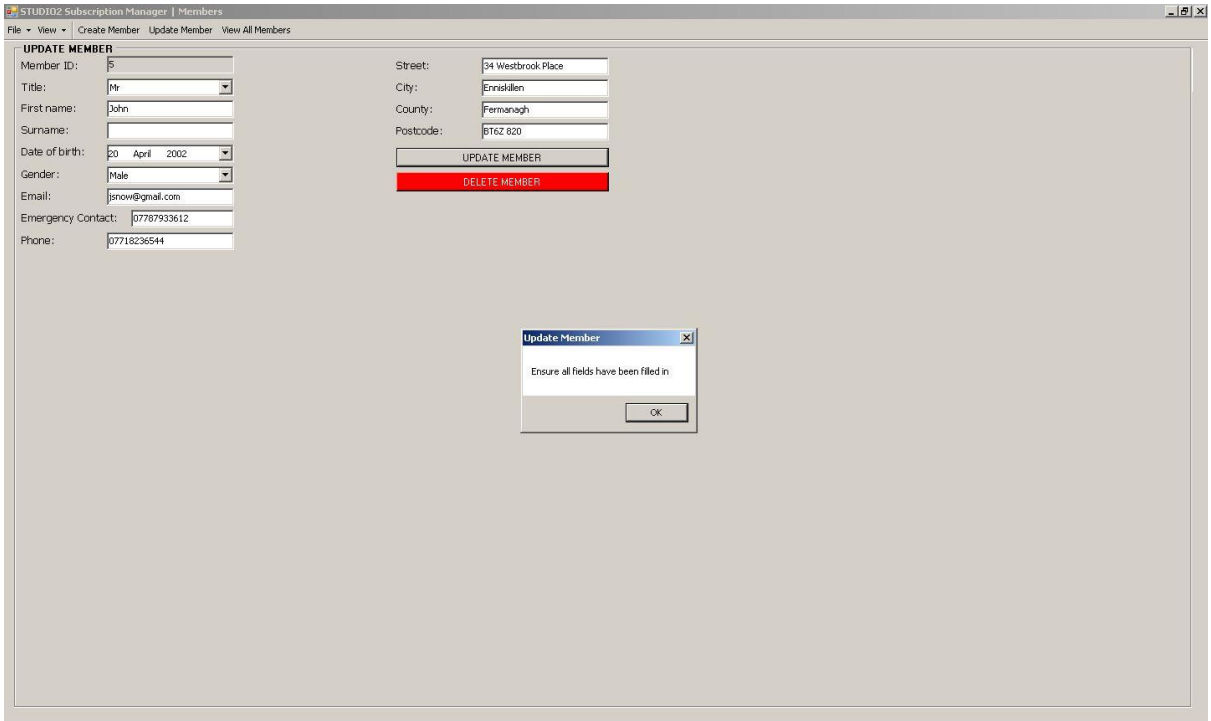
Test 2-20: Corrections made and test re-done



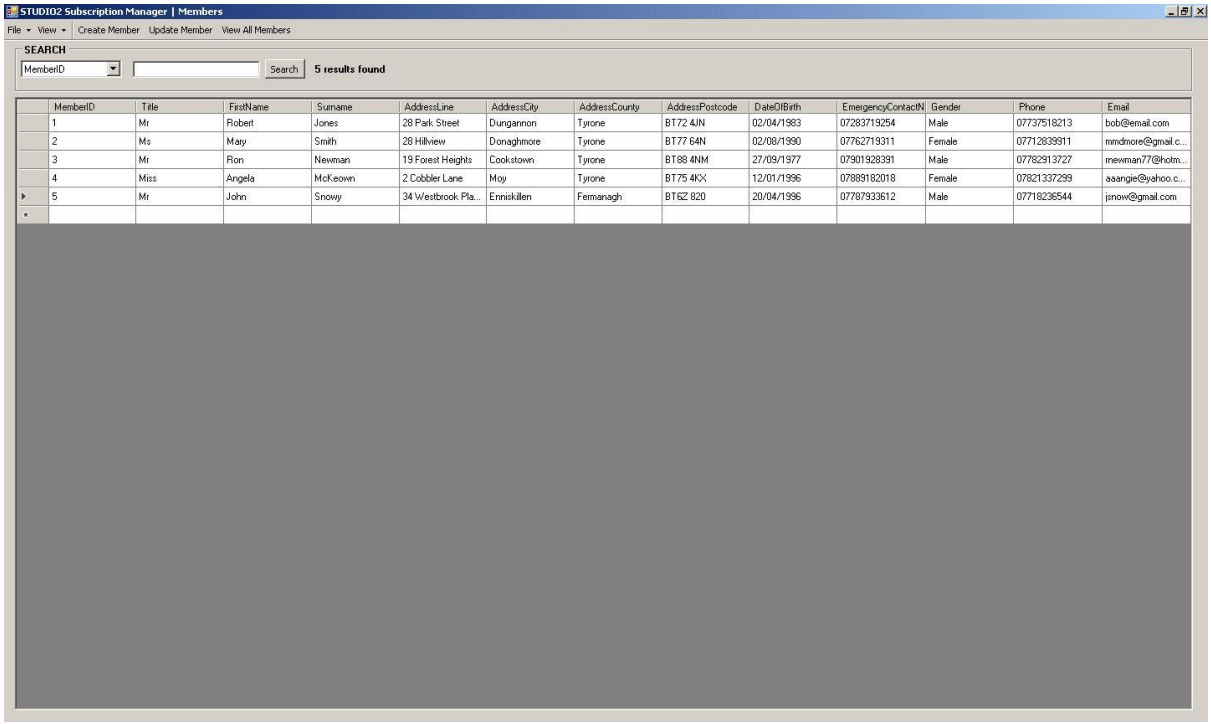
Test 2-20: Surname updated and DateOfBirth remained unaffected



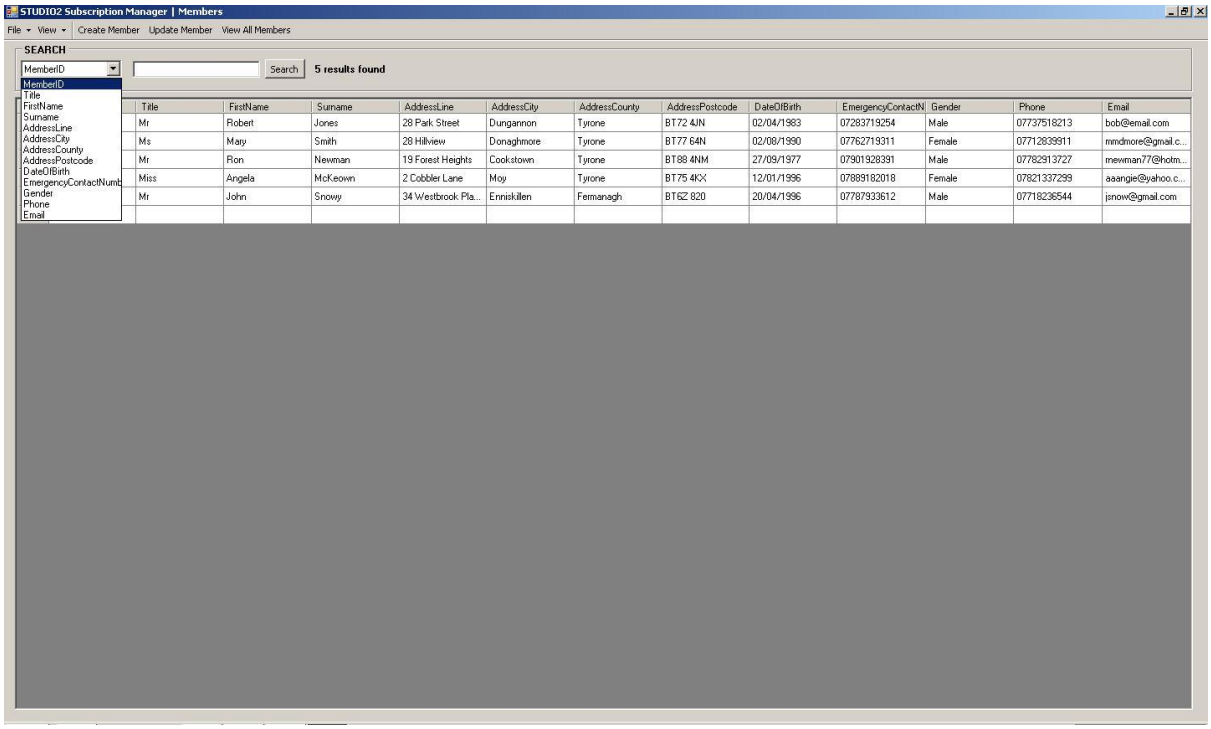
Test 2-21: Click 'Update Member' Button when Date Of Birth is less than 16 years



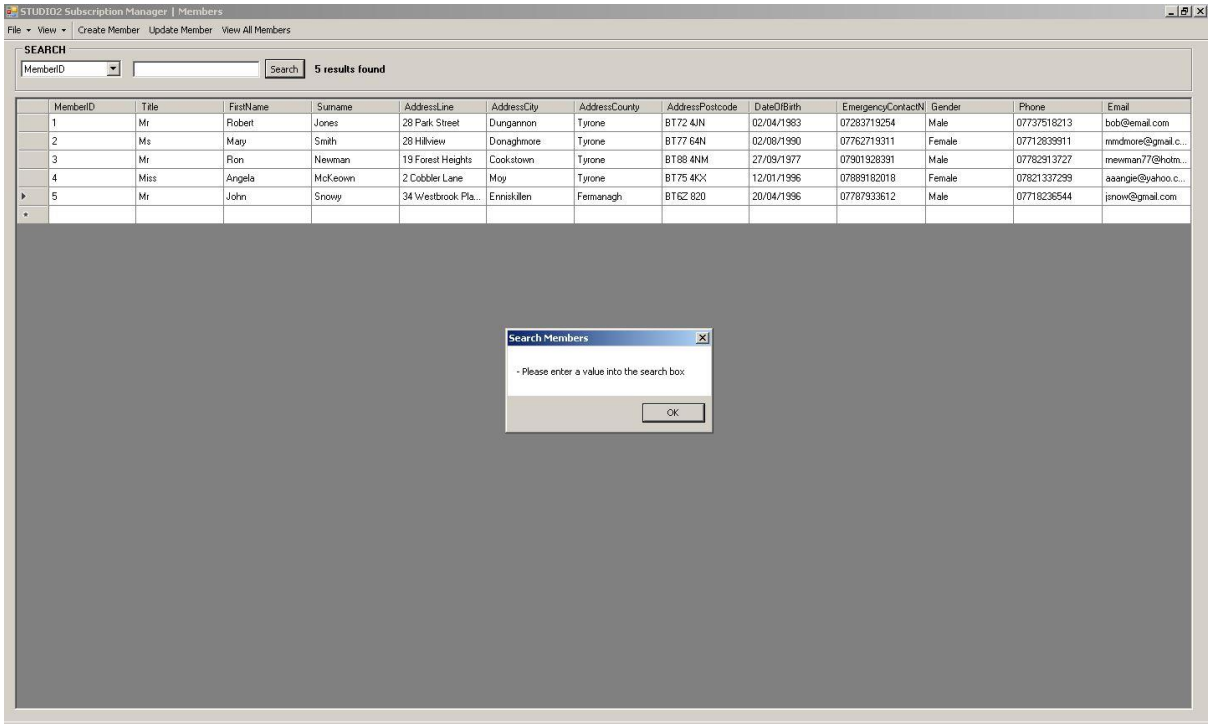
Test 2-22: Click 'Update Member' when one TextBox has not been filled in



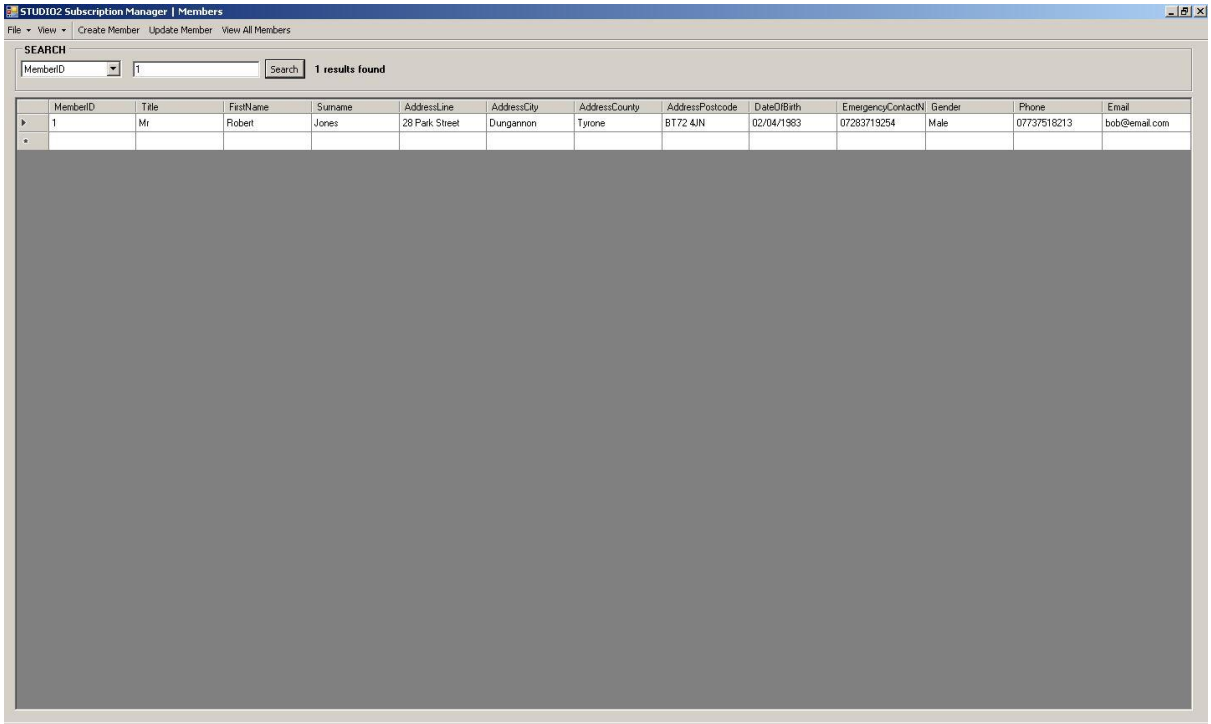
Test 2-23: Click 'View All Members' ToolStripButton



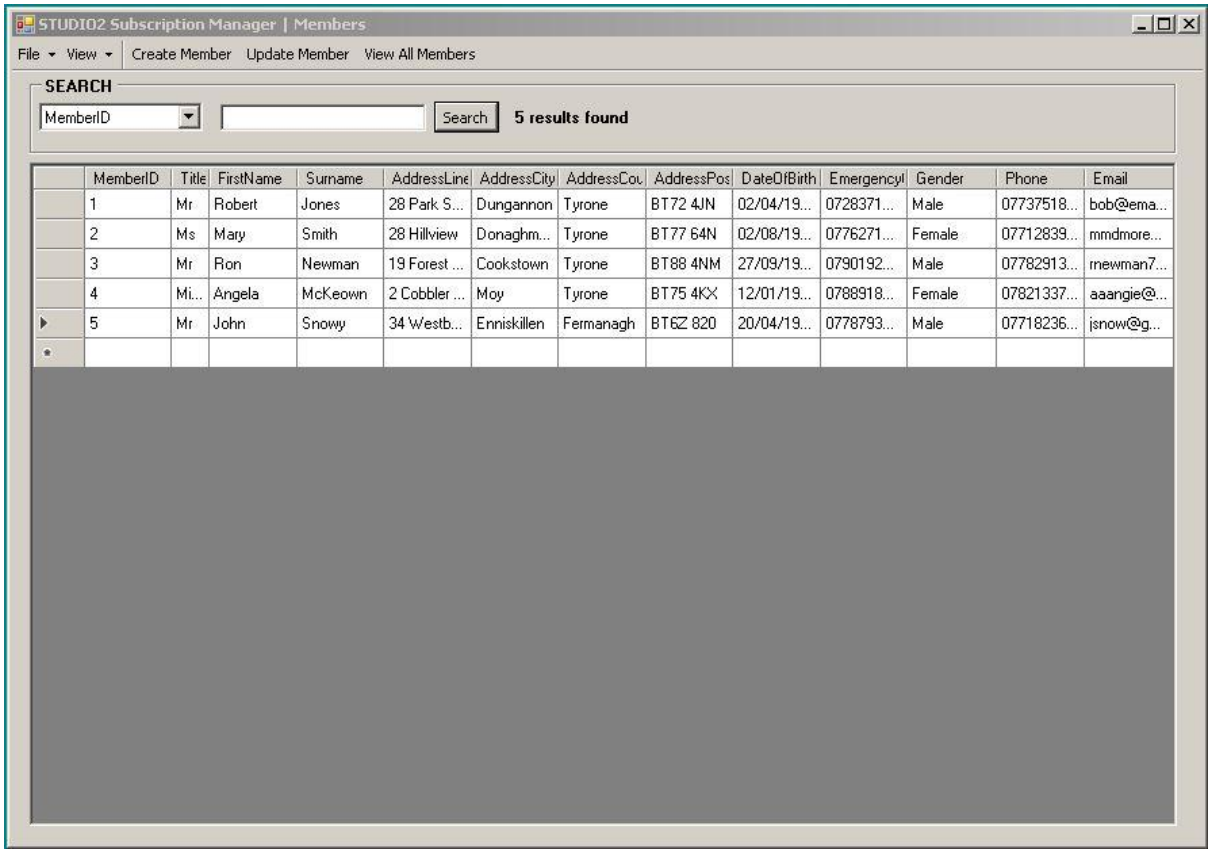
Test 2-24: Click cboSearchField ComboBox



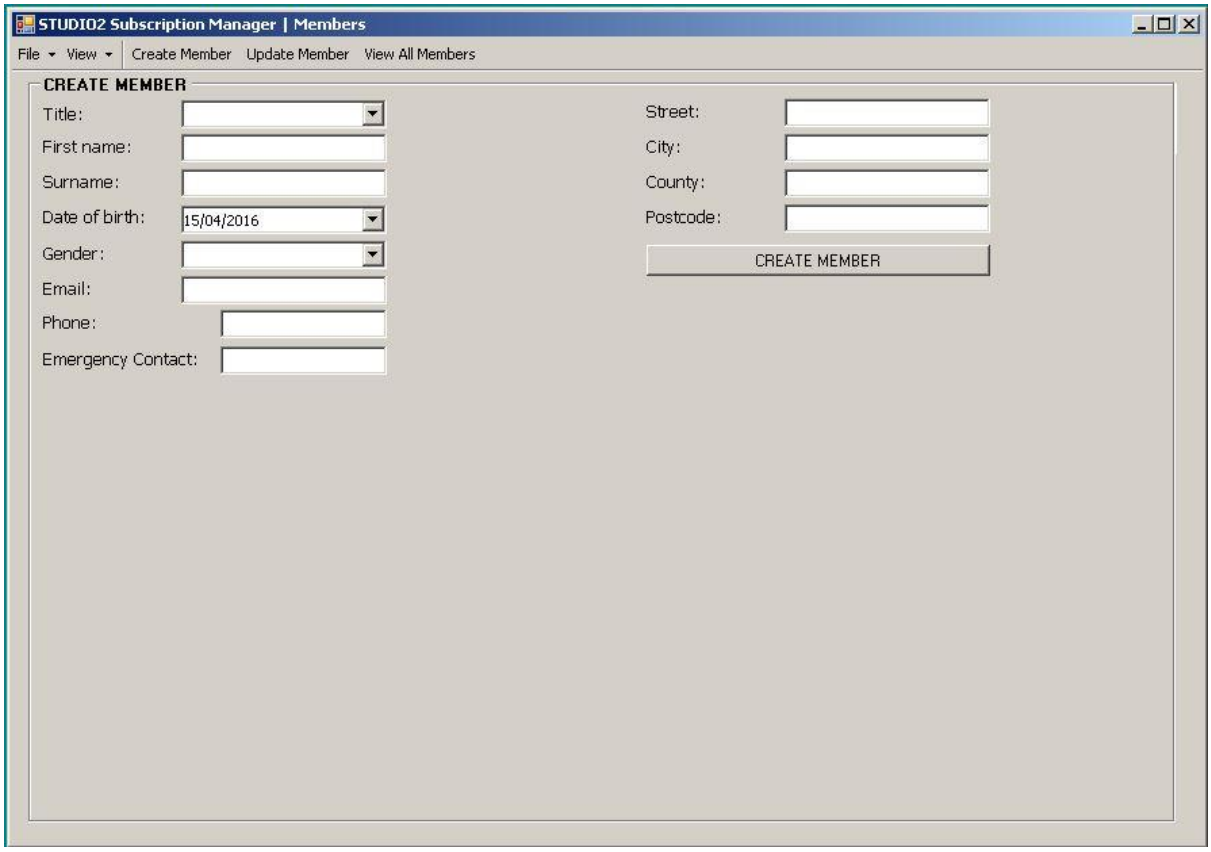
Test 2-25: Click 'Search' Button when no text has been entered into txtSearch TextBox



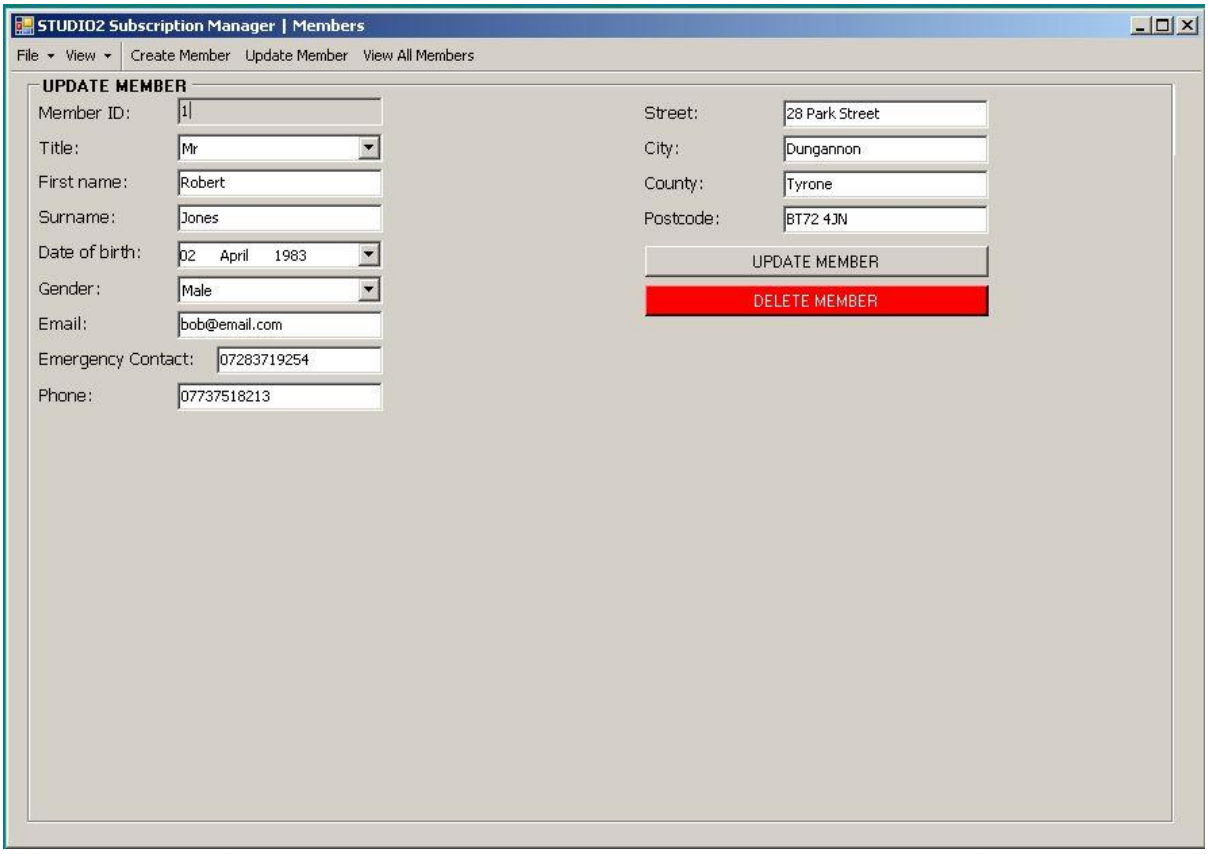
Test 2-26: Click 'Search' Button when '1' has been entered into txtSearch TextBox and cboSearchField text is 'MemberID



Test 2-27: Resize form when dgvSQLOutput is visible

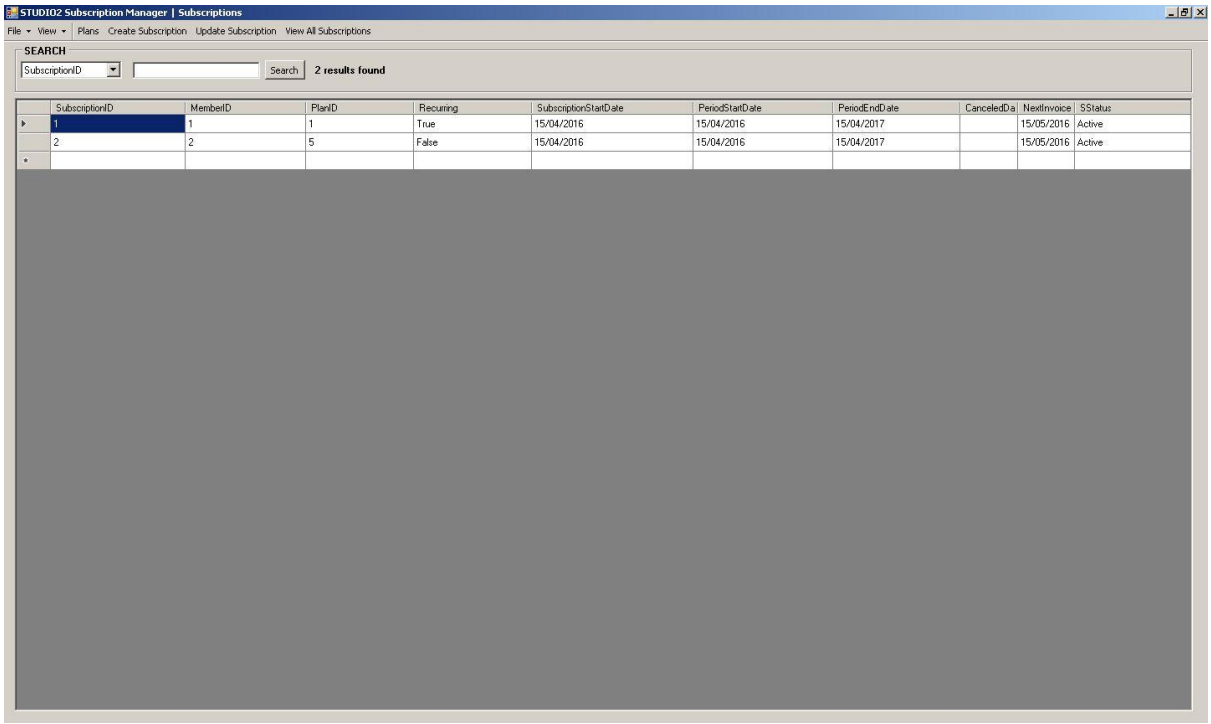


Test 2-28: Resize form when grpCreateMember is visible

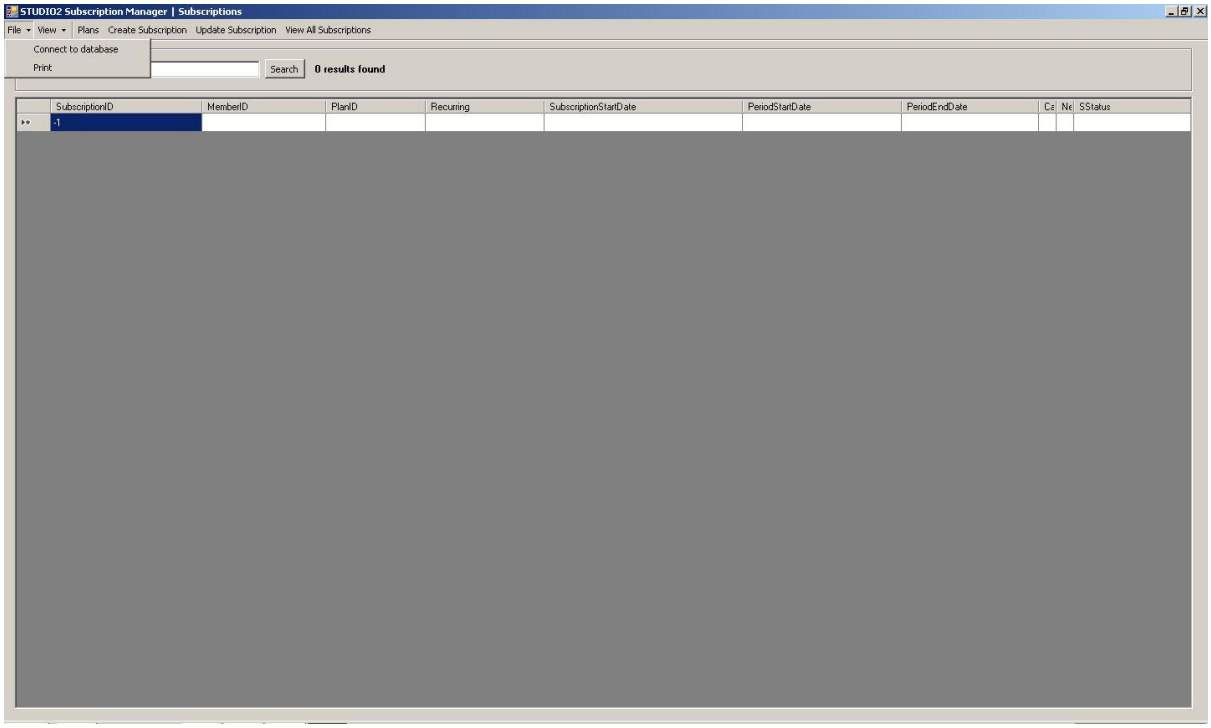


Test 2-28: Resize form when grpUpdateMember is visible

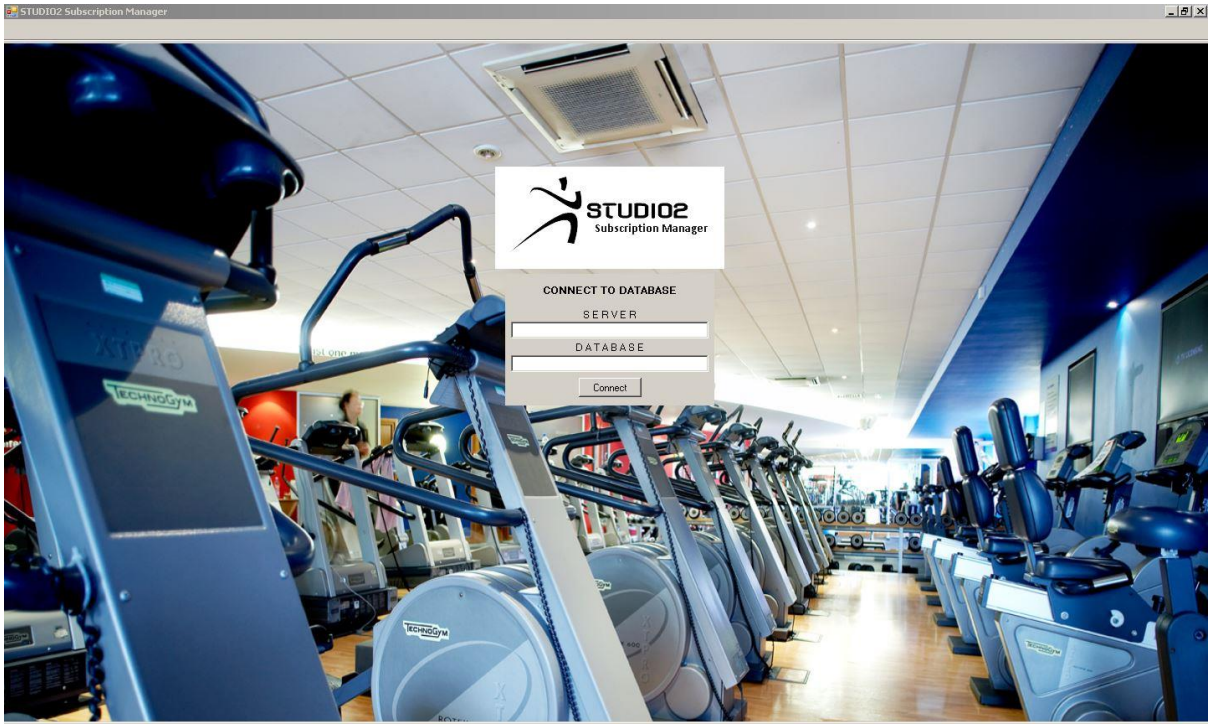
SUBSCRIPTIONS FORM



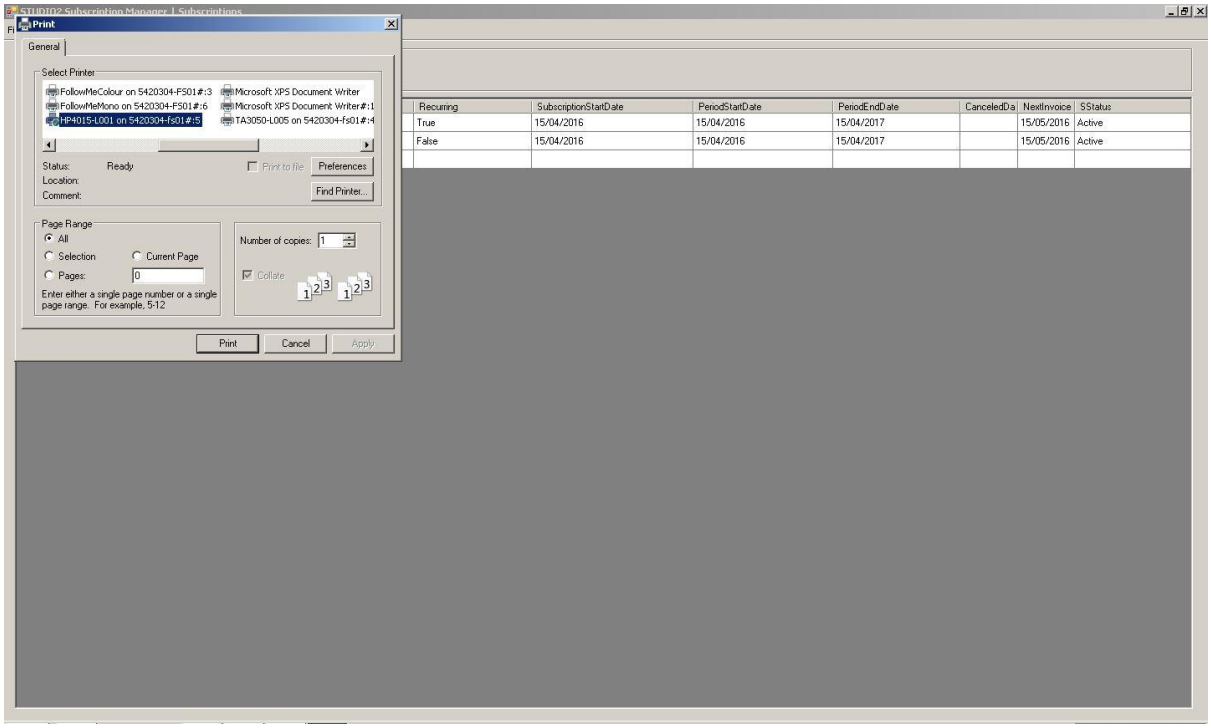
Test 3-1: Load 'Subscriptions' Form



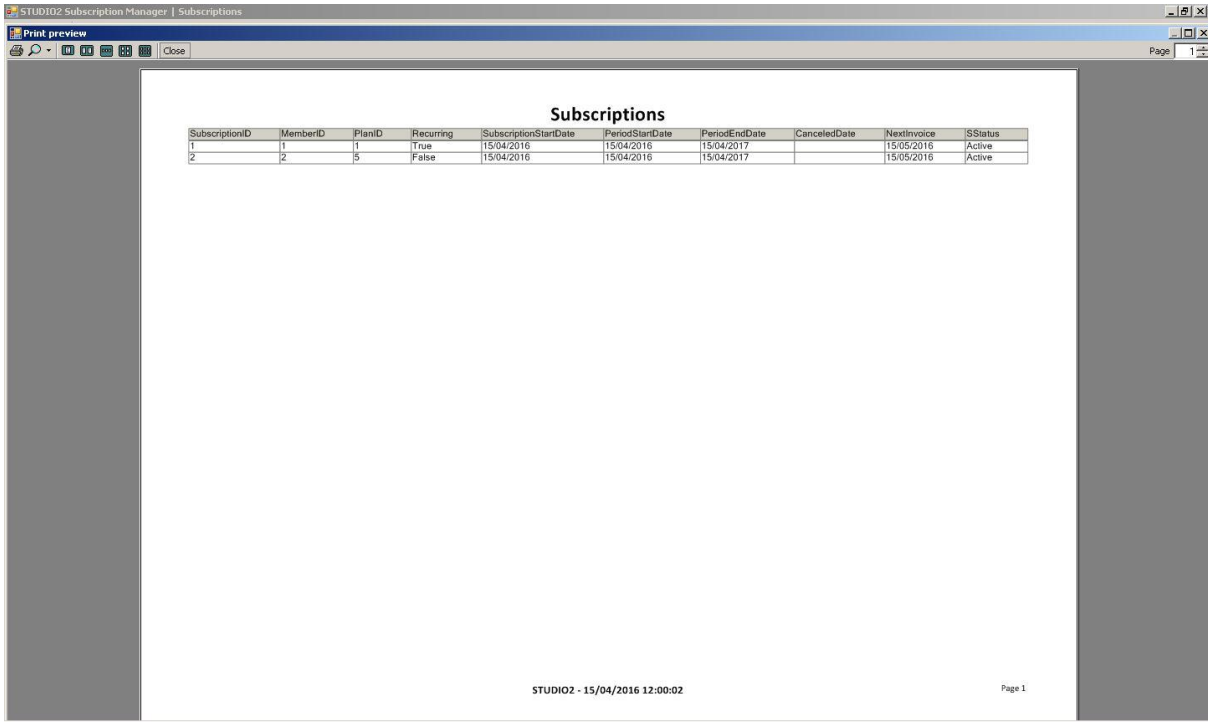
Test 3-2: Click 'File' ToolStripDropDownButton



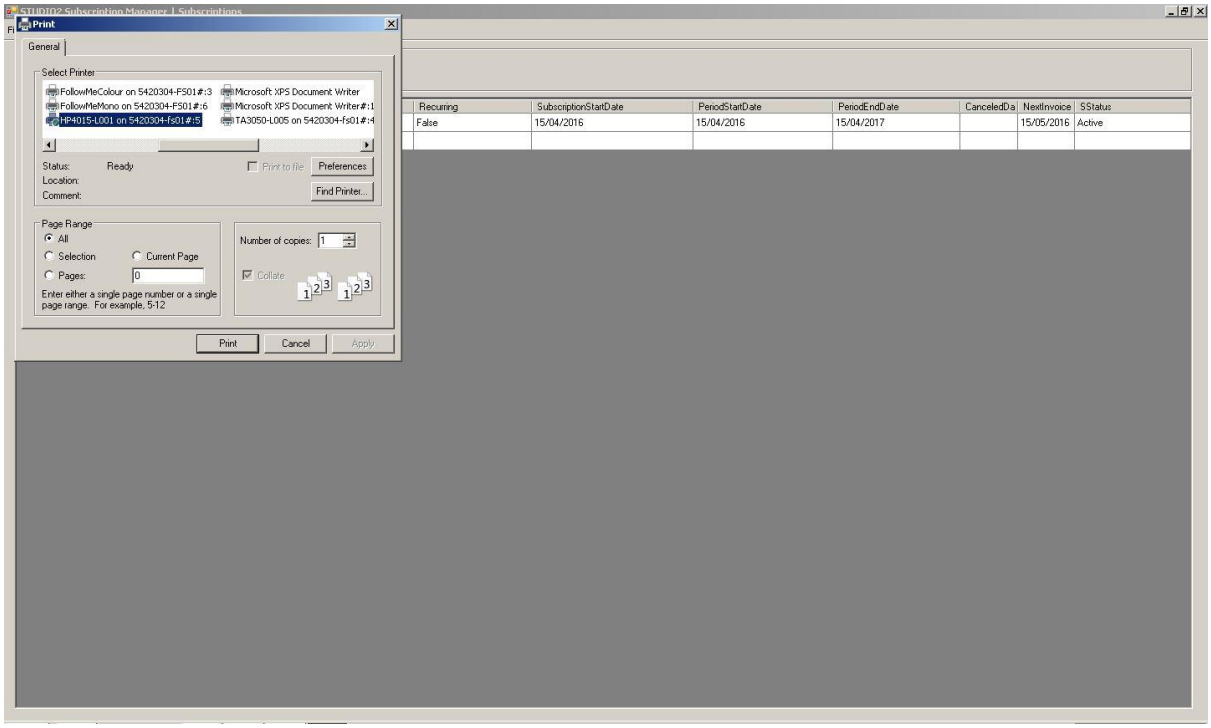
Test 3-3: Click 'Connect to database' ToolStripMenuItem



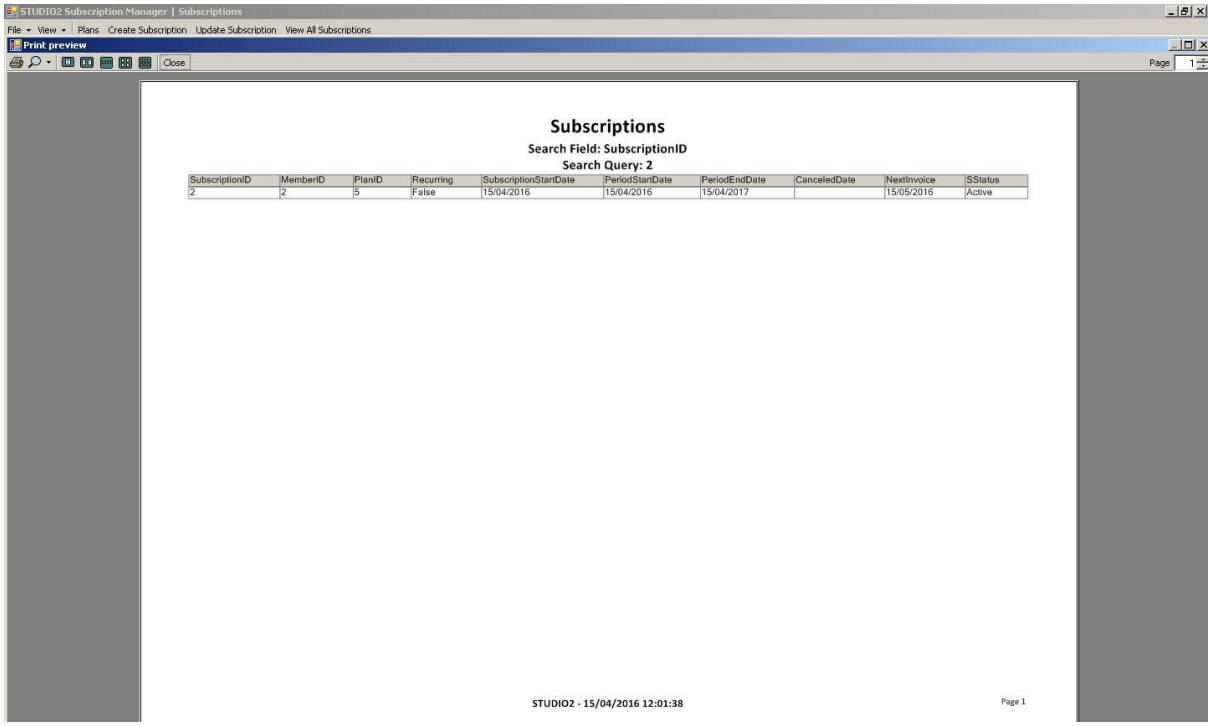
Test 3-4: Click 'Print' ToolStripMenuItem while dgvSQLOutput is visible



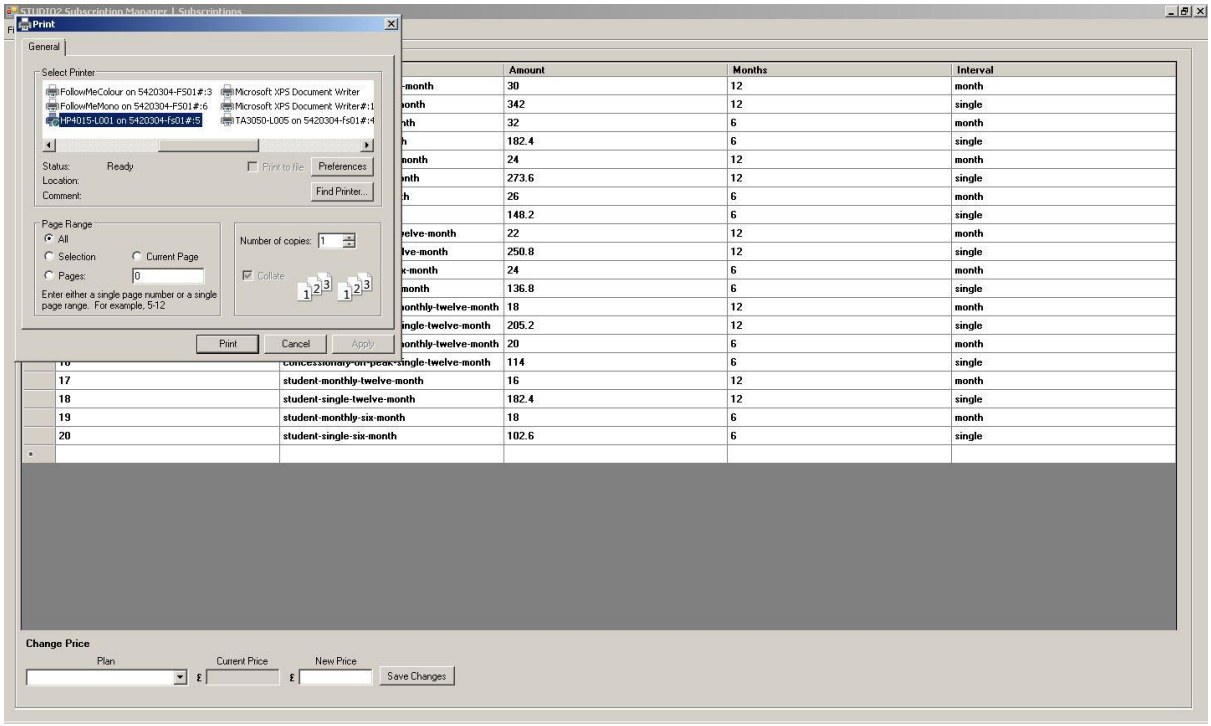
Test 3-4: Click 'Print' in Print Settings



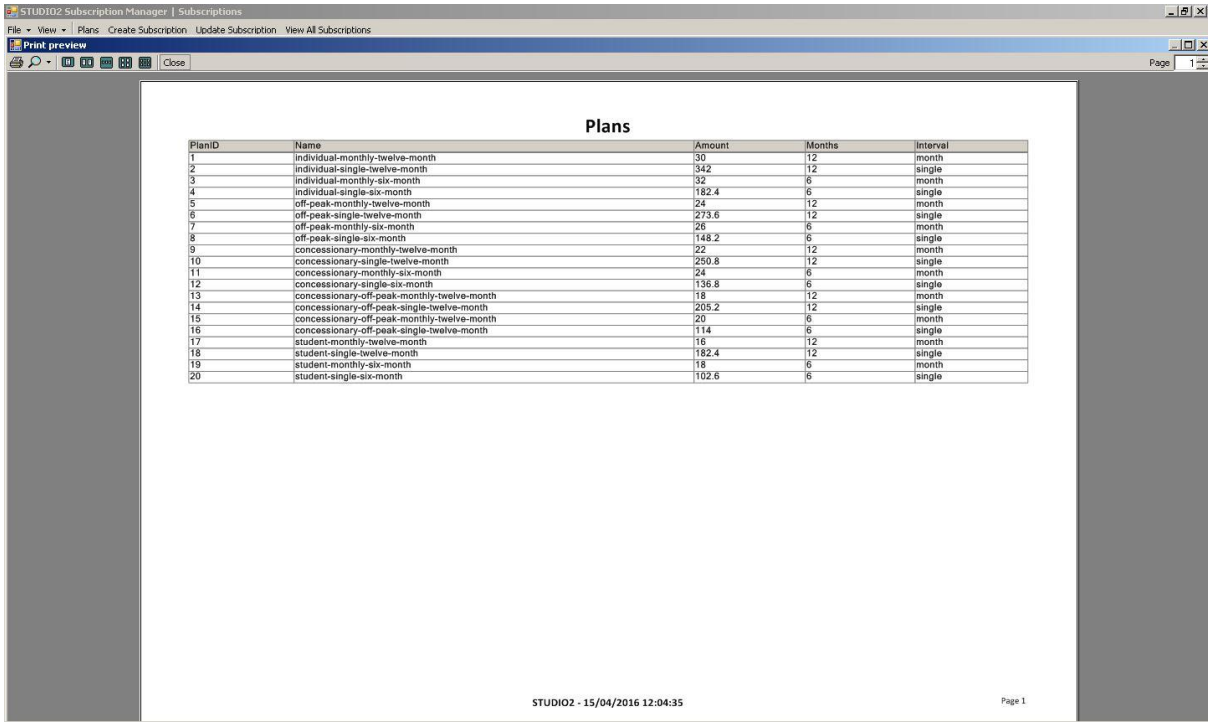
Test 3-5: Click 'Print' ToolStripMenuItem while dgvSQLOutput is visible after doing a search query



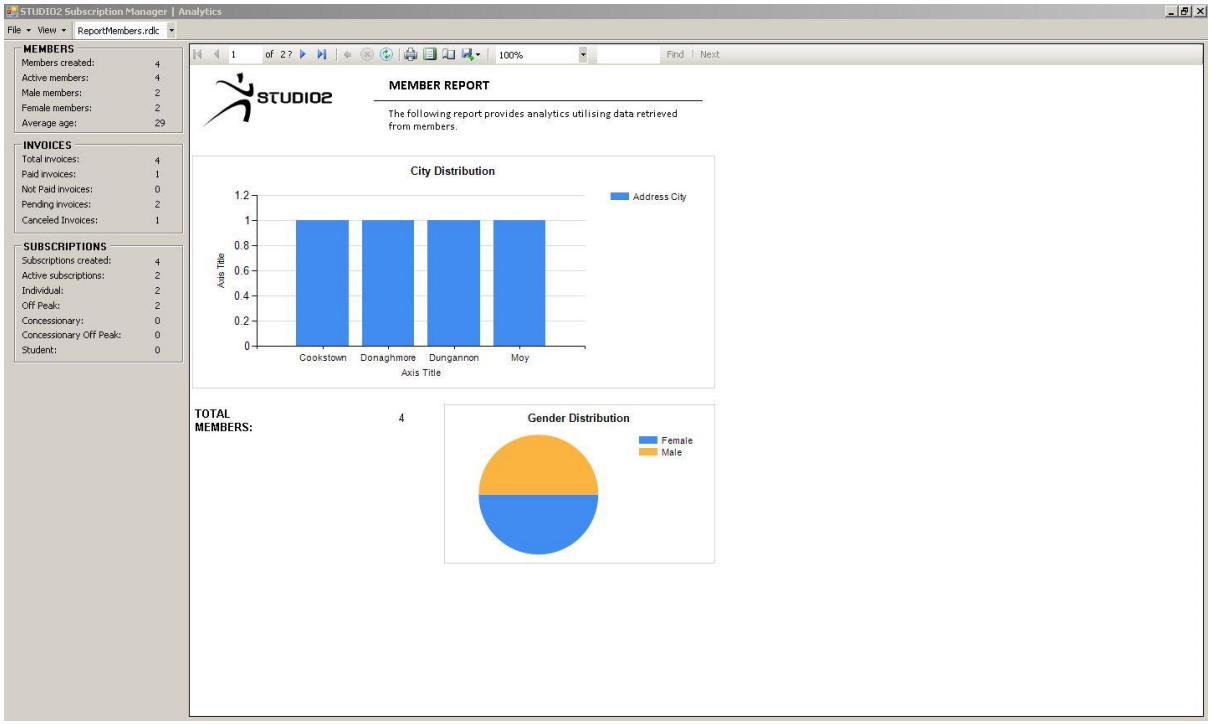
Test 3-5: Click 'Print' in Print Settings



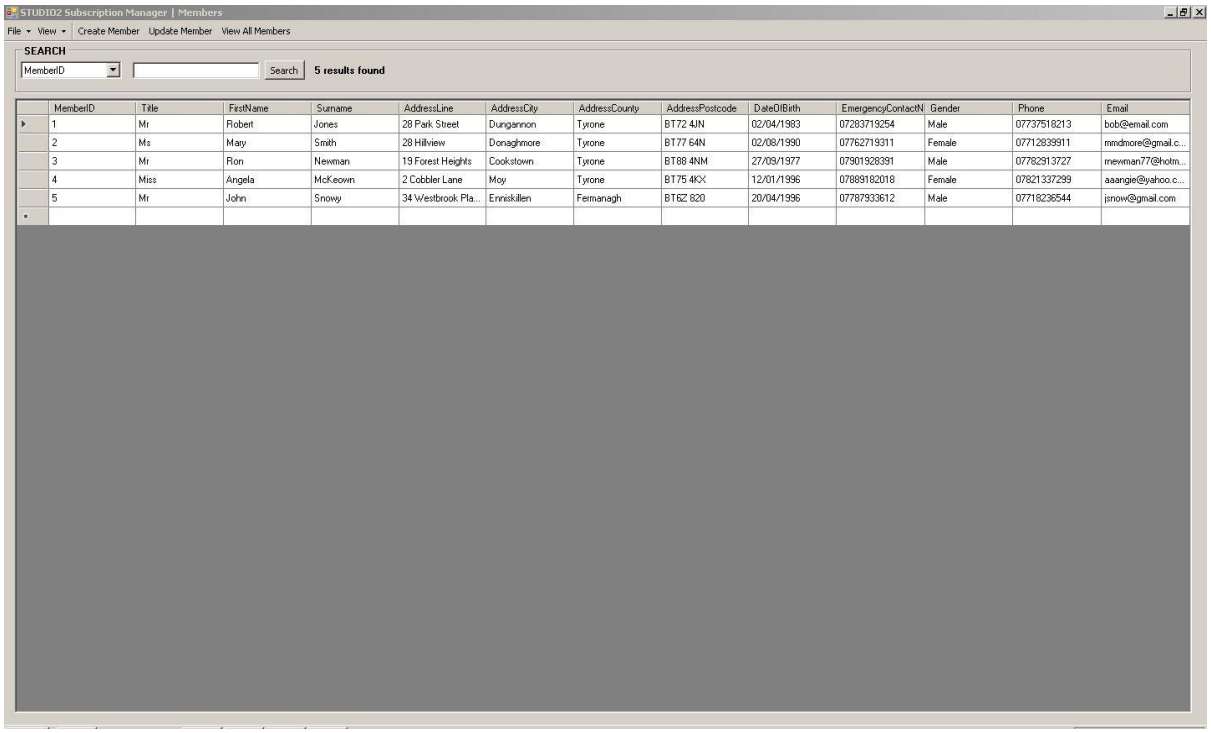
Test 3-6: Click 'Print' ToolStripMenuItem while dgvPlans is visible



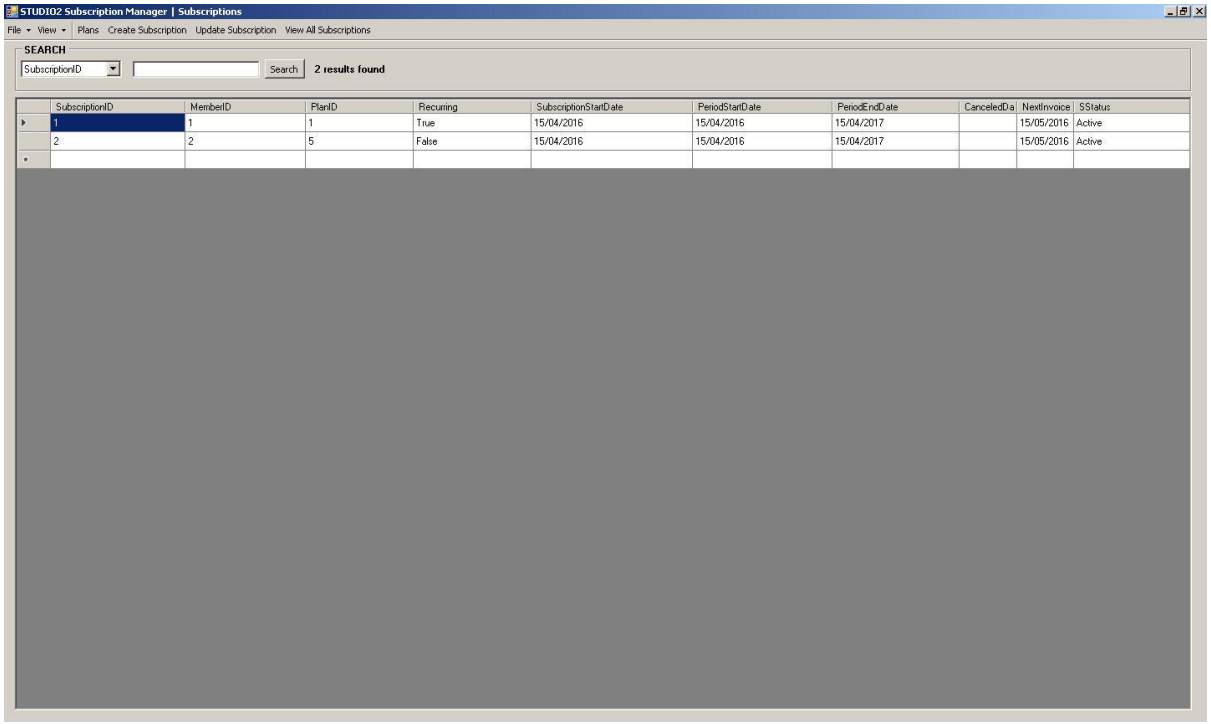
Test 3-3: Click 'Print' in Print Settings



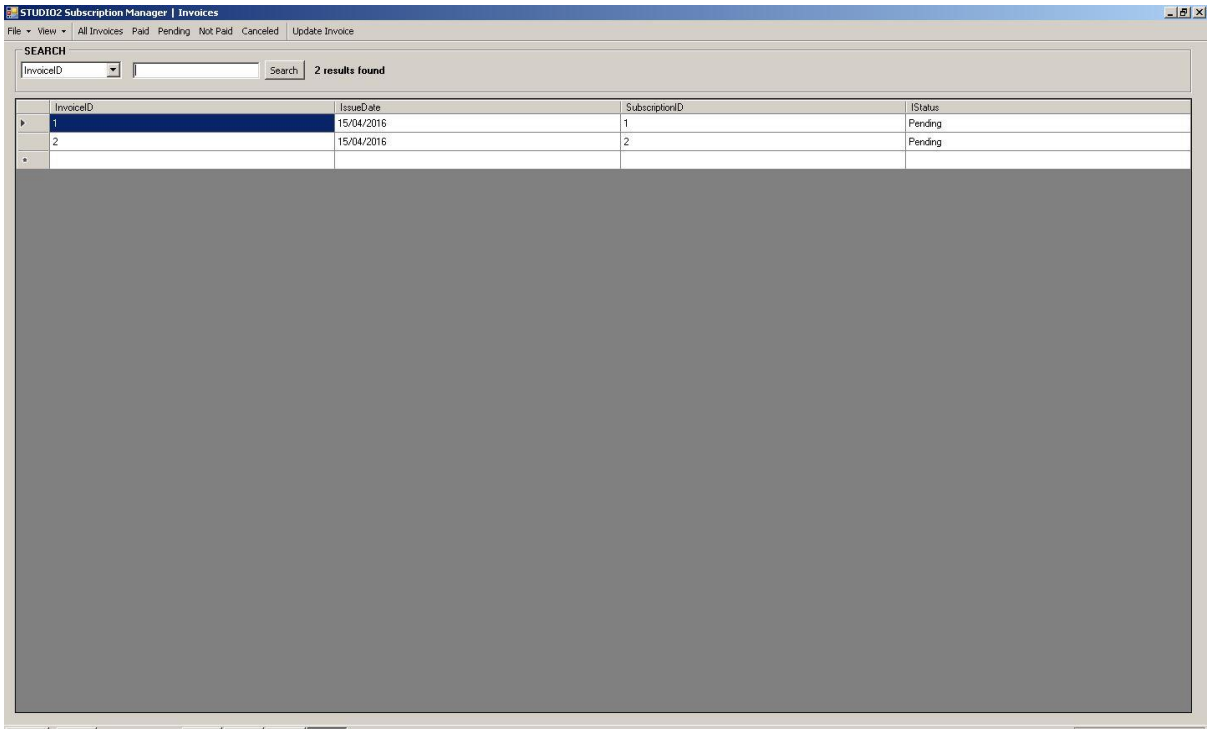
Test 3-7: Click 'Analytics' ToolStripMenuItem



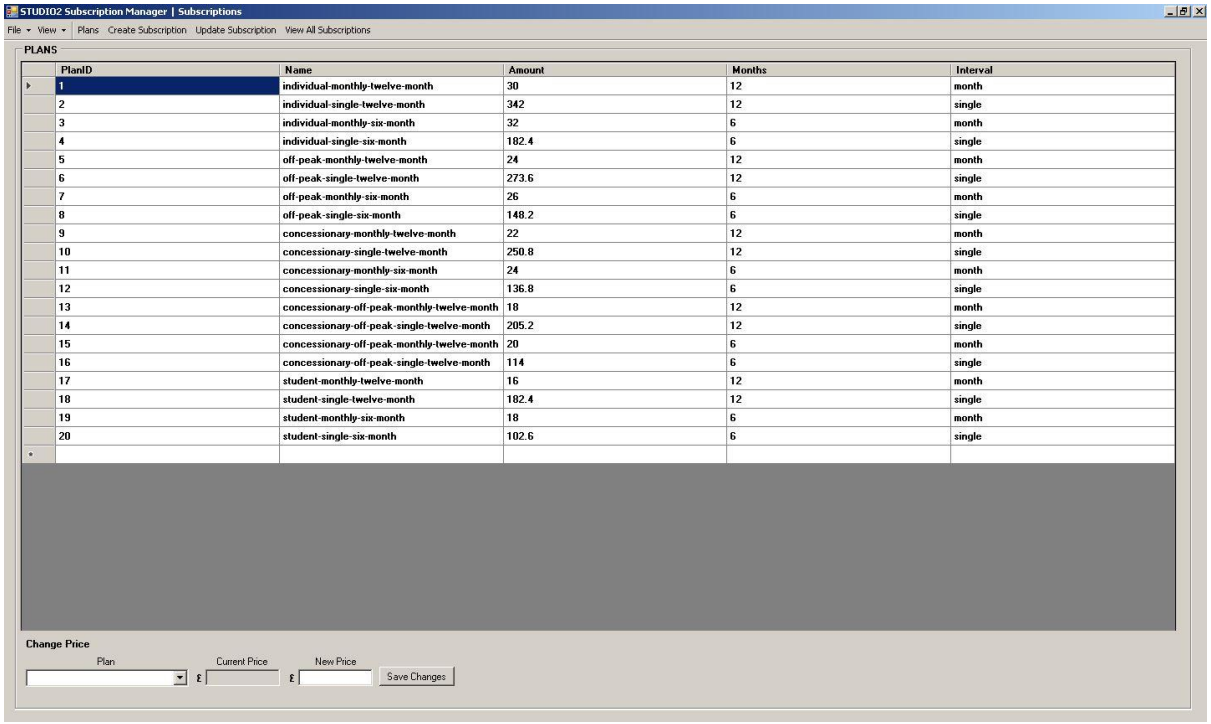
Test 3-8: Click 'Members' ToolStripMenuItem



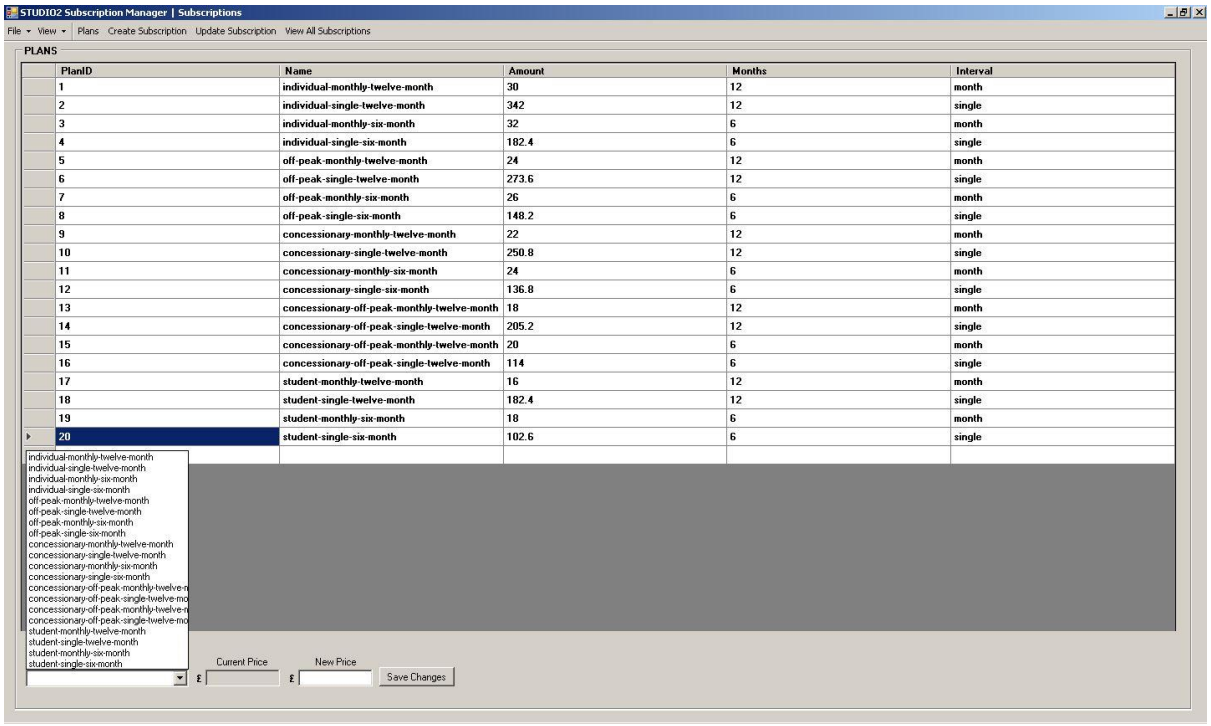
Test 3-9: Click 'Subscriptions' ToolStripMenuItem



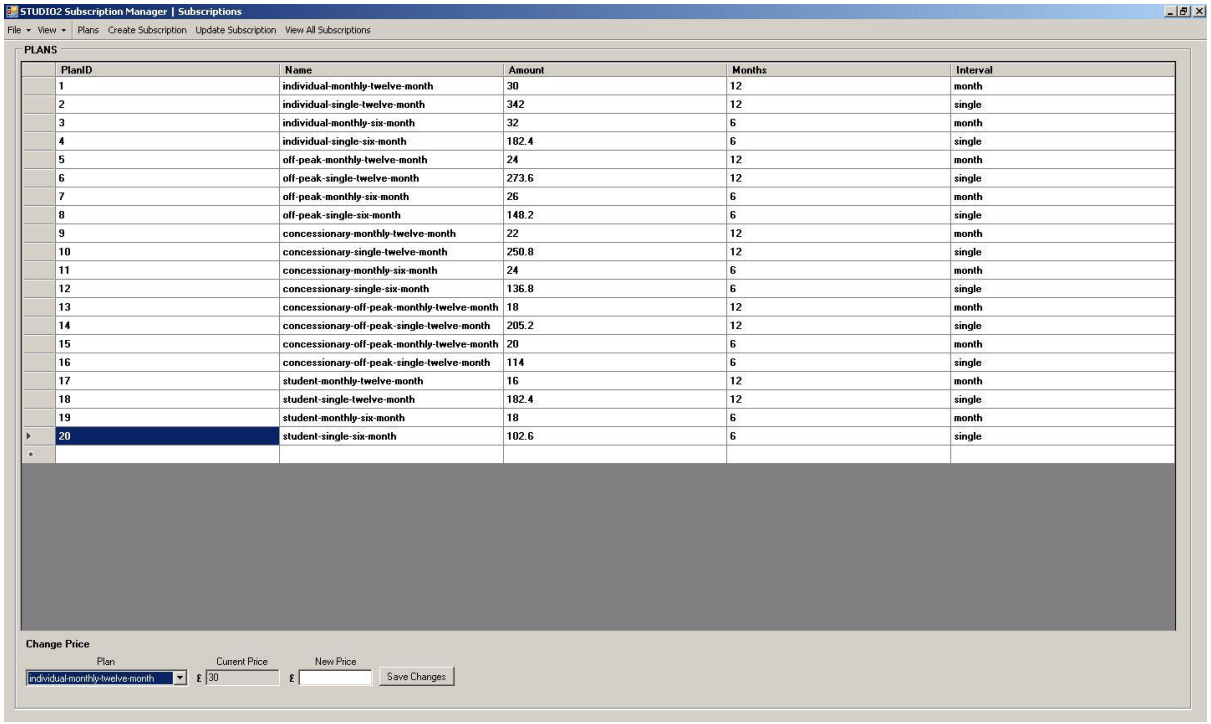
Test 3-10: Click 'Invoices' ToolStripMenuItem



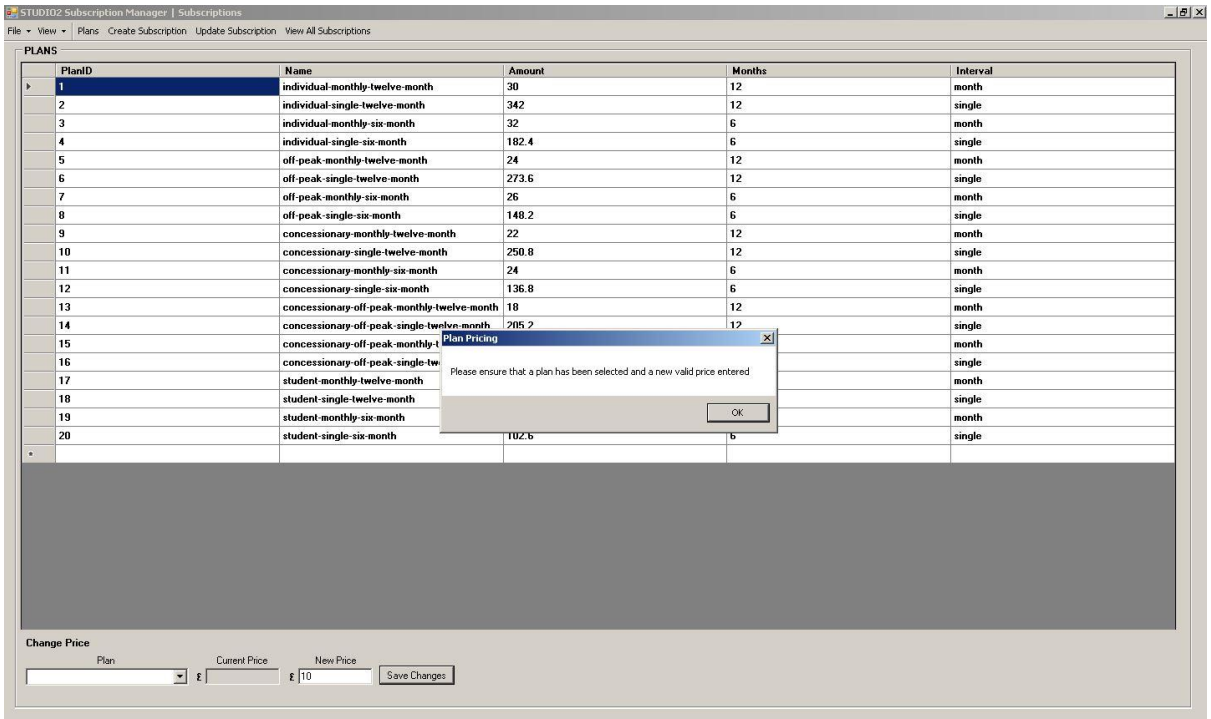
Test 3-11: Click 'Plans' ToolStripButton



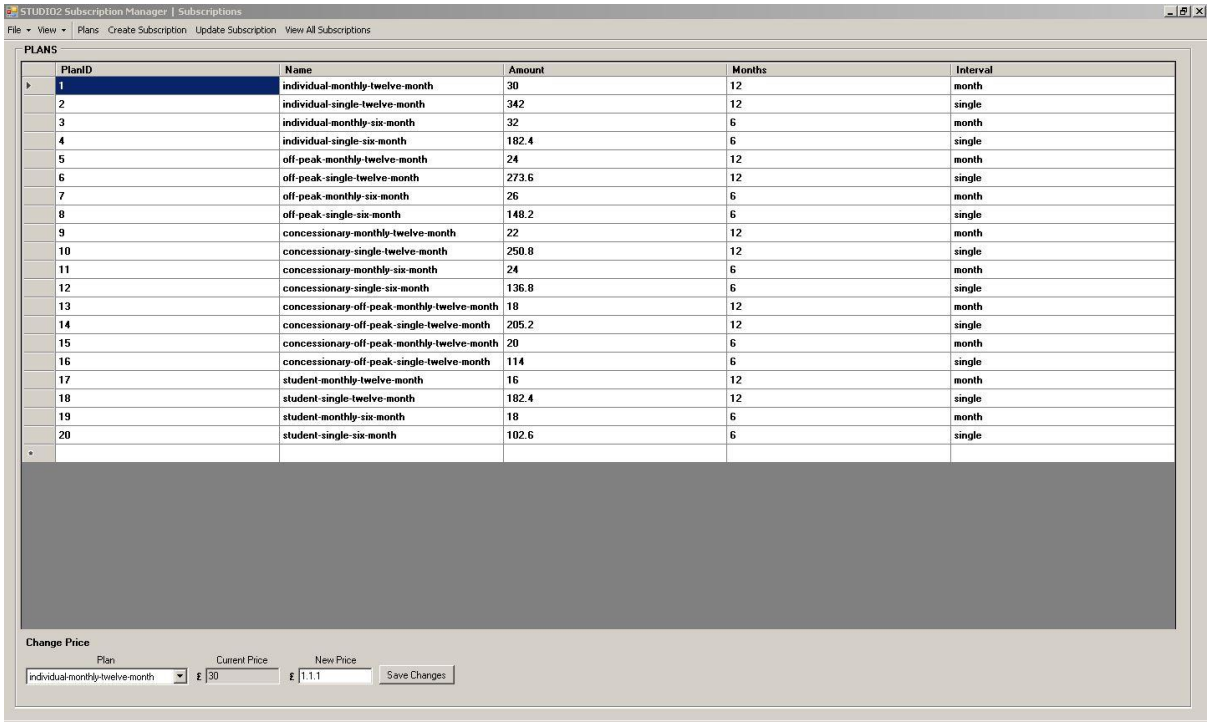
Test 3-12: Click cboPlans ComboBox



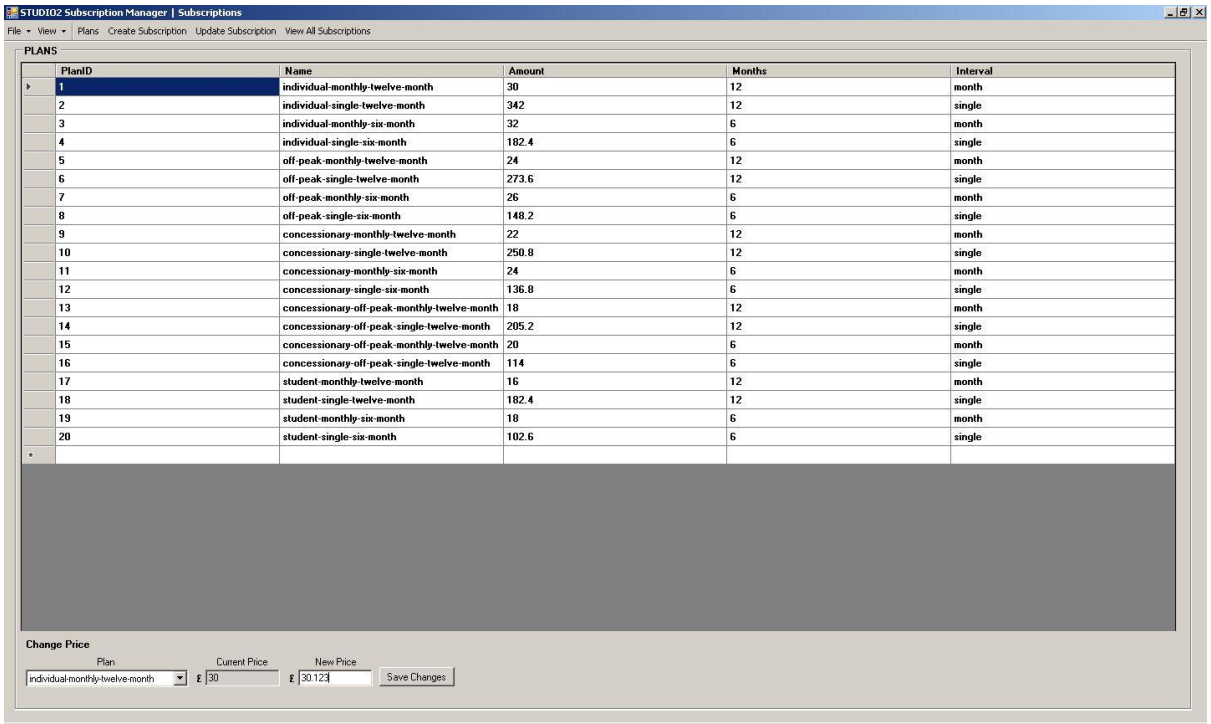
Test 3-13: Click value from cboPlans DropDownList



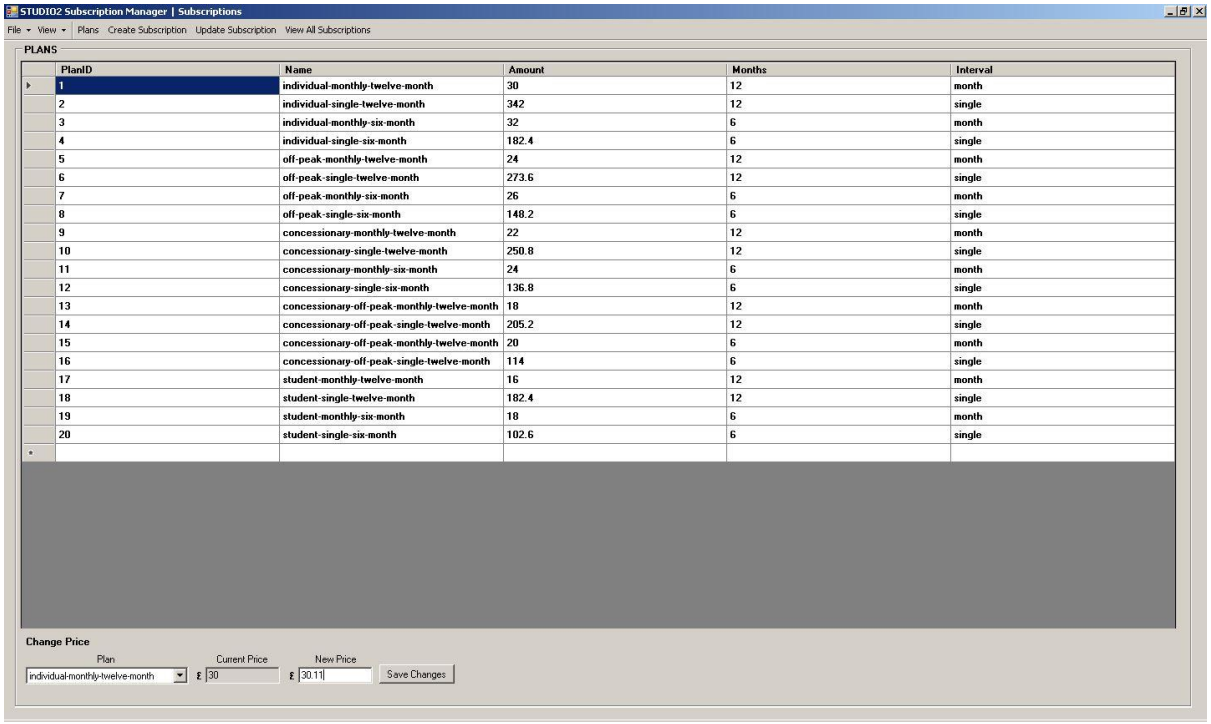
Test 3-14: Click 'Save Changes' Button when no value have been entered in txtNewPrice TextBox or cboPlansComboBox



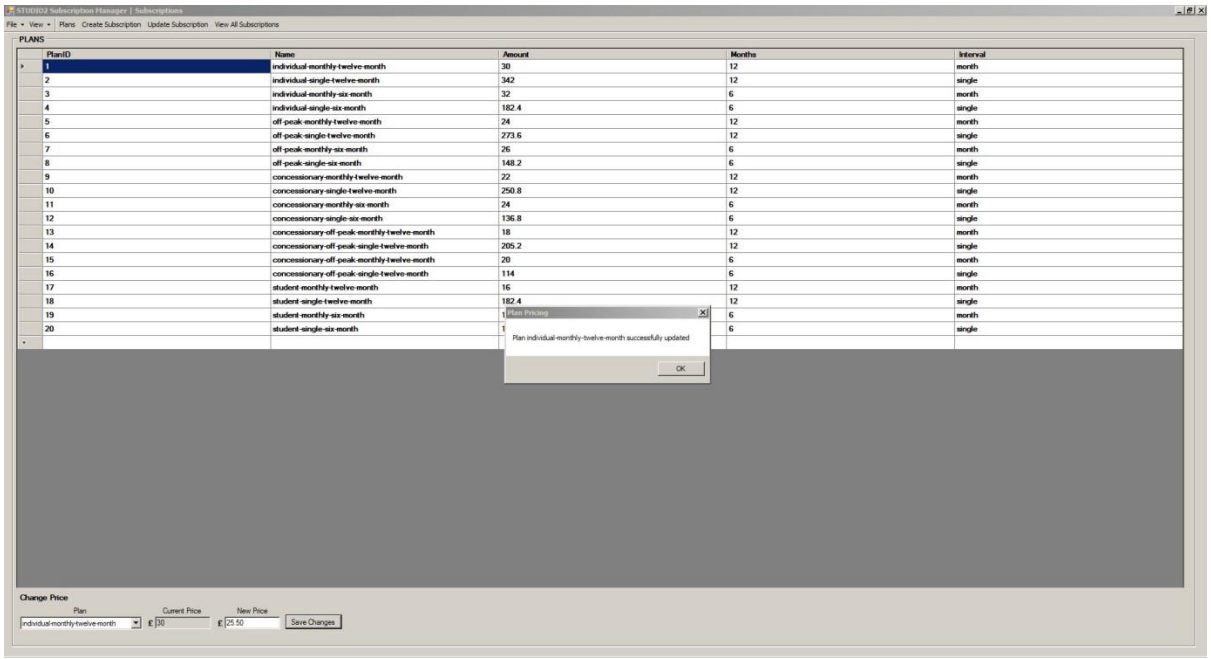
Test 3-15: Enter "1.1.1" into txtNewPrice



Test 3-15: Enter "3.123" into txtNewPrice

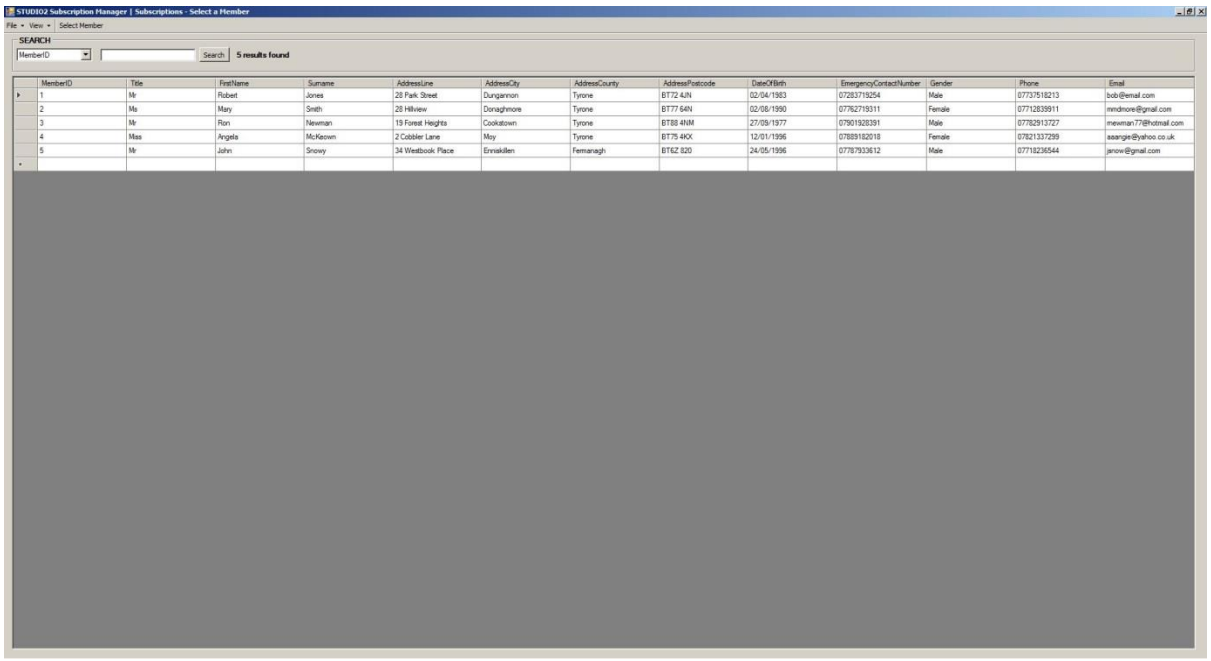


Test 3-15: Corrective action taken. Now unable to type invalid values such as "1.1.1" or "30.123"

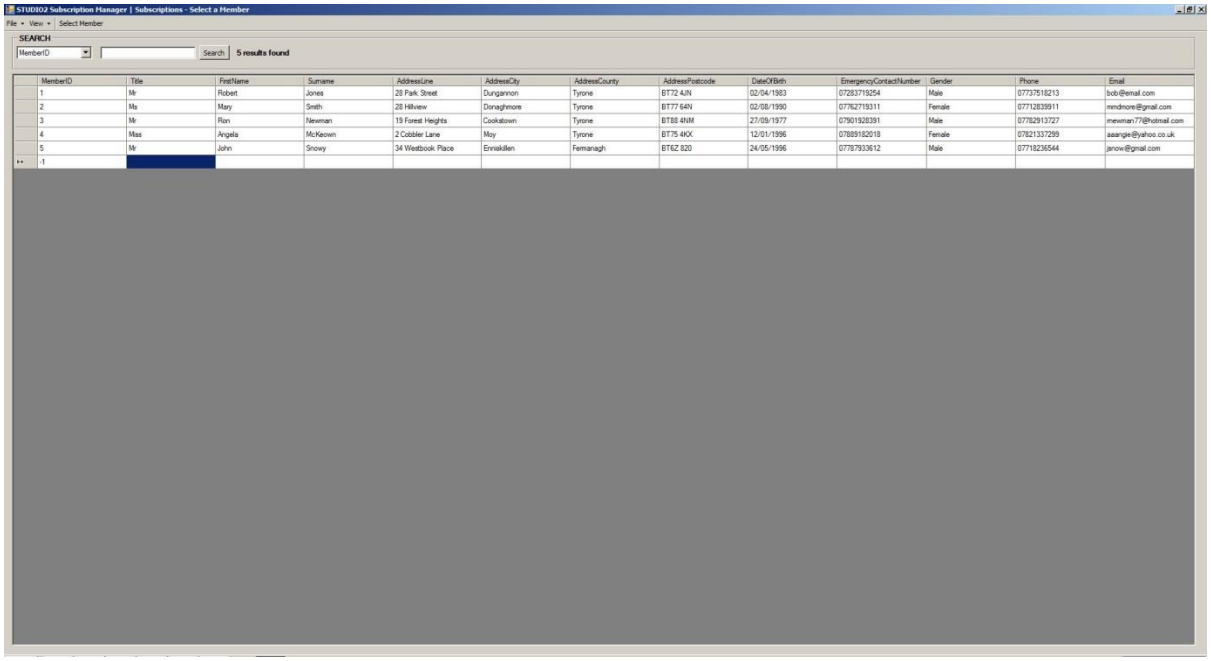


Test 3-16: Click 'Save Changes' Button when a valid value has been entered in txtNewPrice TextBox and cboPlansComboBox

Test 3-17: Click 'Create Subscription' ToolStripButton

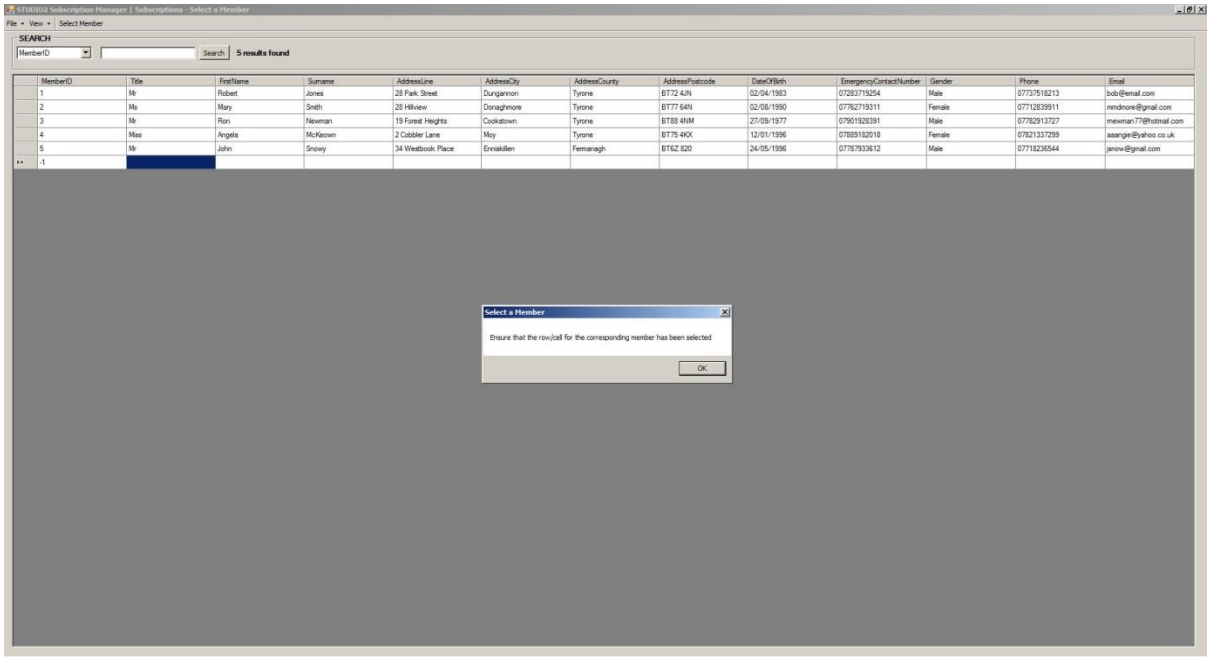


Test 3-18: Click 'Search' button

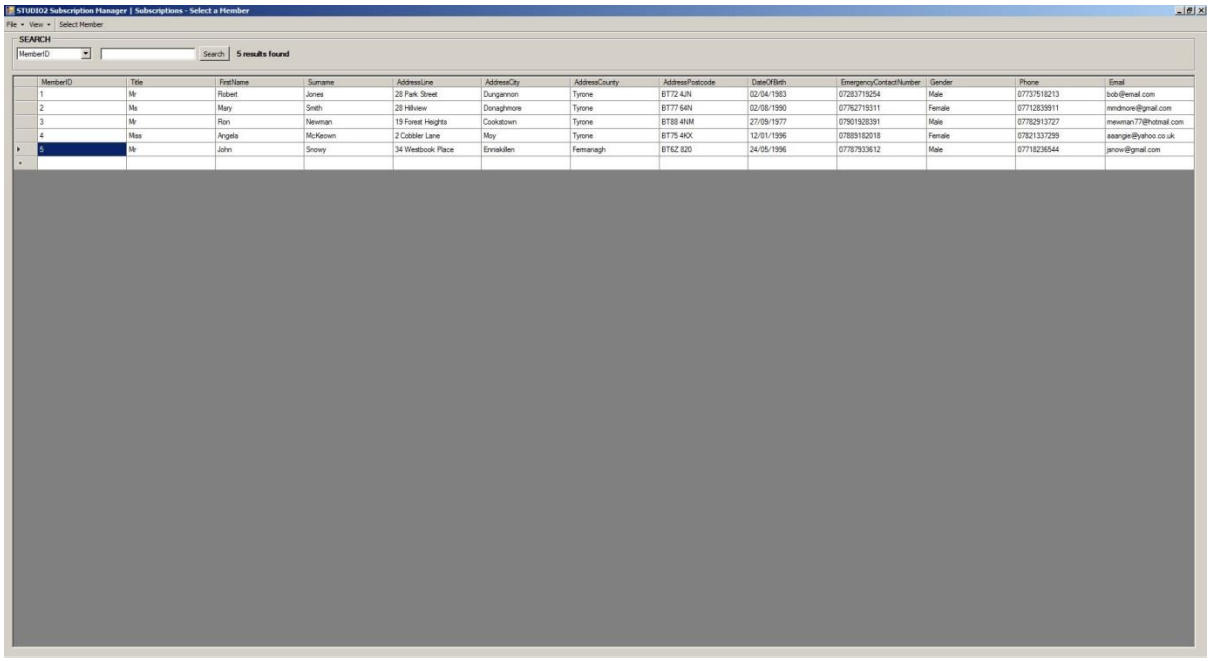


Test 3-19: Click 'Select Member' ToolStripButton when an invalid cell/row has been selected

Test 3-19: MemberID of selected member displays in Member ID TextBox as opposed to a MessageBox being displayed



Test 3-19: Corrective action taken. MessageBox now displays as intended



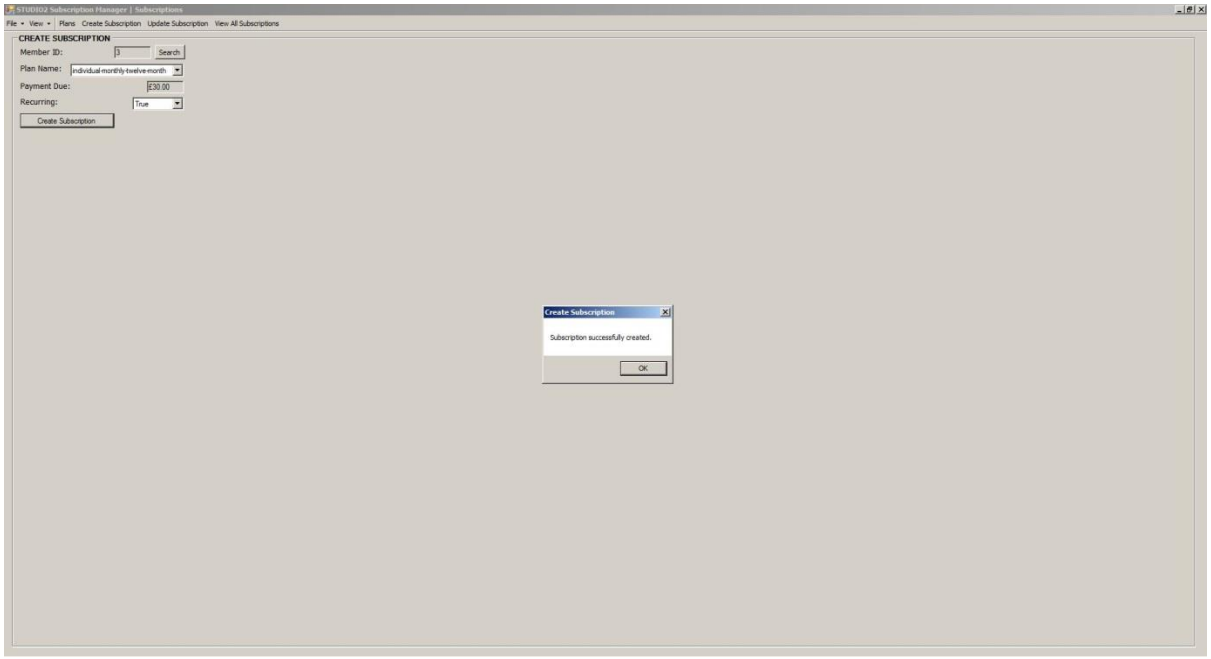
Test 3-20: Click 'Select Member' ToolStripButton when a valid cell/row has been selected

Test 3-20: MemberID displays in Member ID TextBox

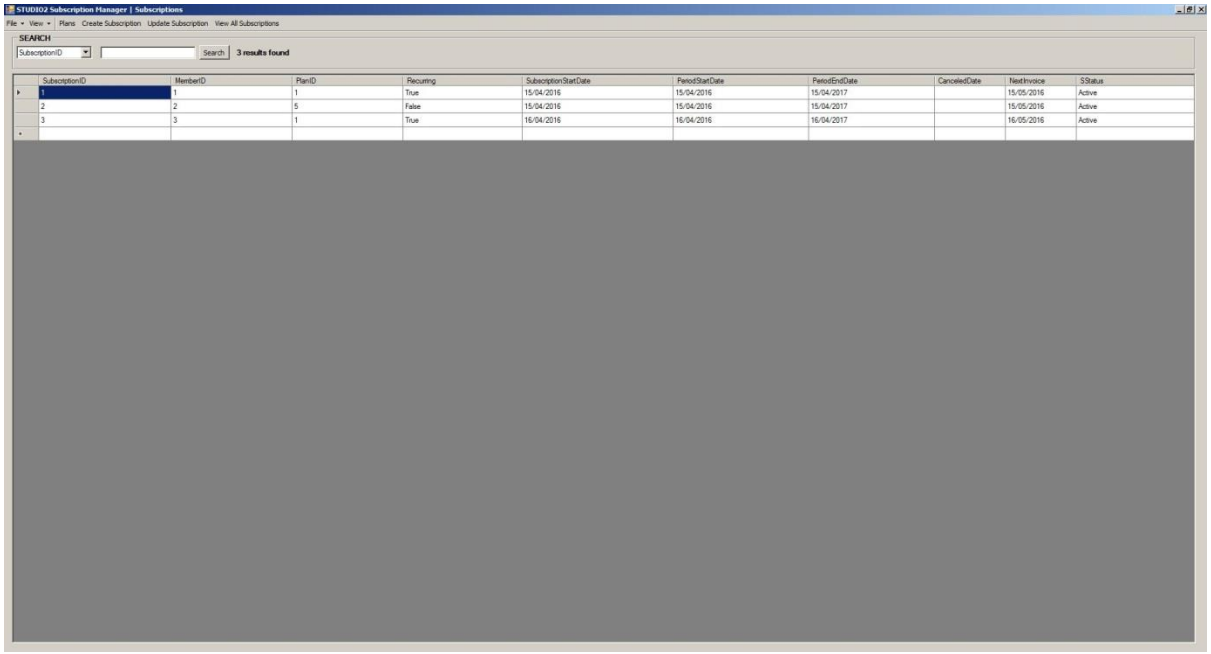
Test 3-21: Click cboCreatePlans ComboBox

Test 3-22: Click value from cboCreatePlans DropDownList

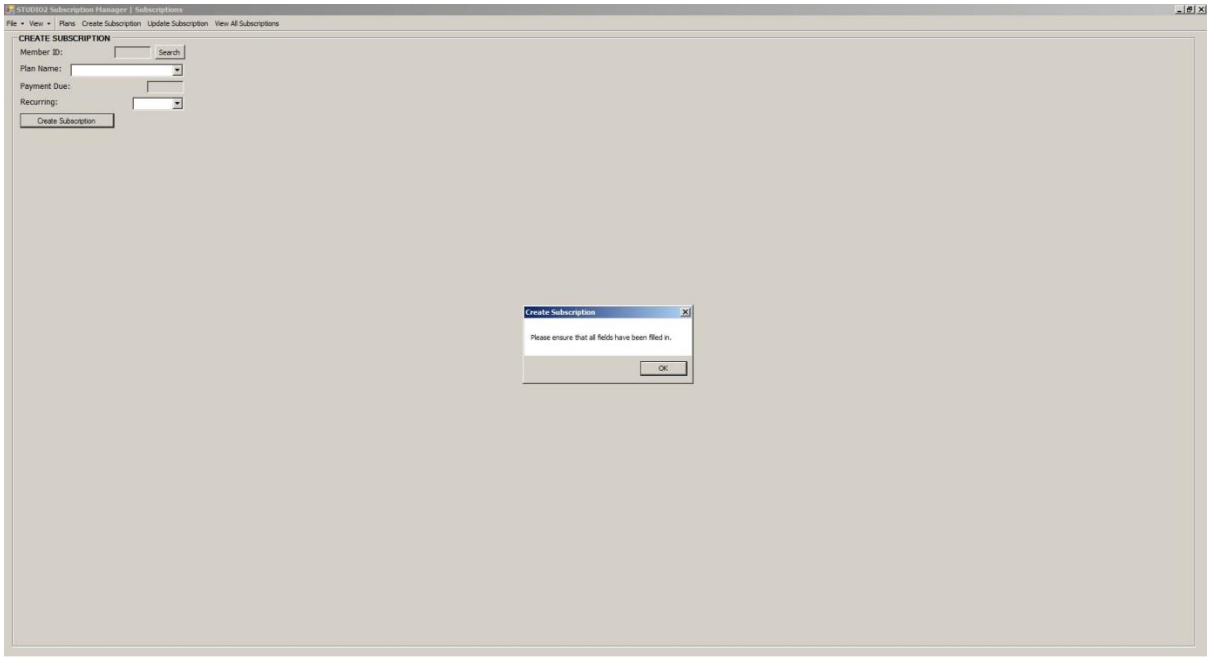
Test 3-23: Click cboCreateRecurring ComboBox



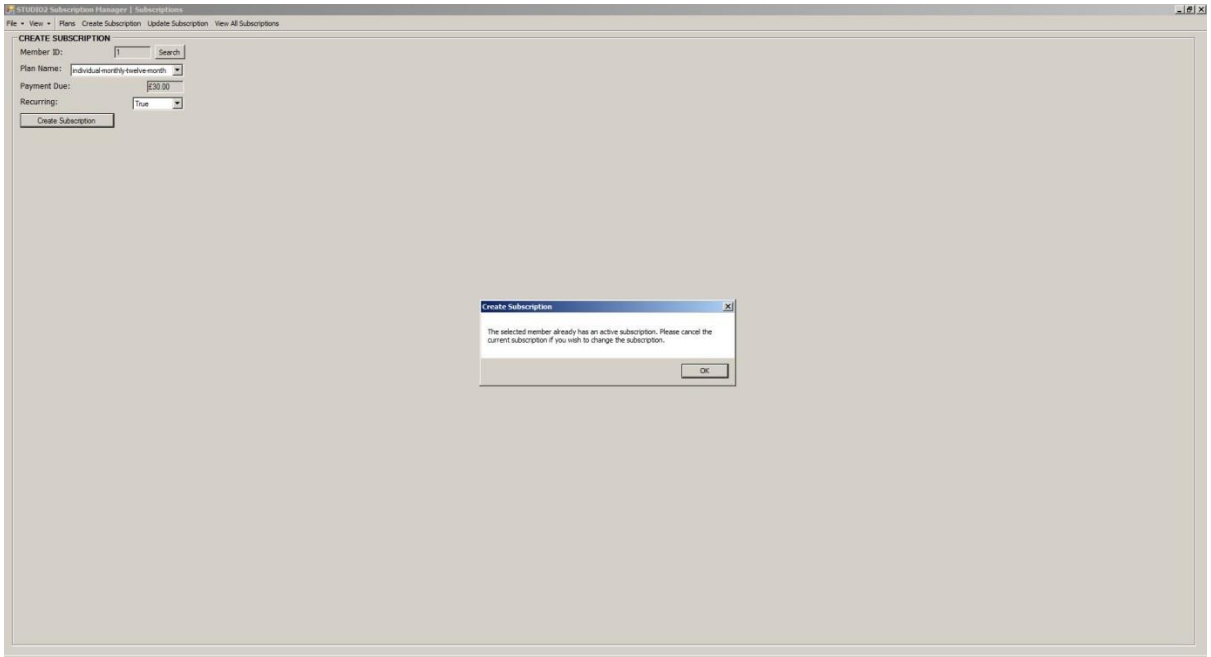
Test 3-24: Click 'Create Subscription' Button when txtCreateMemberID, cboCreatePlans and cboCreateRecurring Text properties all contain valid values ('3', 'individual-monthly-twelve-month', 'True' respectively). Current date is 16/04/2016



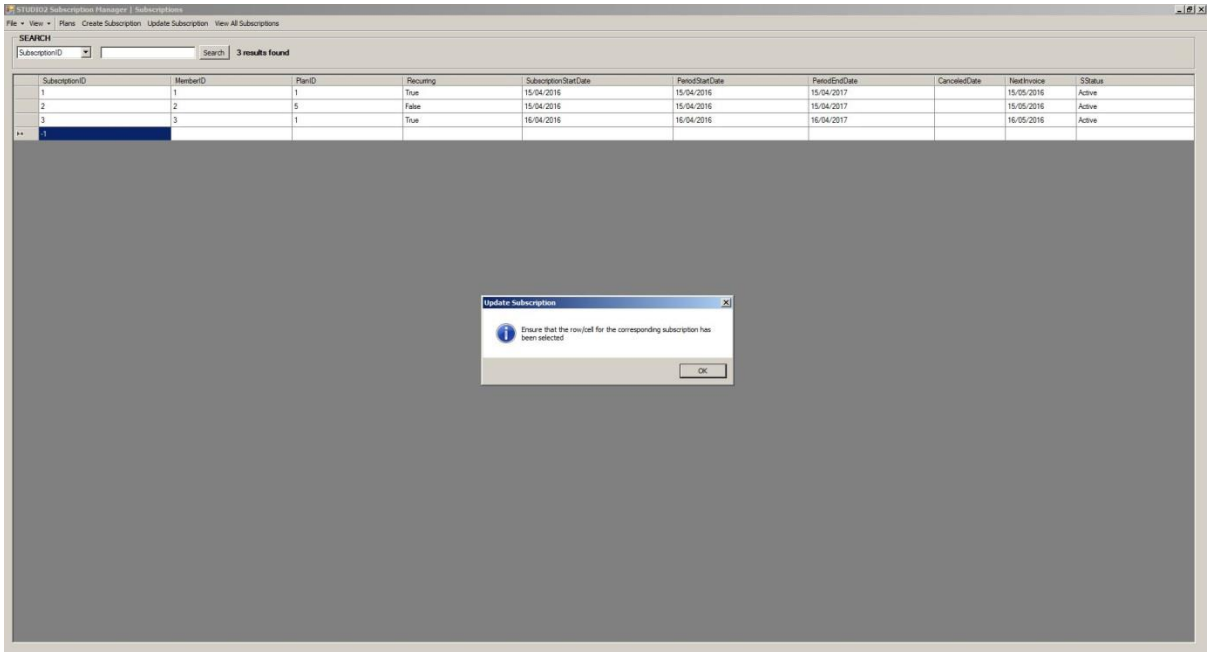
Test 3-24: New subscription displayed in dgvSQLOutput



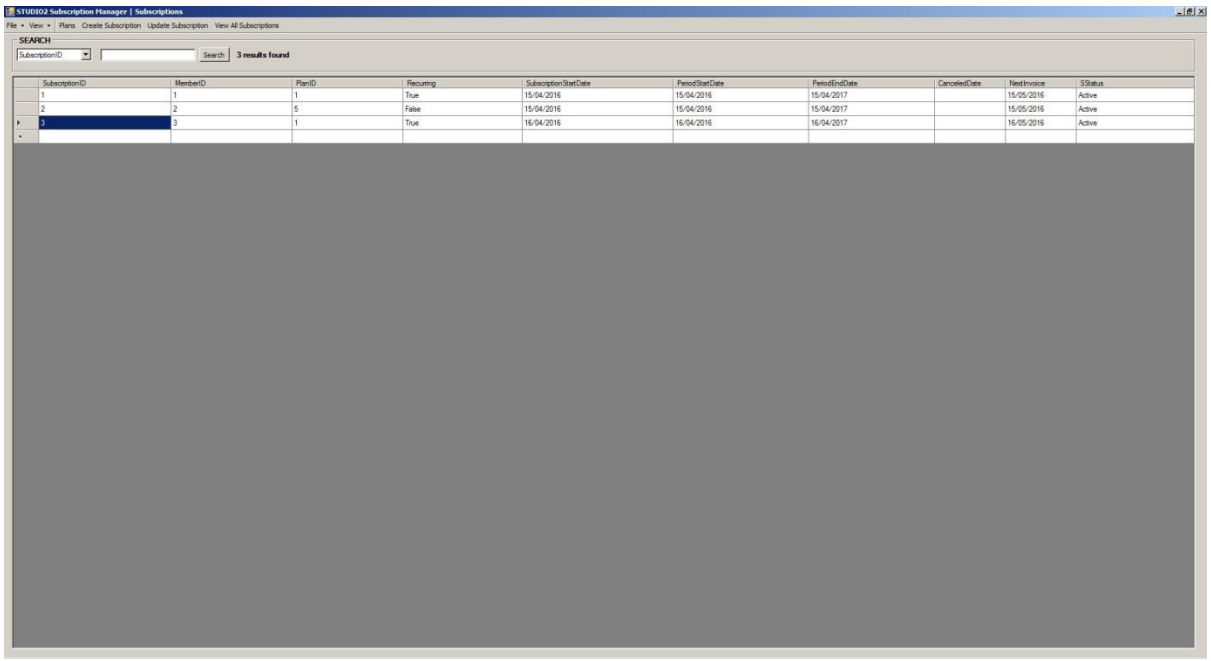
Test 3-25: Click 'Create Subscription' Button when txtCreateMemberID, cboCreatePlans and cboCreateRecurring Text properties contain no values



Test 3-26: Click 'Create Subscription' Button when txtCreateMemberID, cboCreatePlans and cboCreateRecurring Text properties all contain valid values ('1', 'individual-monthly-twelve-month', 'True' respectively) but MemberID = 1 is already assigned an active subscription

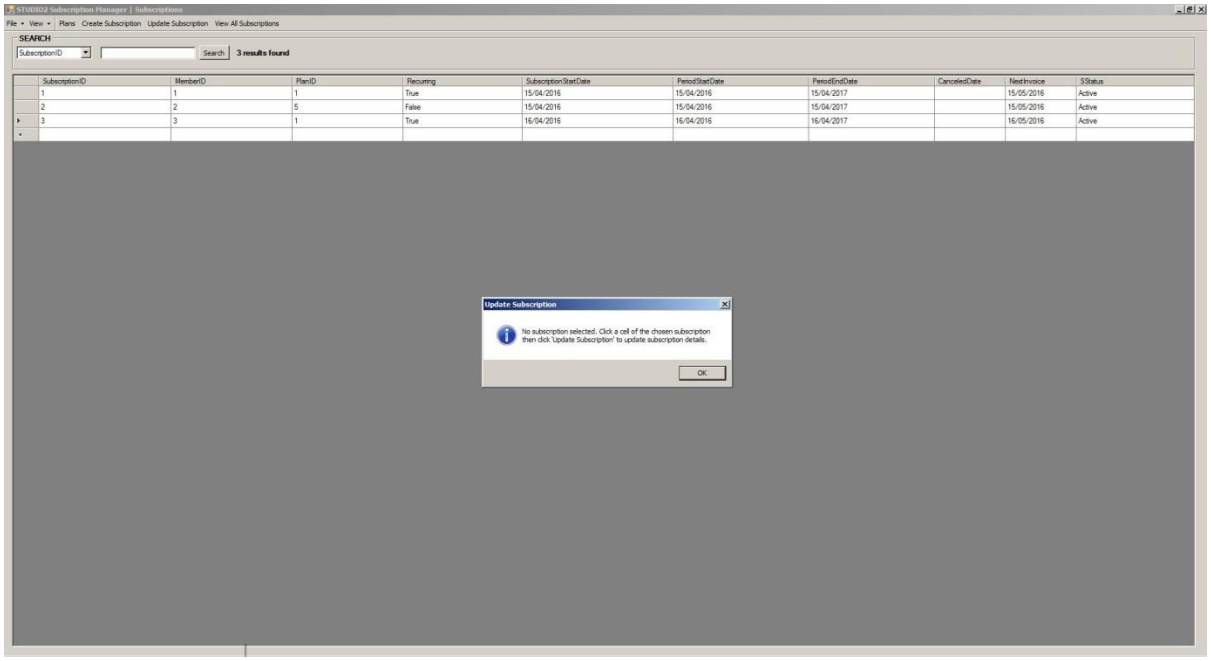


Test 3-27: Click 'Update Subscription' ToolStripButton when an invalid row/cell has been selected



Test 3-28: Click 'Update Subscription' ToolStripButton when a valid row/cell has been selected

Test 3-28: Data corresponding to selected subscription filled into each appropriate TextBox and ComboBox

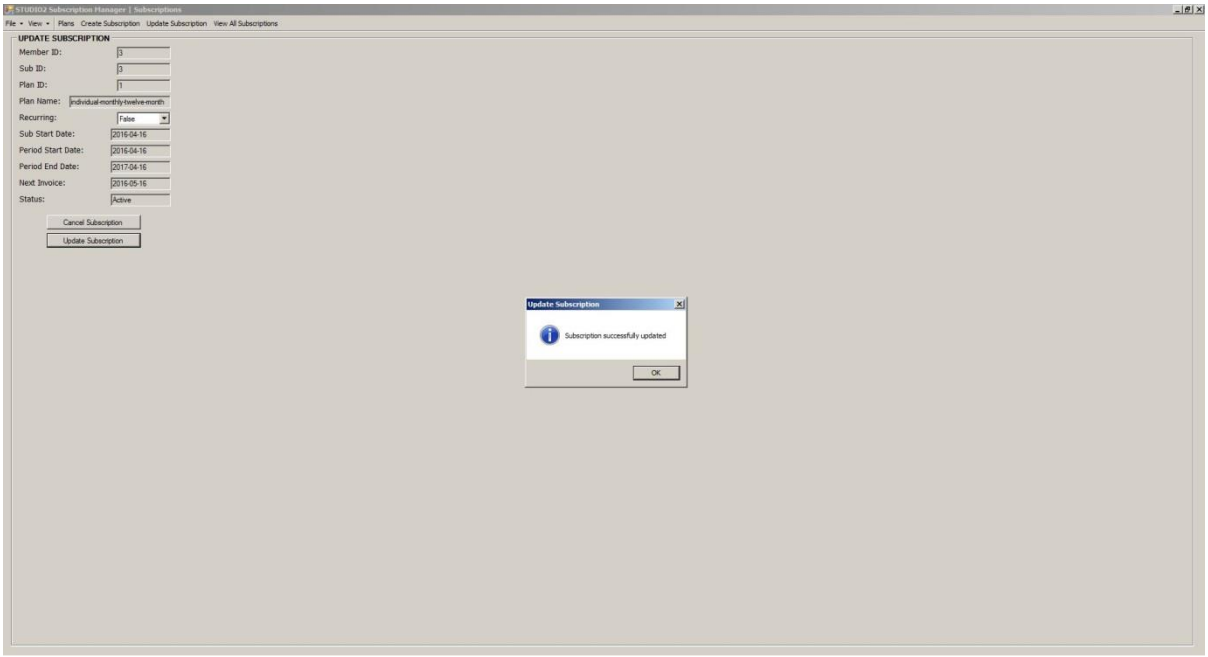


Test 3-29: Click 'Update Subscription' ToolStripButton when 'Create Subscription' GroupBox or 'Update Subscription' GroupBox is Visible

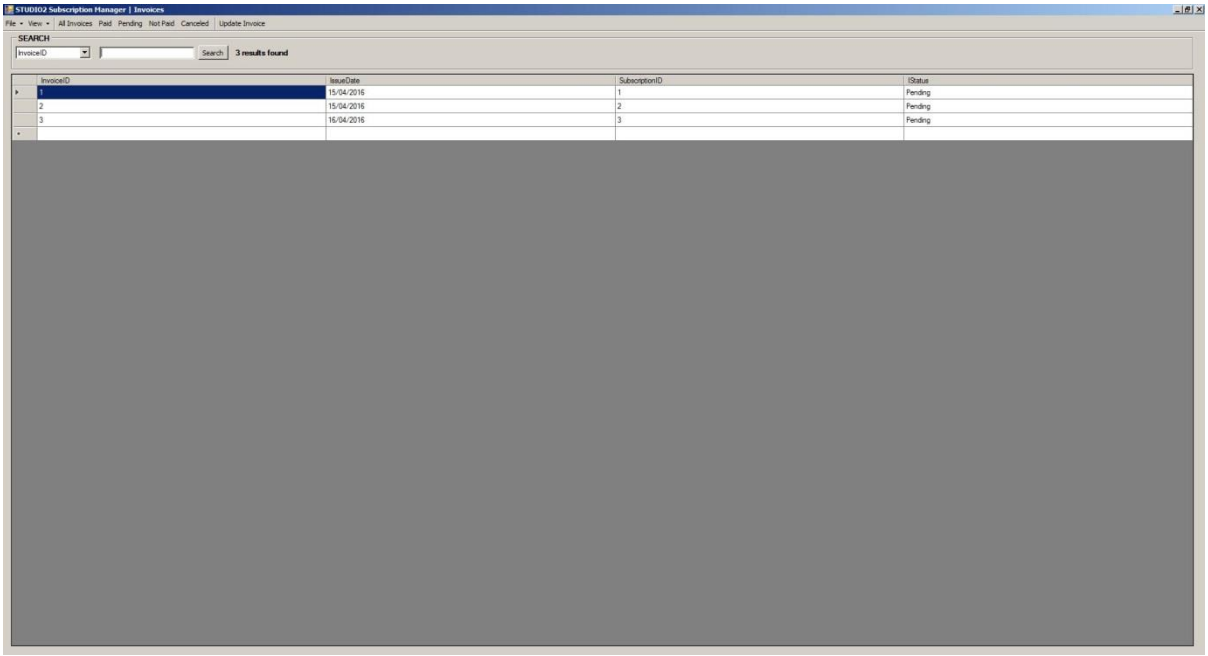
Test 3-30: Click 'Recurring' ComboBox

Test 3-31: Select the other value in 'Recurring' ComboBox

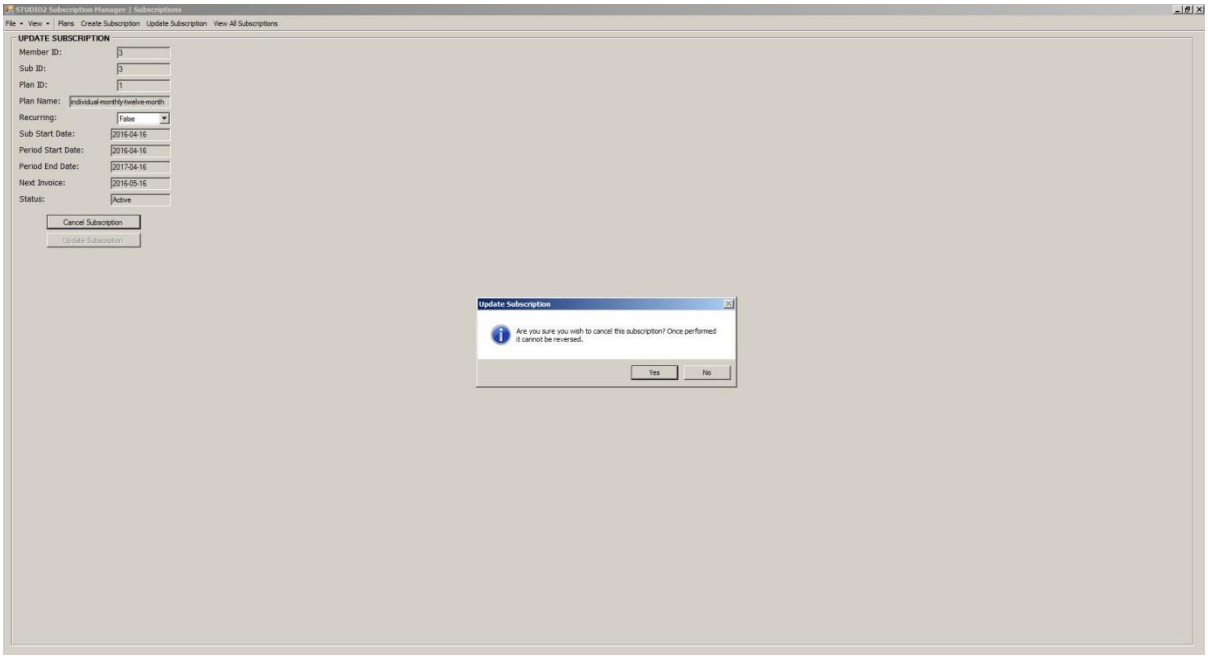
Test 3-31: Select original value again



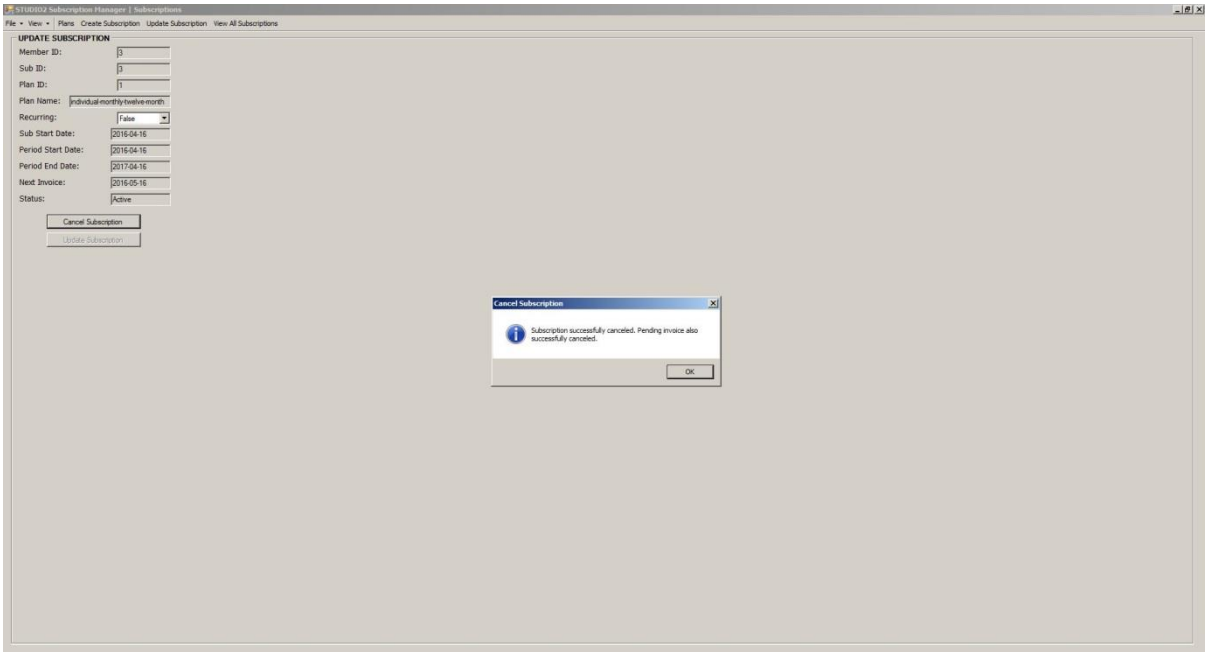
Test 3-32: Change 'Recurring' ComboBox value then click 'Update Subscription' Button



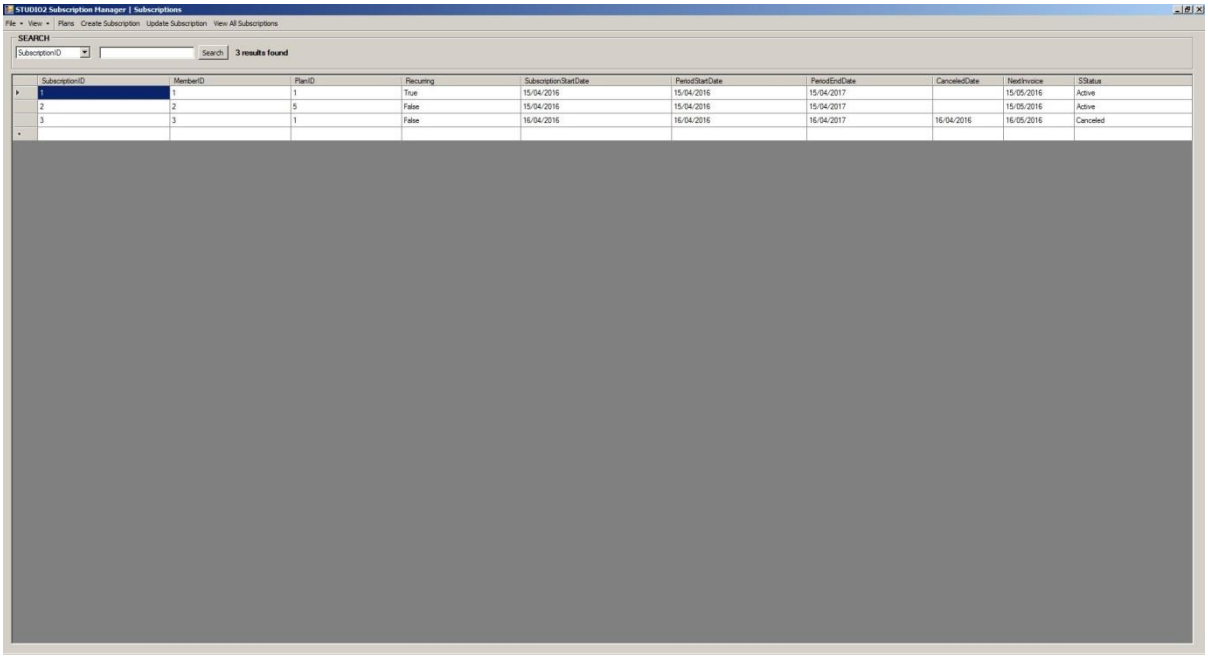
Test 3-33: Invoice corresponding to selected subscription (SubscriptionID = 3) IStatus = 'Pending'



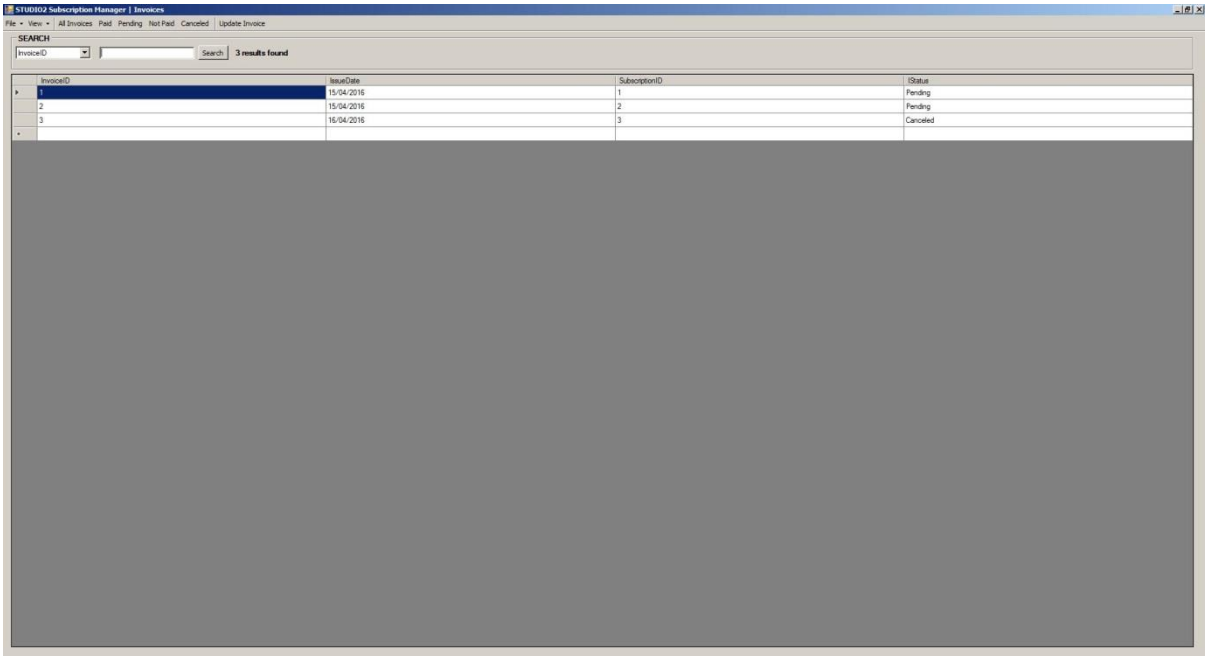
Test 3-33: Click 'Cancel Subscription' Button when an invoice record with IStatus = 'Pending' is linked to the subscription



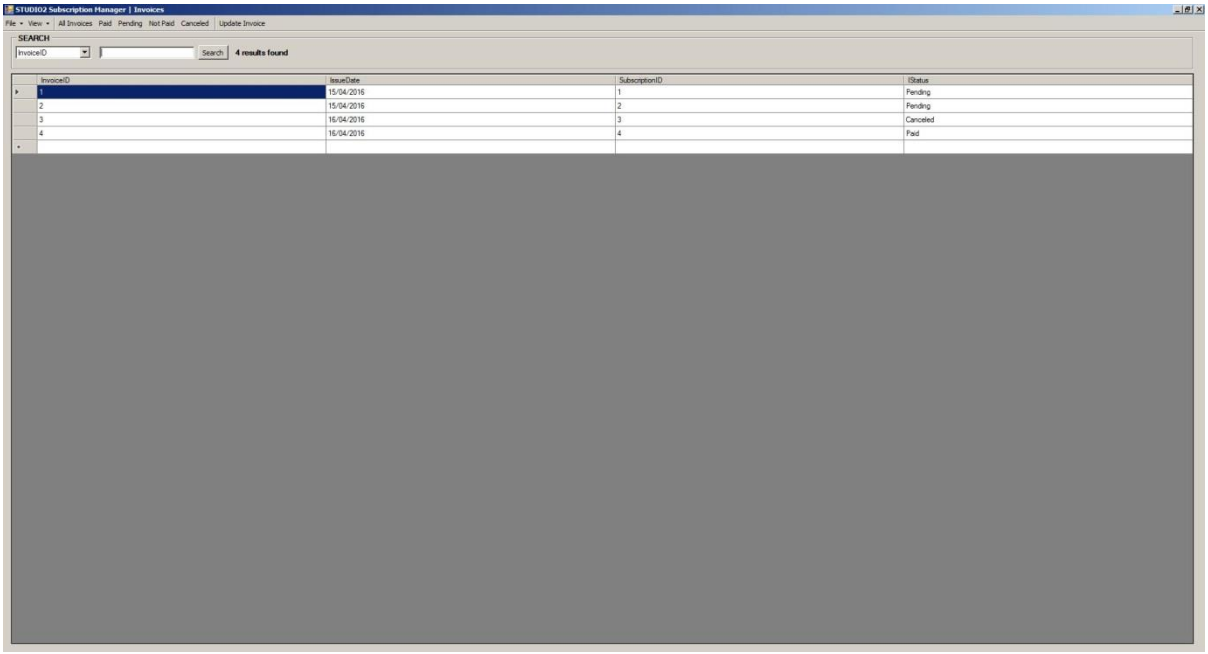
Test 3-33: Confirmation MessageBox



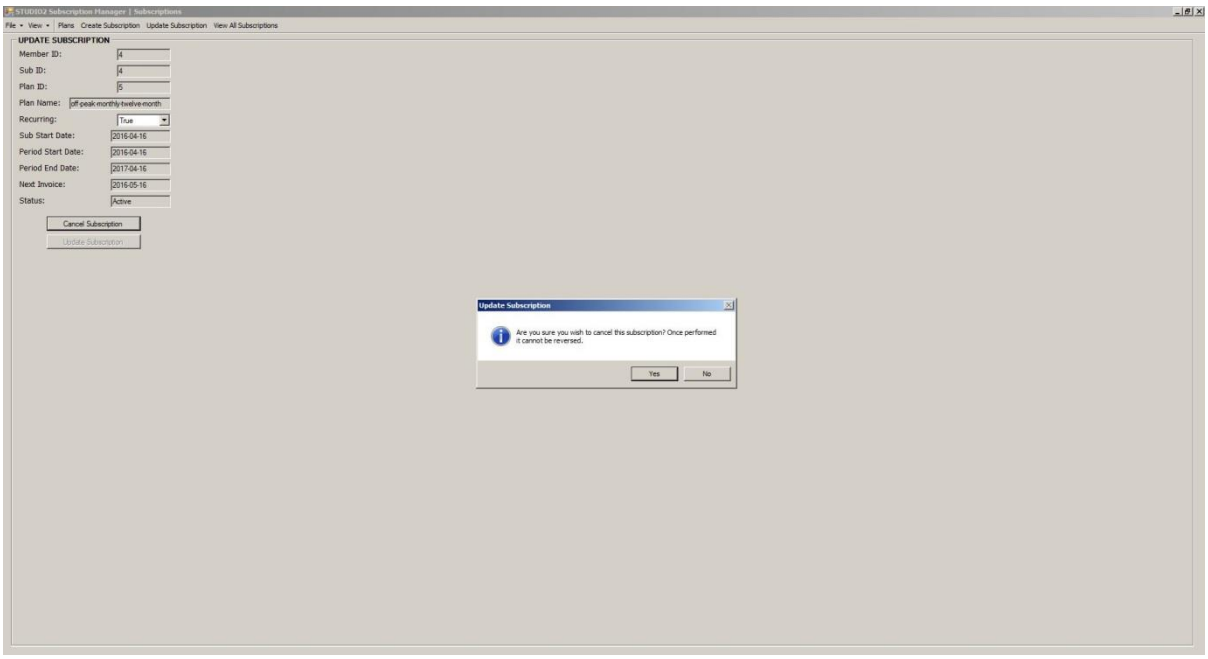
Test 3-33: Displays dgvSQLOutput with updated subscription with SStatus = 'Canceled' and CanceledDate = '16/05/2016'



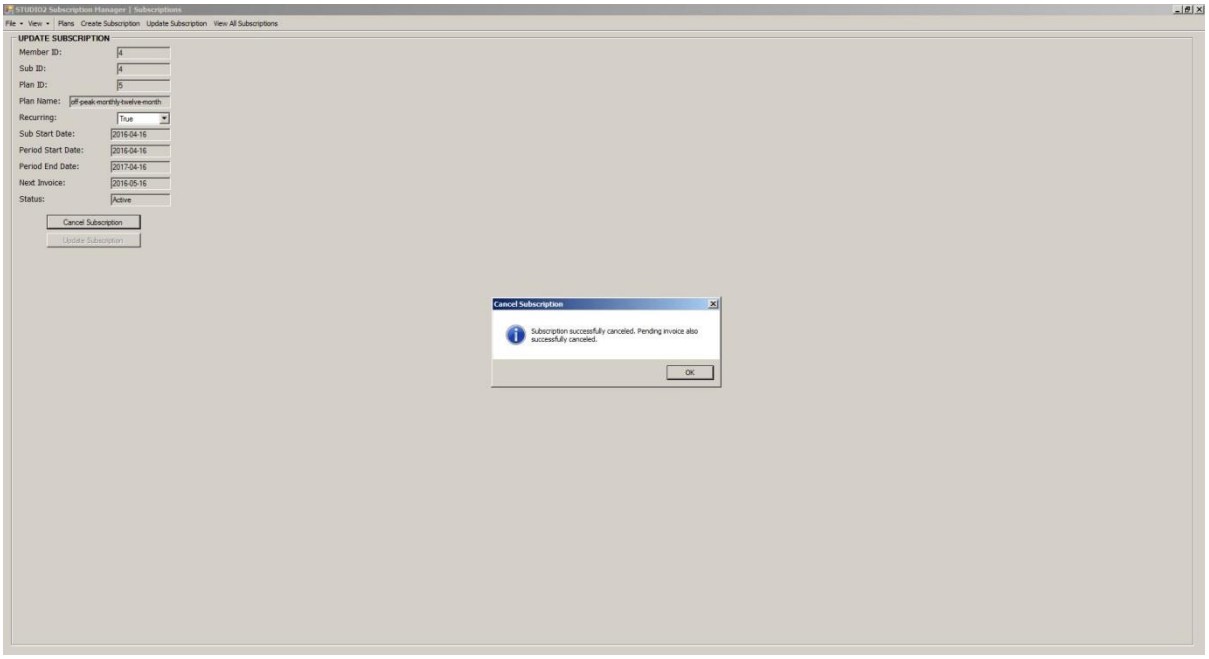
Test 3-33: Corresponding invoice with IStatus = 'Pending' now has IStatus = 'Canceled'



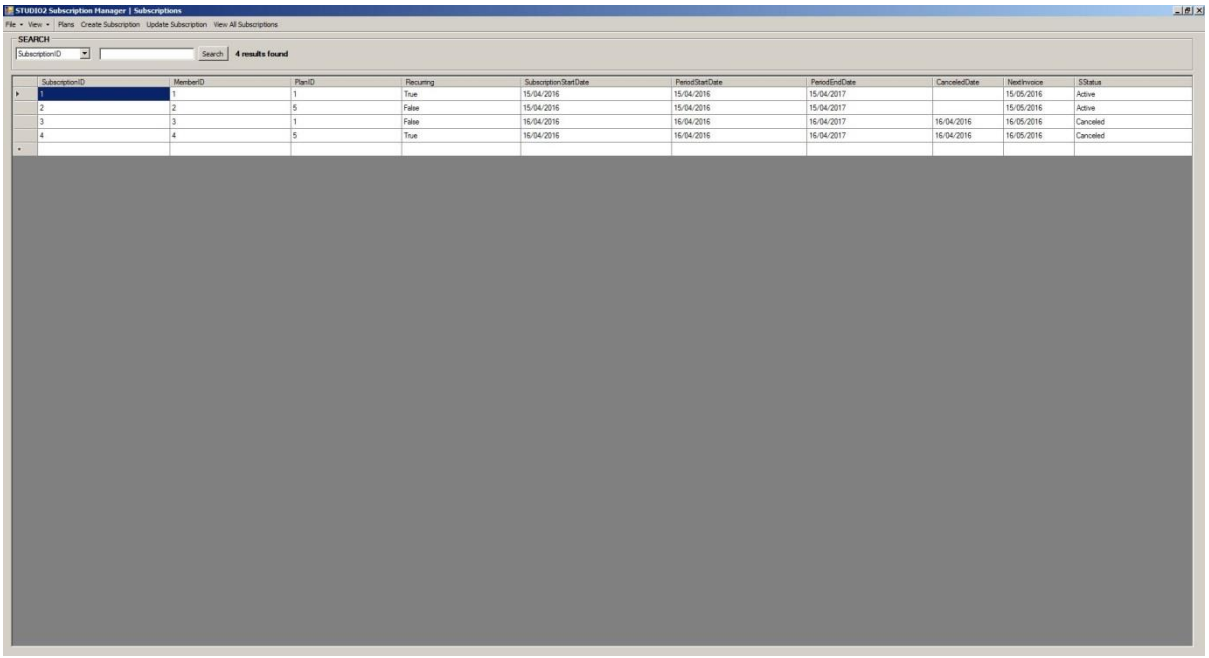
Test 3-34: Invoice corresponding to selected subscription (SubscriptionID = 4) IStatus = 'Paid'



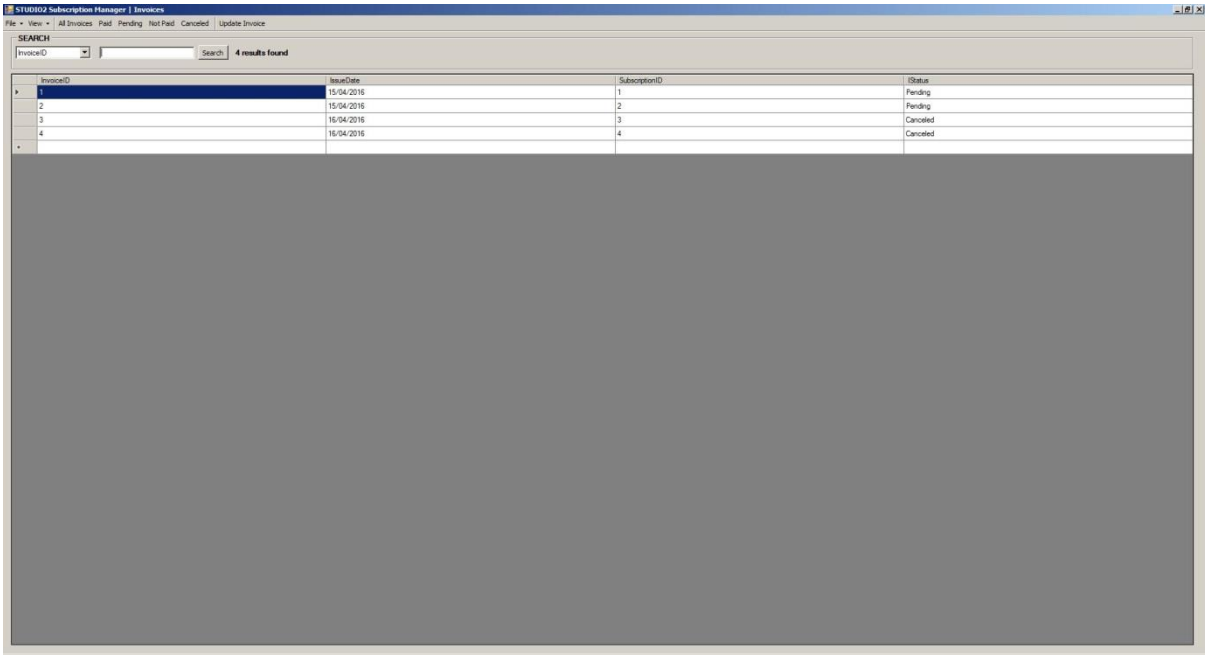
Test 3-34: Click 'Cancel Subscription' Button when there is no invoice record with IStatus = 'Pending' linked to the subscription



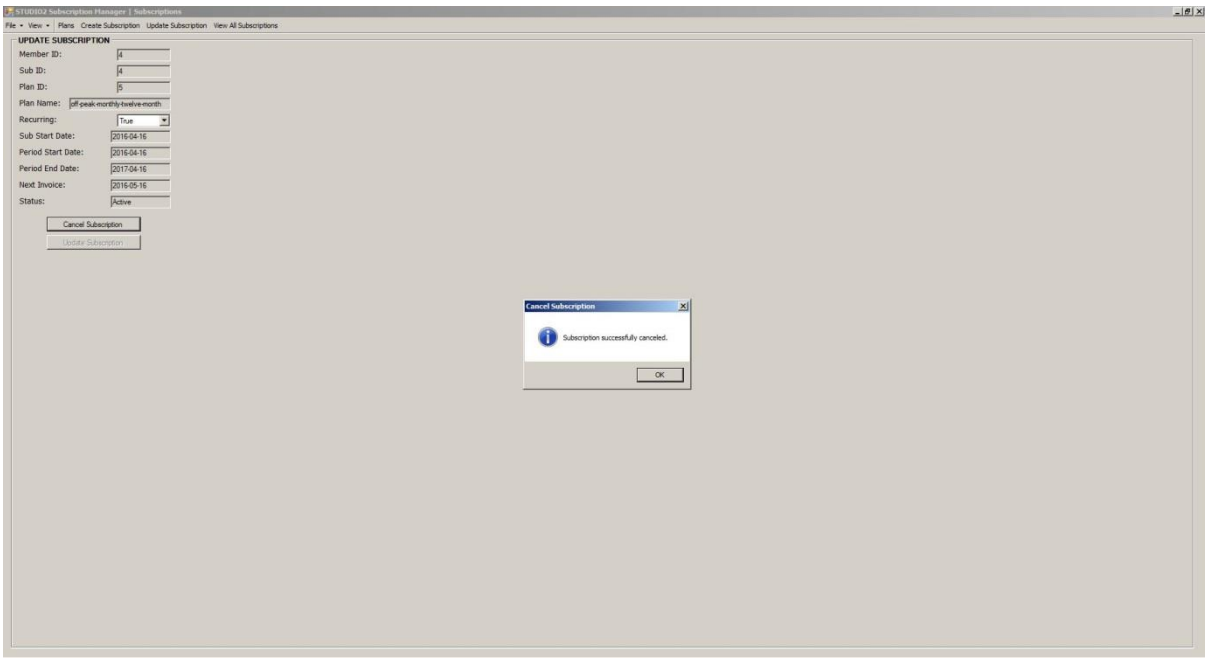
Test 3-34: MessageBox still states that the pending invoice was canceled when there is none



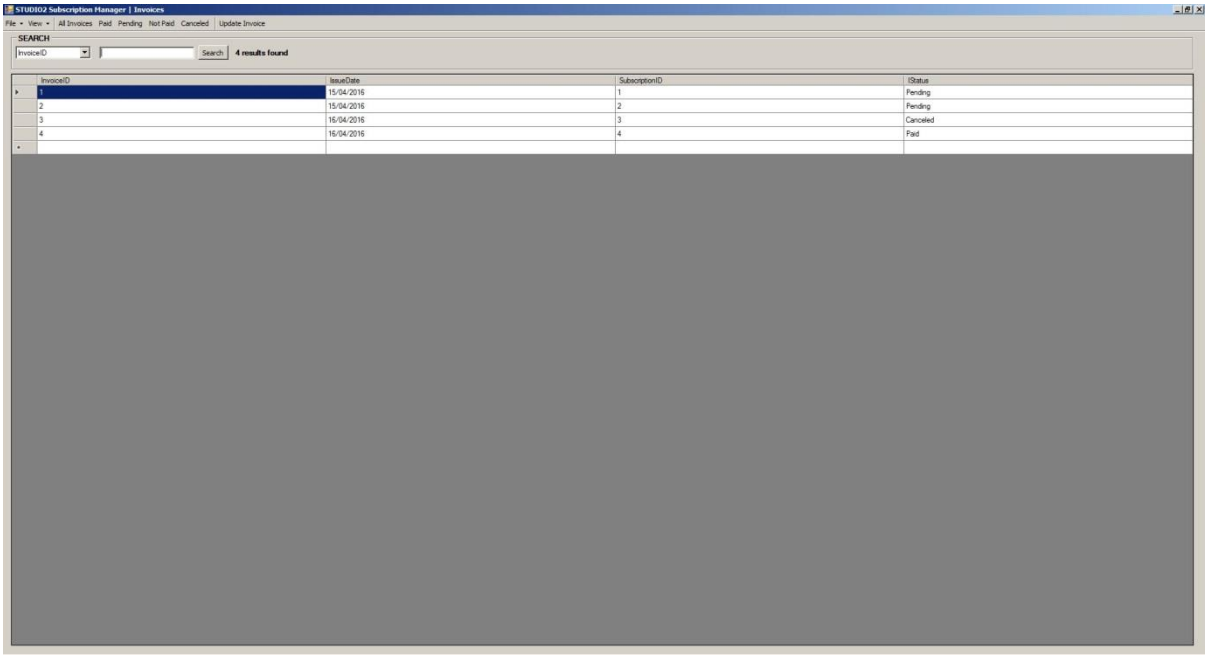
Test 3-34: Displays dgvSQLOutput with updated subscription with SStatus = 'Canceled' and CanceledDate = '16/05/2016' as intended



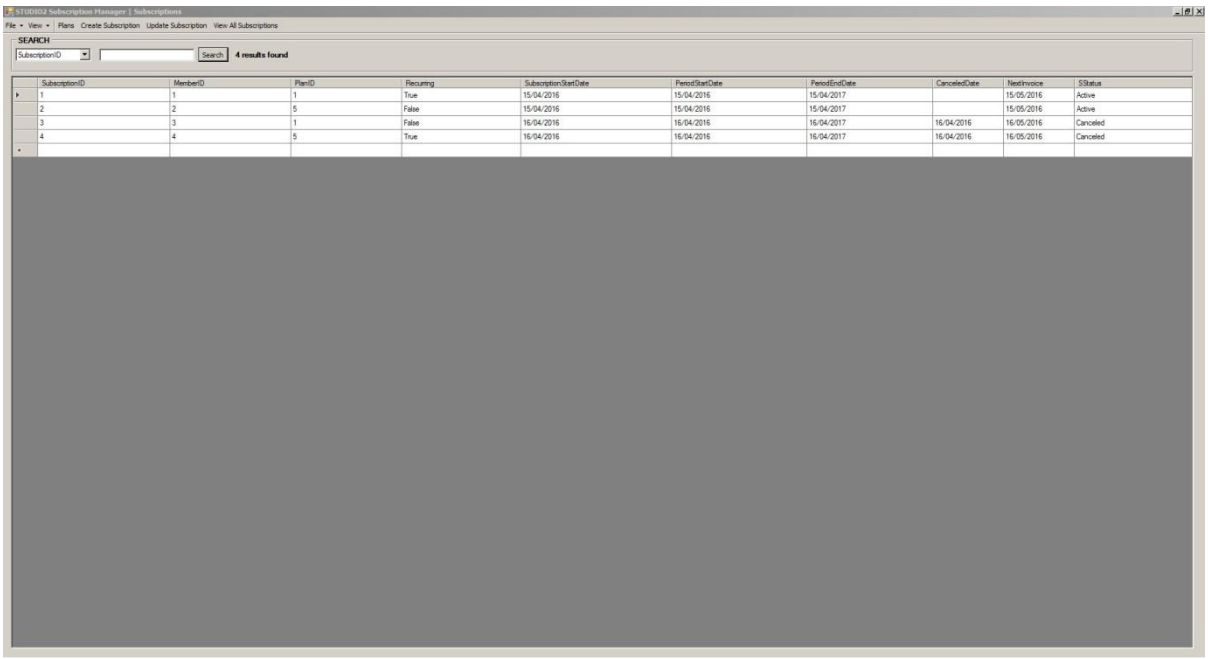
Test 3-34: Corresponding 'Paid' invoice SStatus has been changed to 'Canceled'



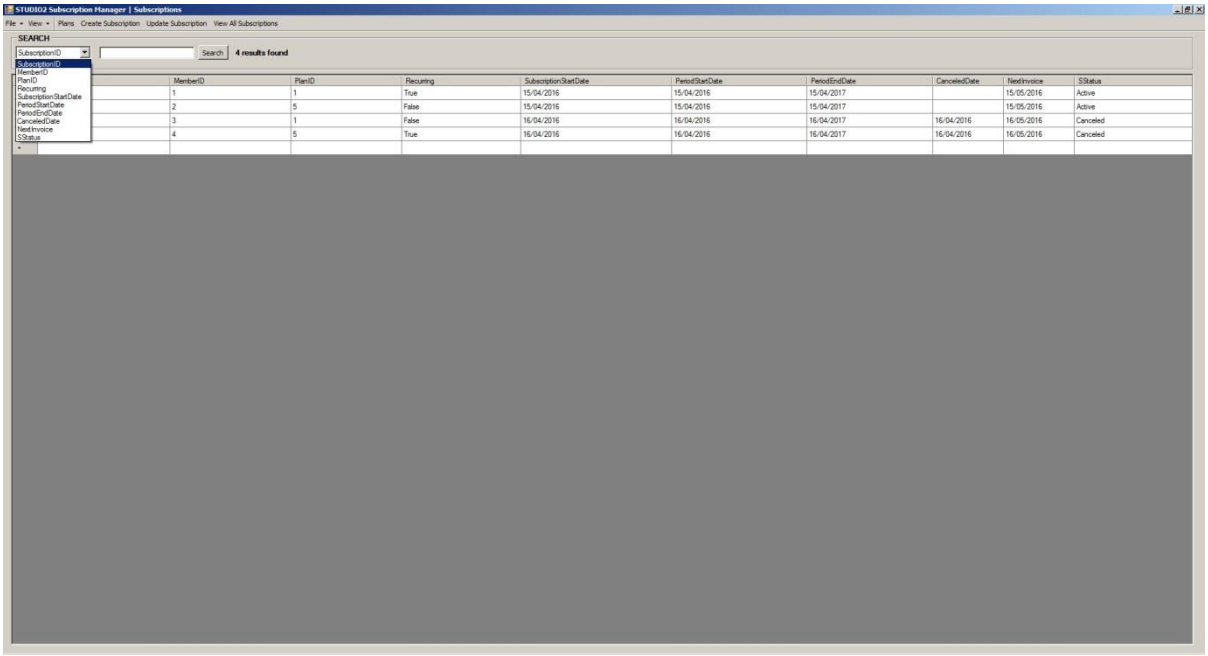
Test 3-34: Corrective action taken. Correct message now displays in MessageBox



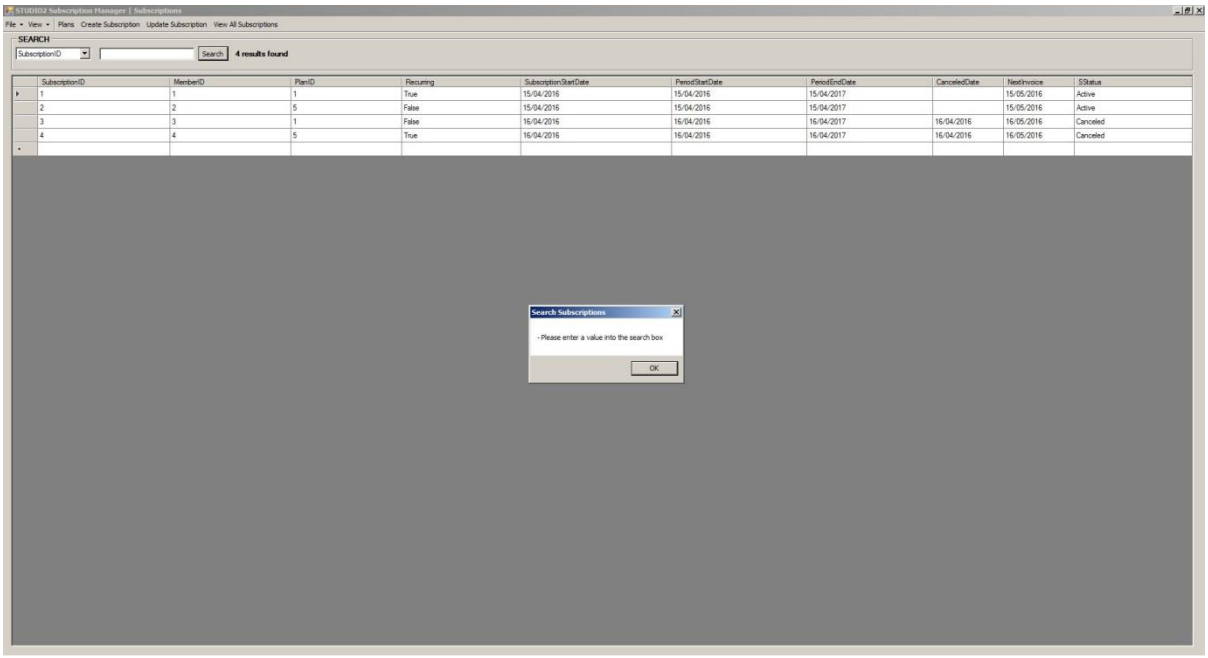
Test 3-34: 'Paid' invoice remained unaffected as intended



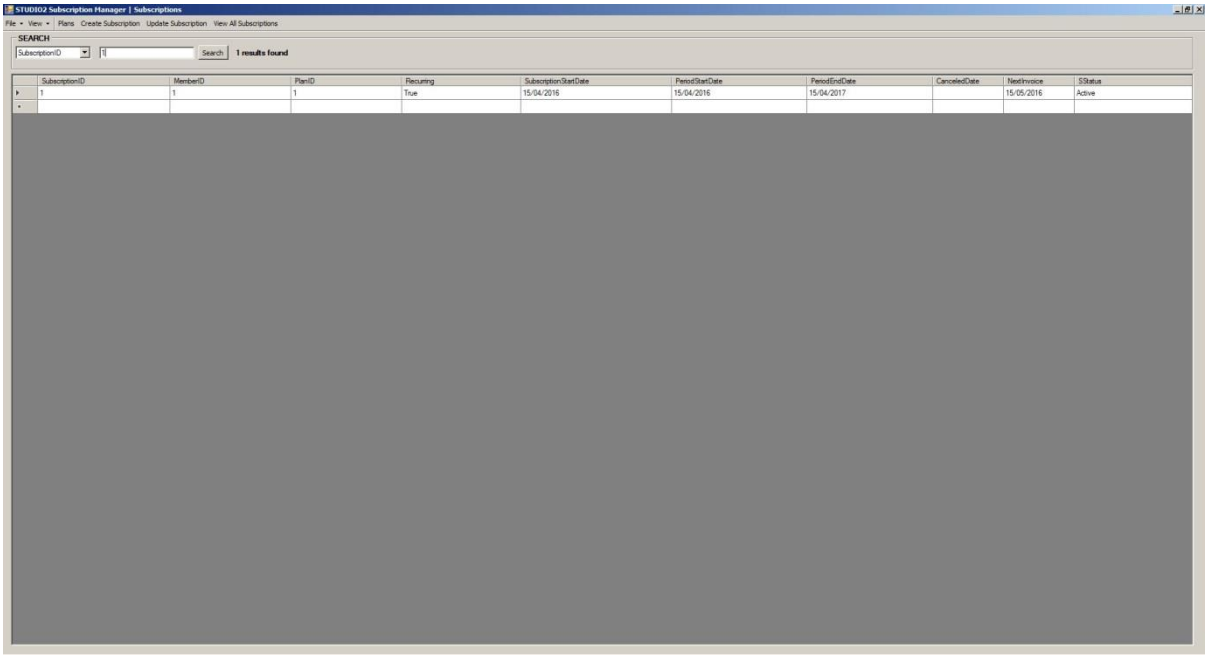
Test 3-35: Click 'View All Subscriptions' ToolStripButton



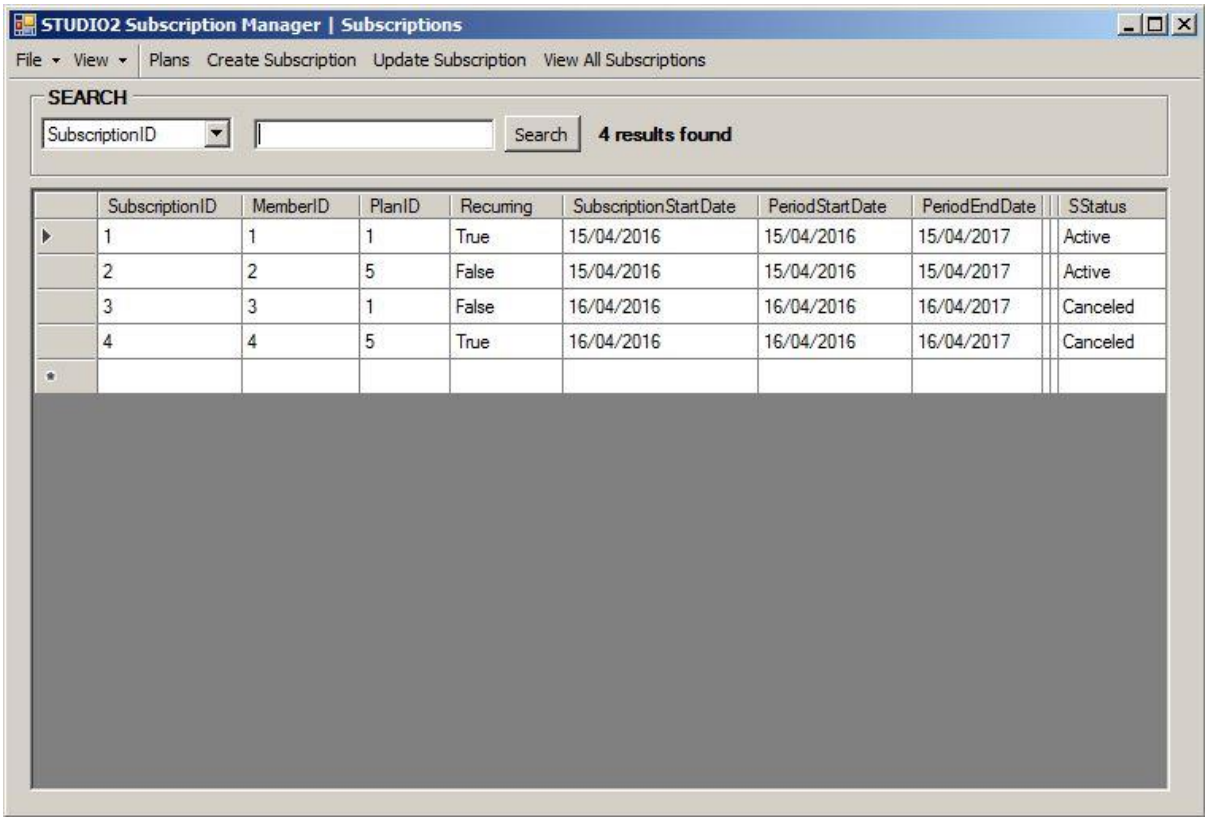
Test 3-36: Click cboSearchField ComboBox



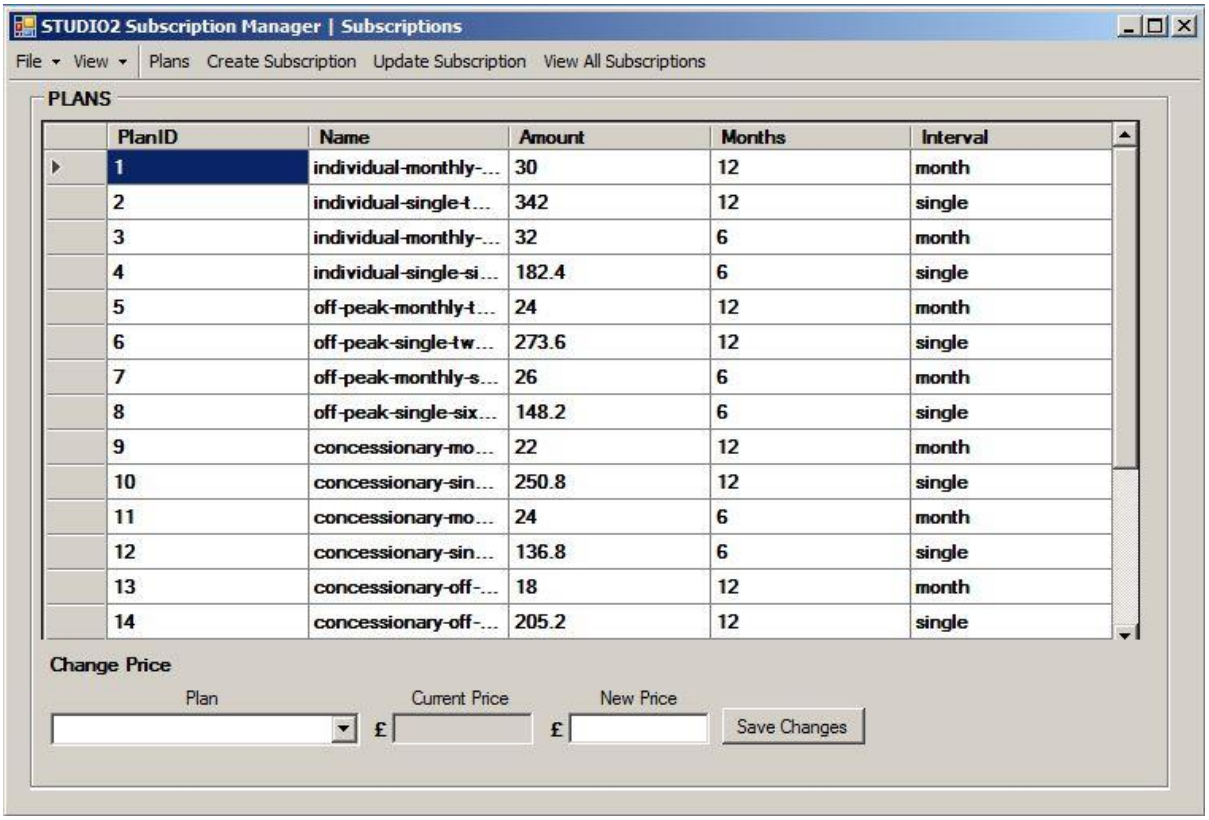
Test 3-37: Click 'Search' Button when no text has been entered into txtSearch TextBox



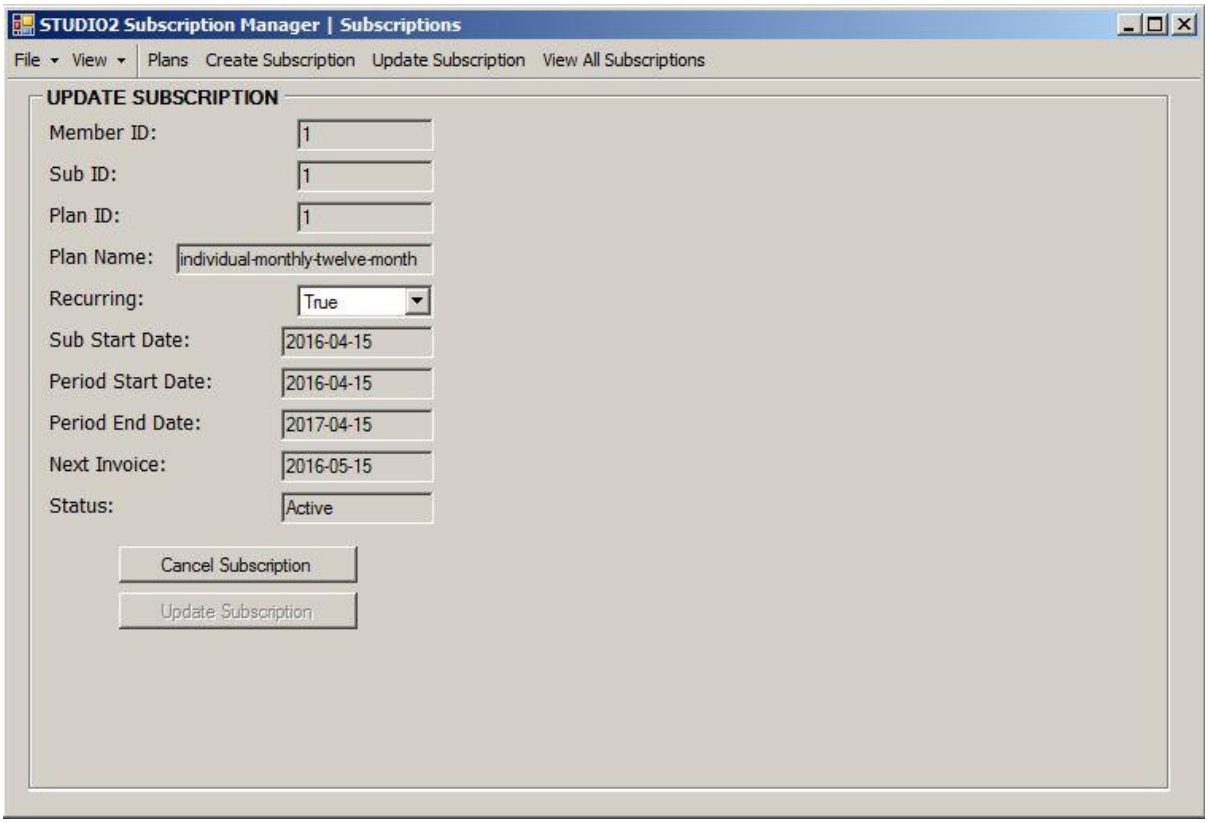
Test 3-38: Click 'Search' Button when '1' has been entered into txtSearch TextBox and cboField text is 'SubscriptionID'



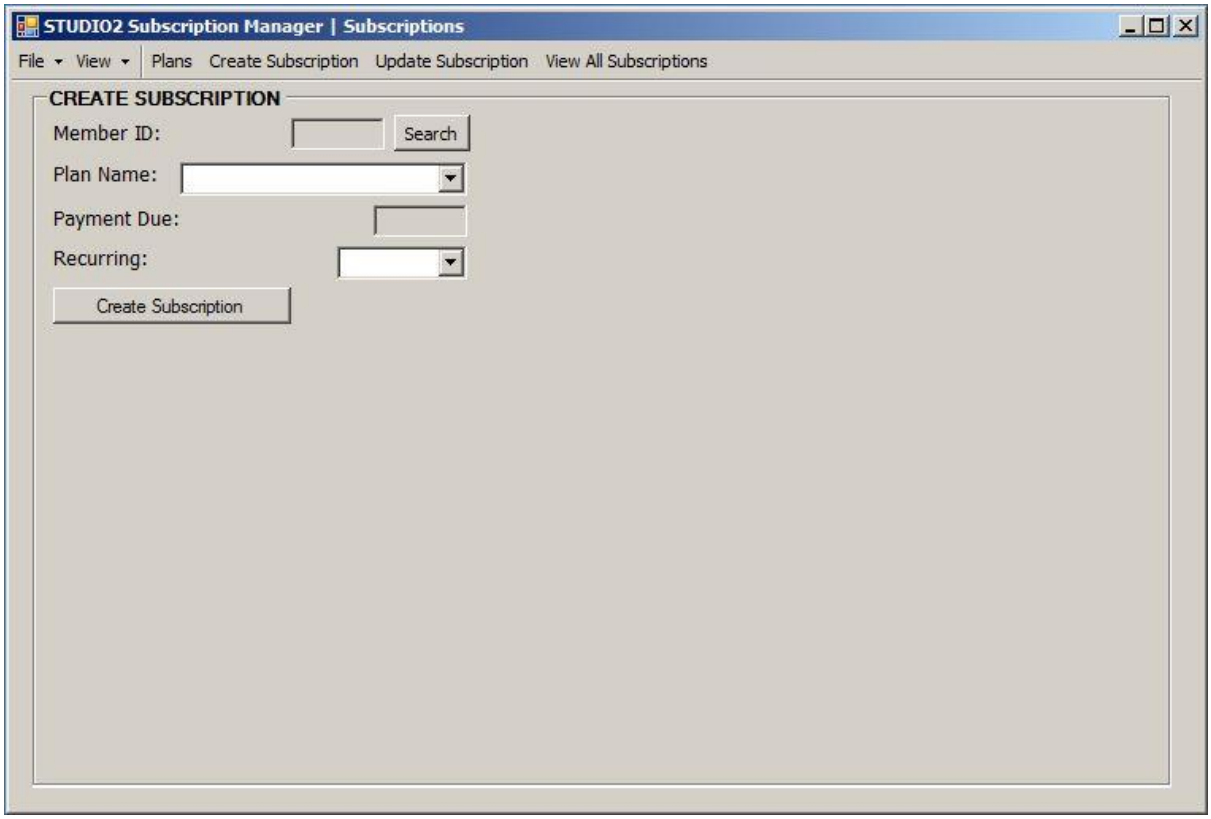
Test 3-39: Resize form when dgvSQLOutput is Visible



Test 3-40: Resize form when grpPlans GroupBox is Visible

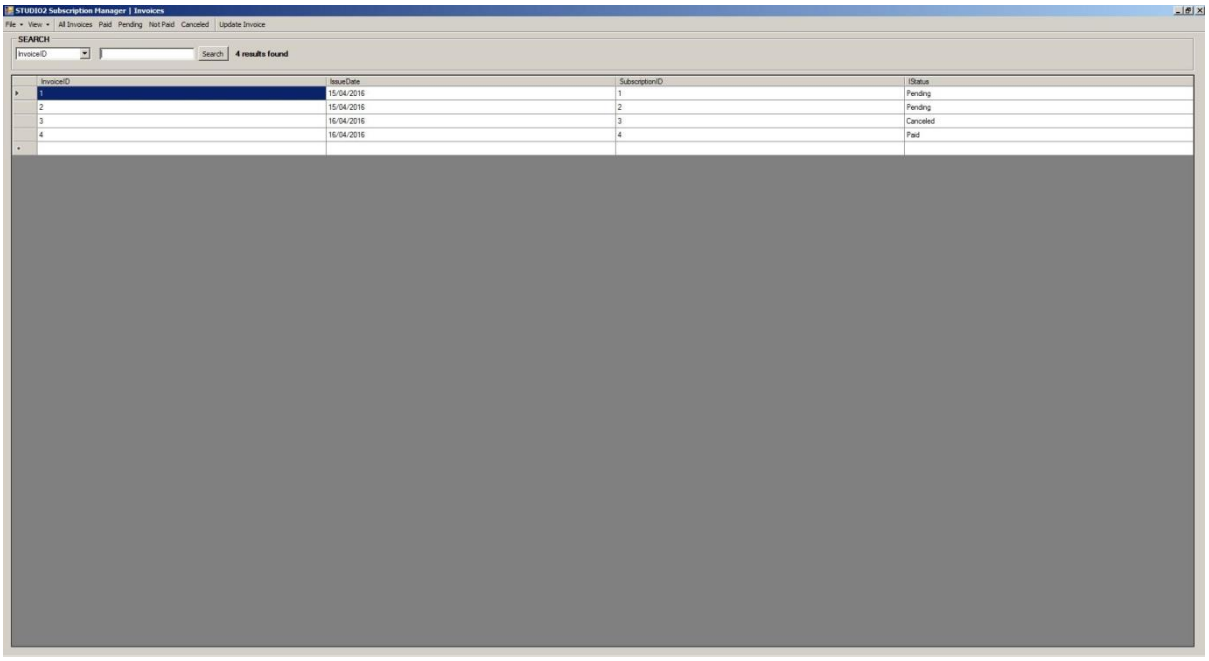


Test 3-41: Resize form when grpUpdateSubscription is Visible

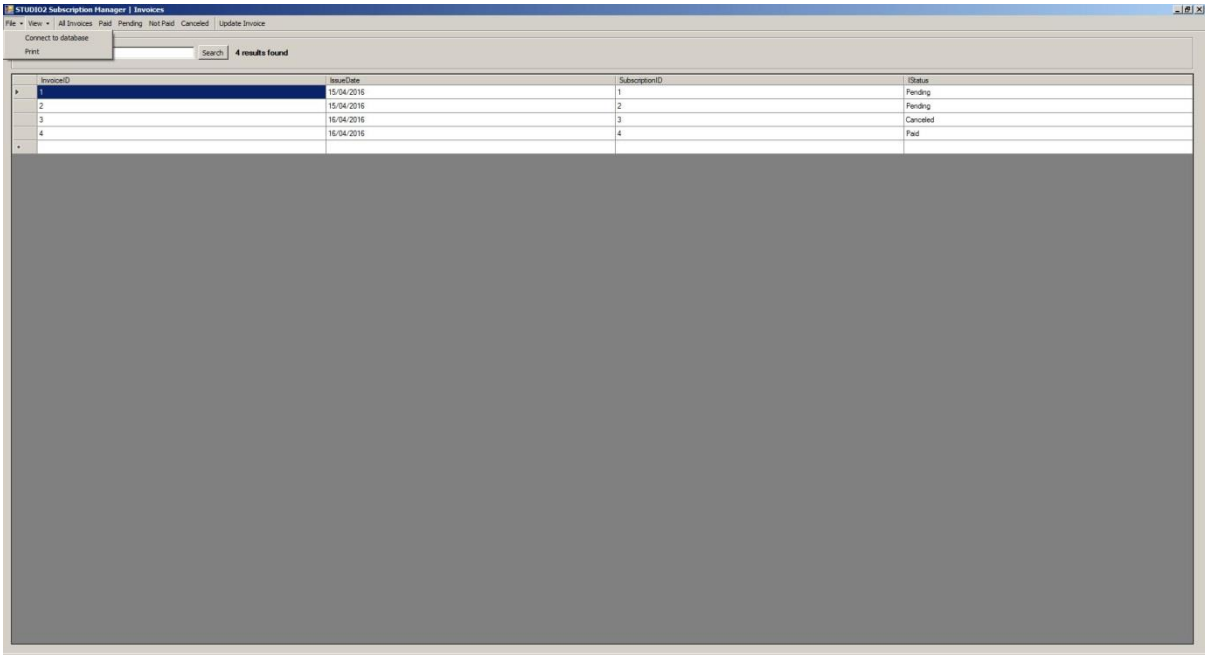


Test 3-41: Resize form when grpCreateSubscription is Visible

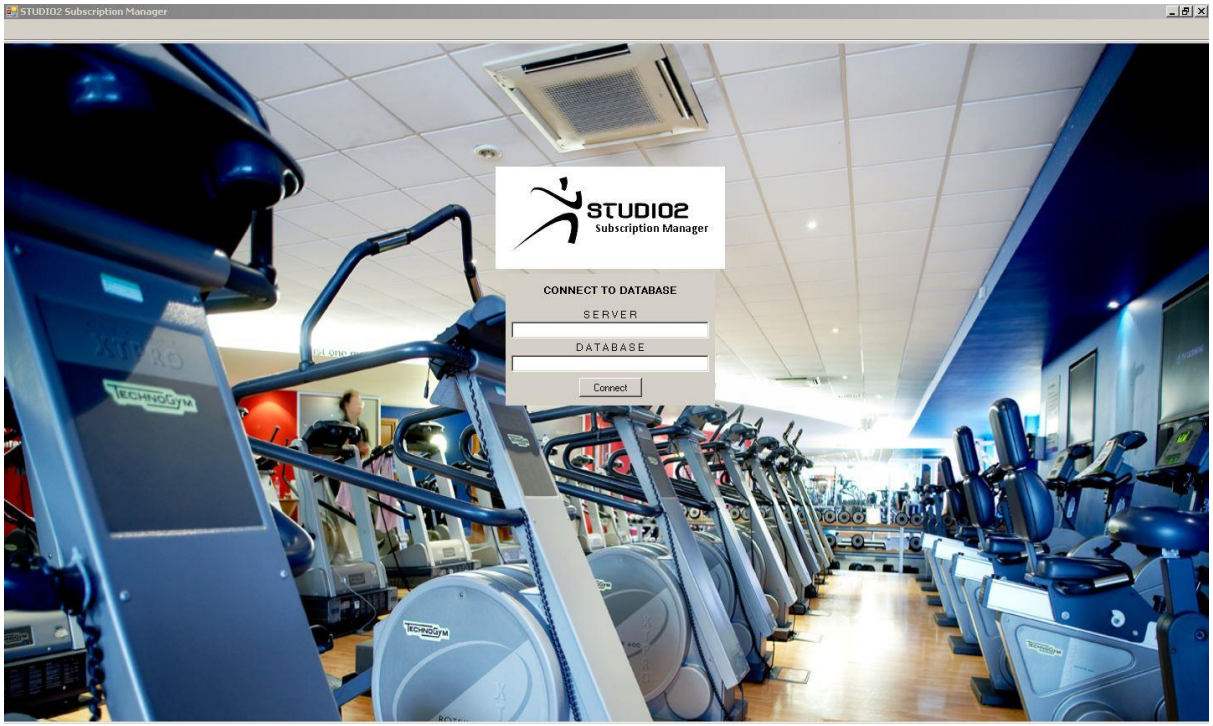
INVOICES FORM



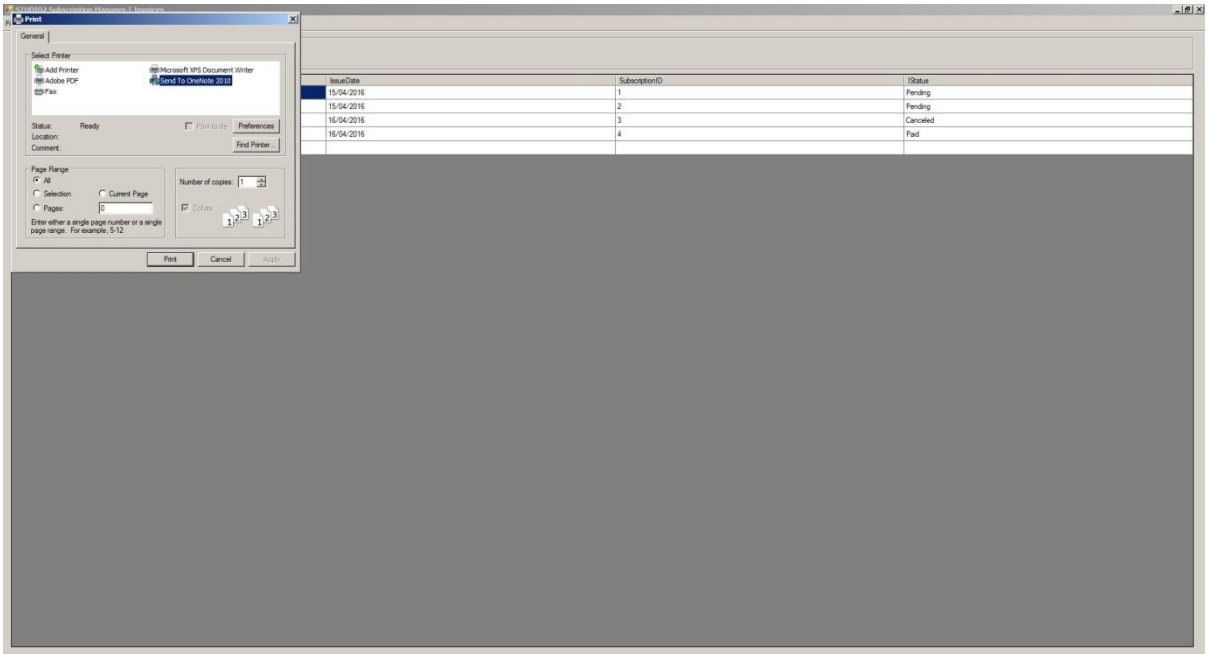
Test 4-1: Load 'Invoices' form



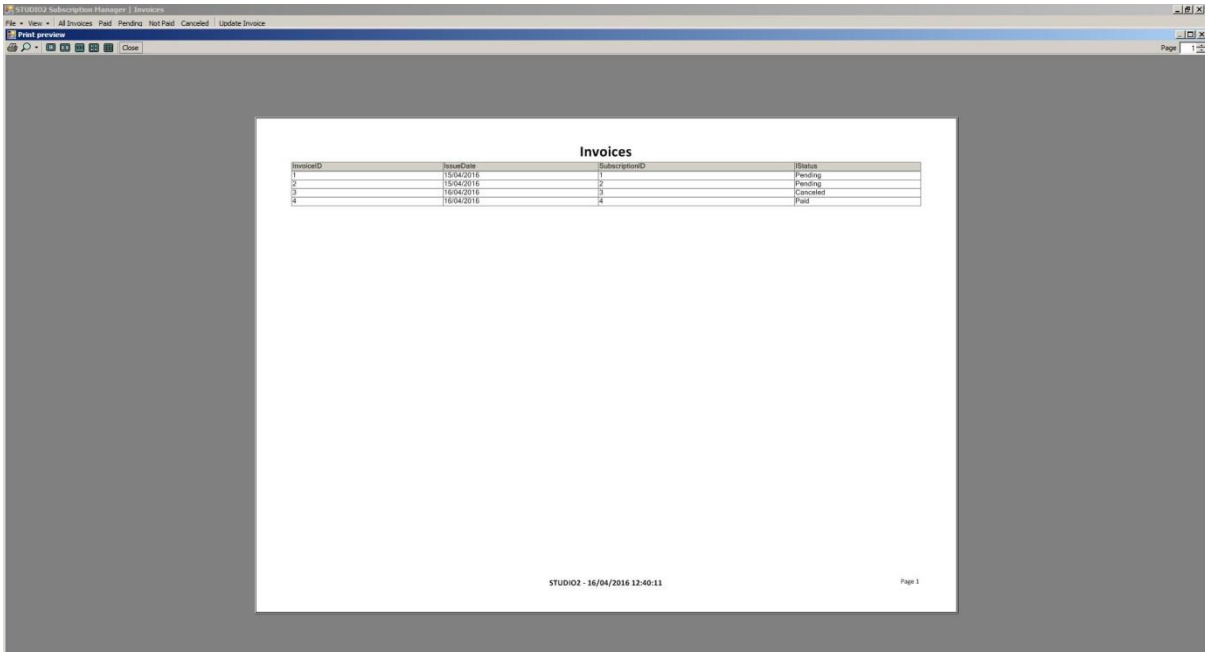
Test 4-2: Click 'File' ToolStripDropDownButton



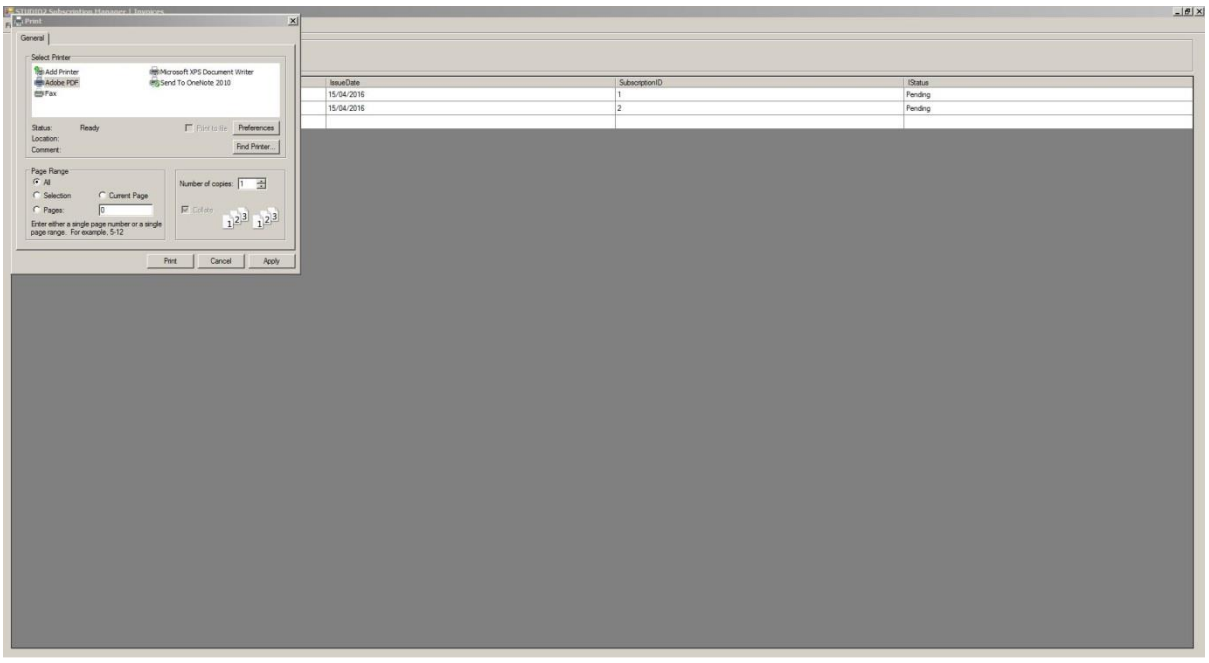
Test 4-3: Corrective action taken. Click 'Connect to database' ToolStripMenuItem shows 'Start' Form as intended



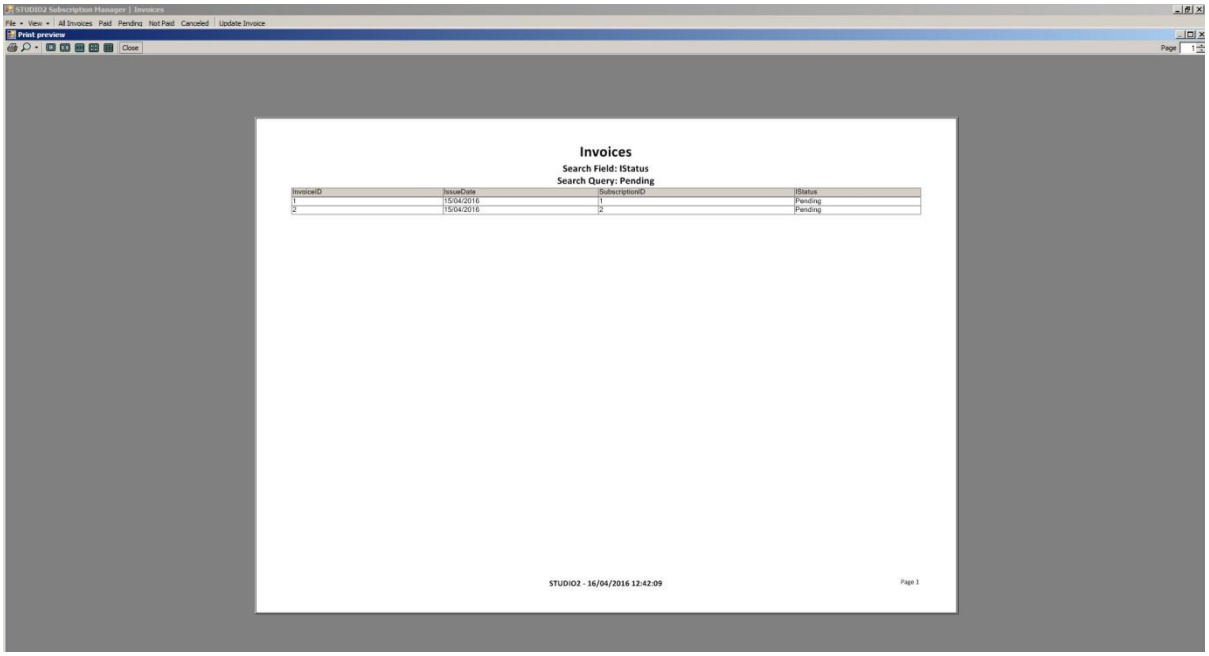
Test 4-4: Click 'Print' ToolStripMenuItem



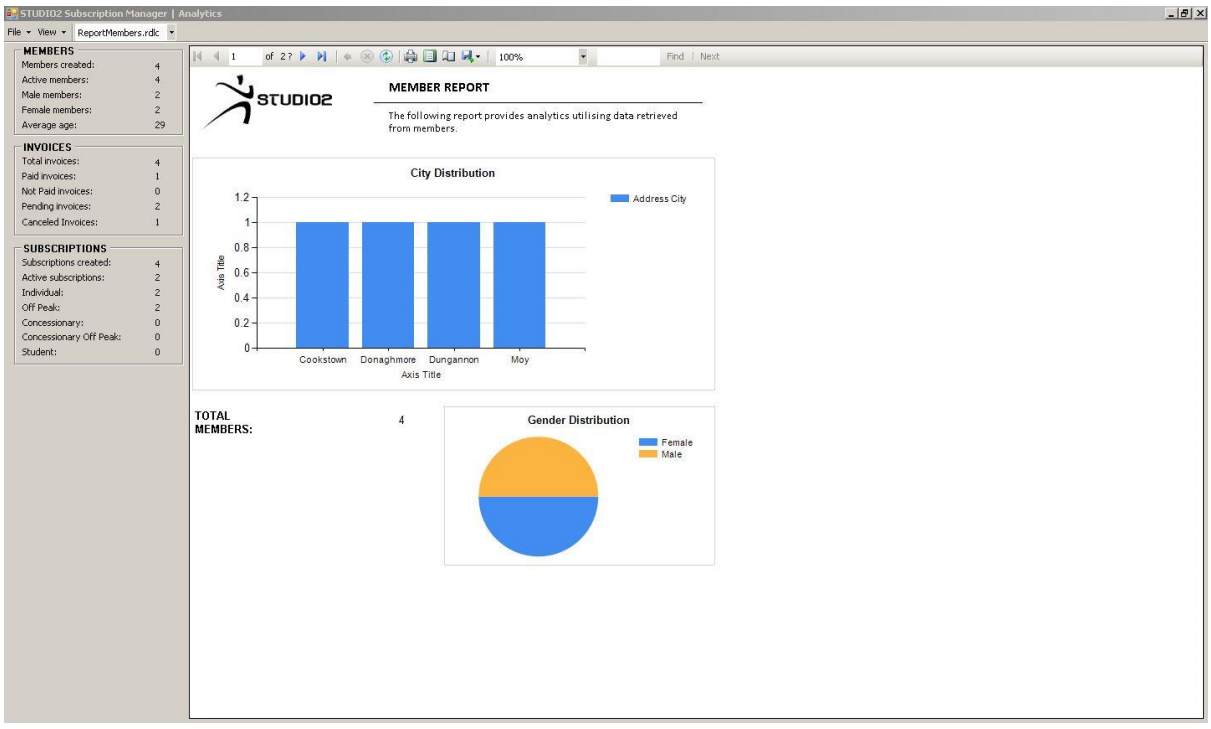
Test 4-4: Click 'Print' in PrintSettings



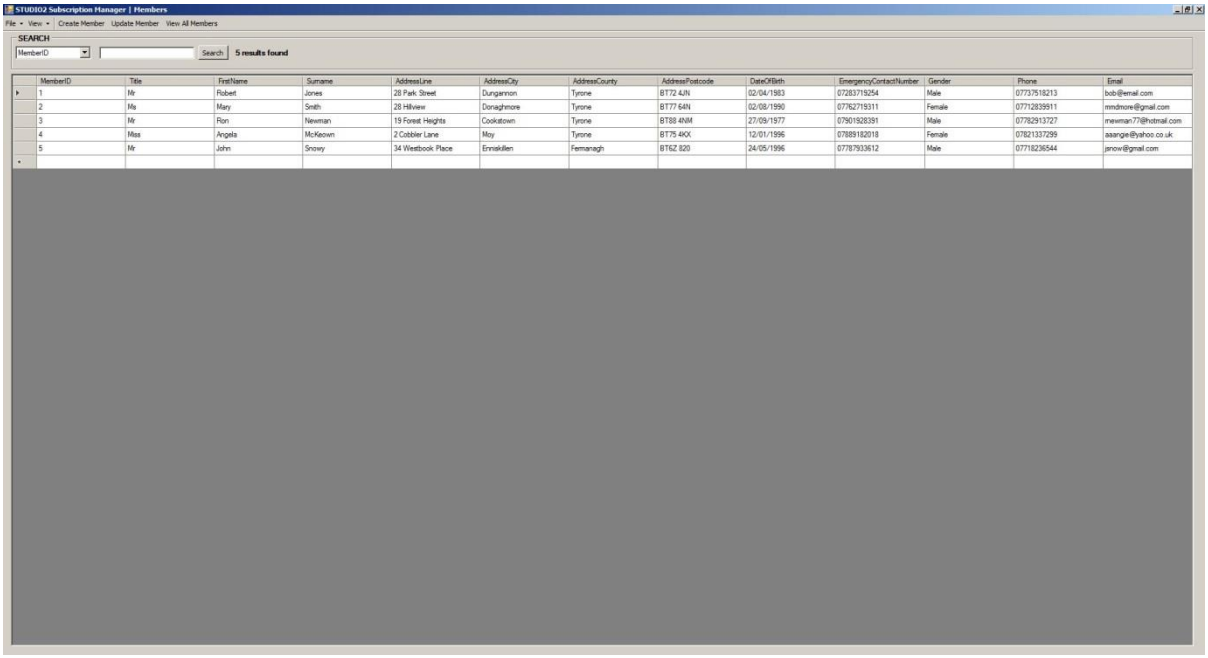
Test 4-5: Click 'Print' ToolStripMenuItem after doing a search query



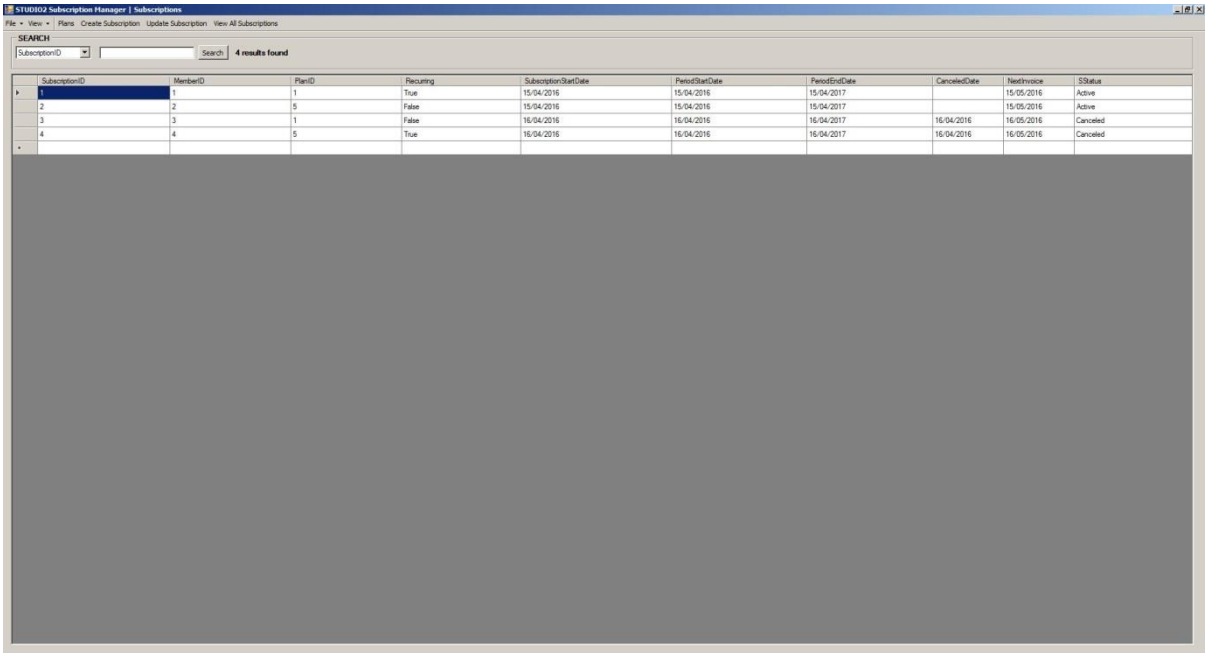
Test 4-5: Click 'Print' in PrintSettings



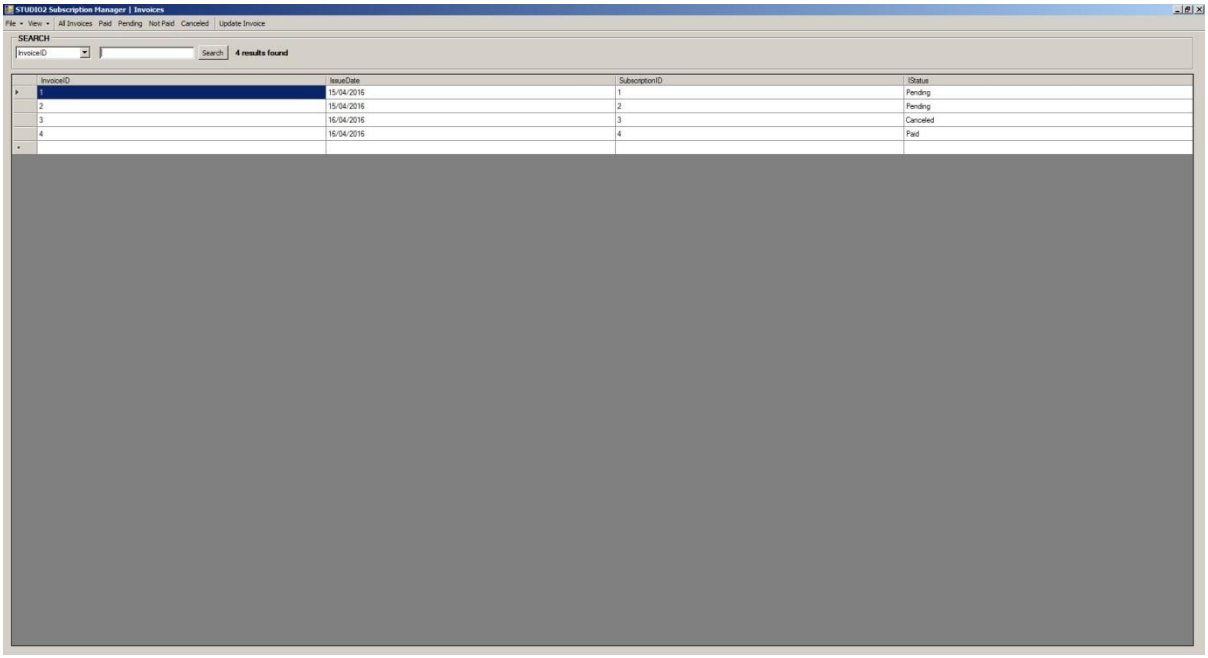
Test 4-6: Click 'Analytics' ToolStripMenuItem



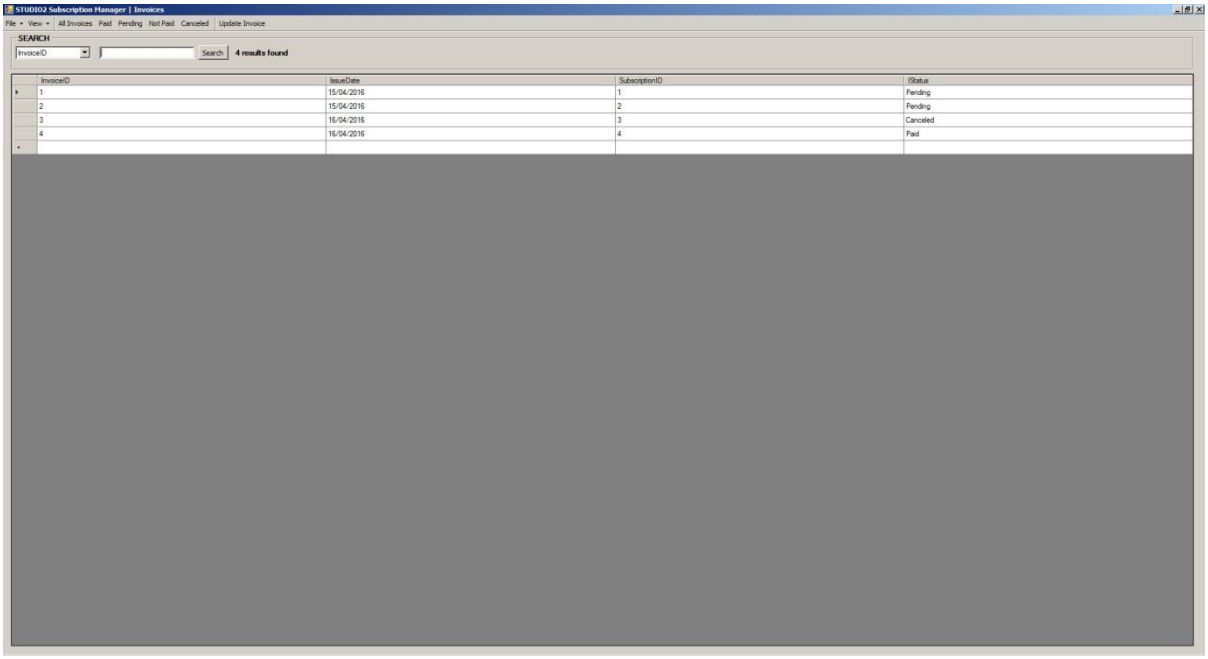
Test 4-7: Click 'Members' ToolStripMenuItem



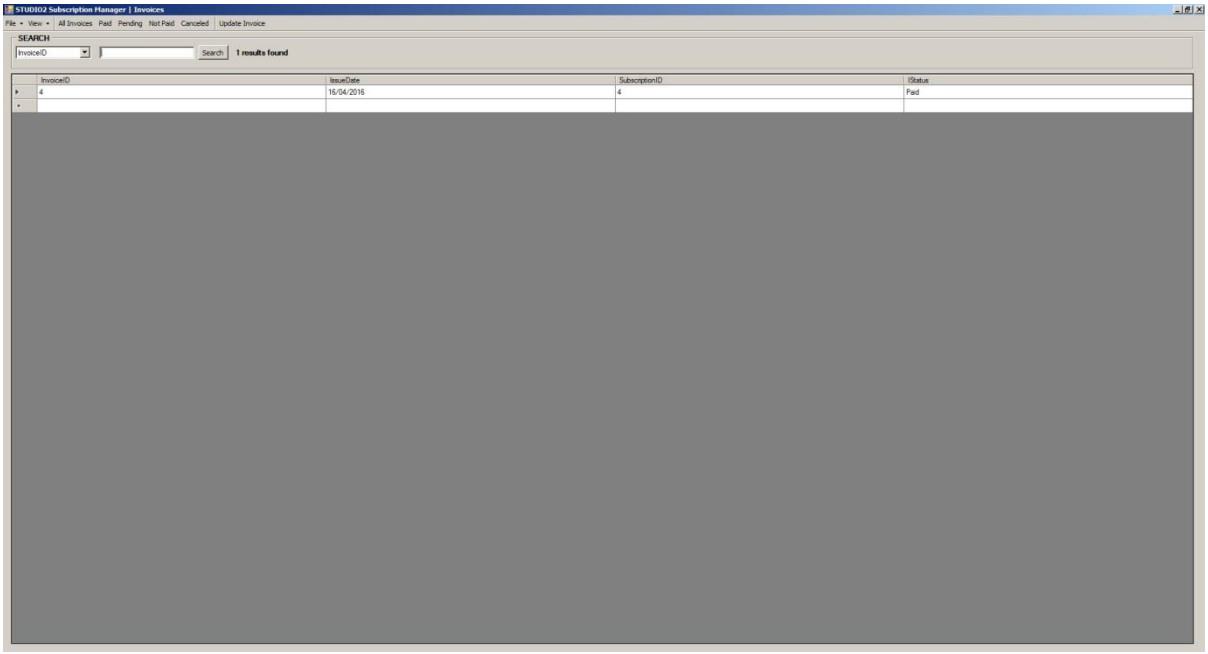
Test 4-8: Click 'Subscriptions' ToolStripMenuItem



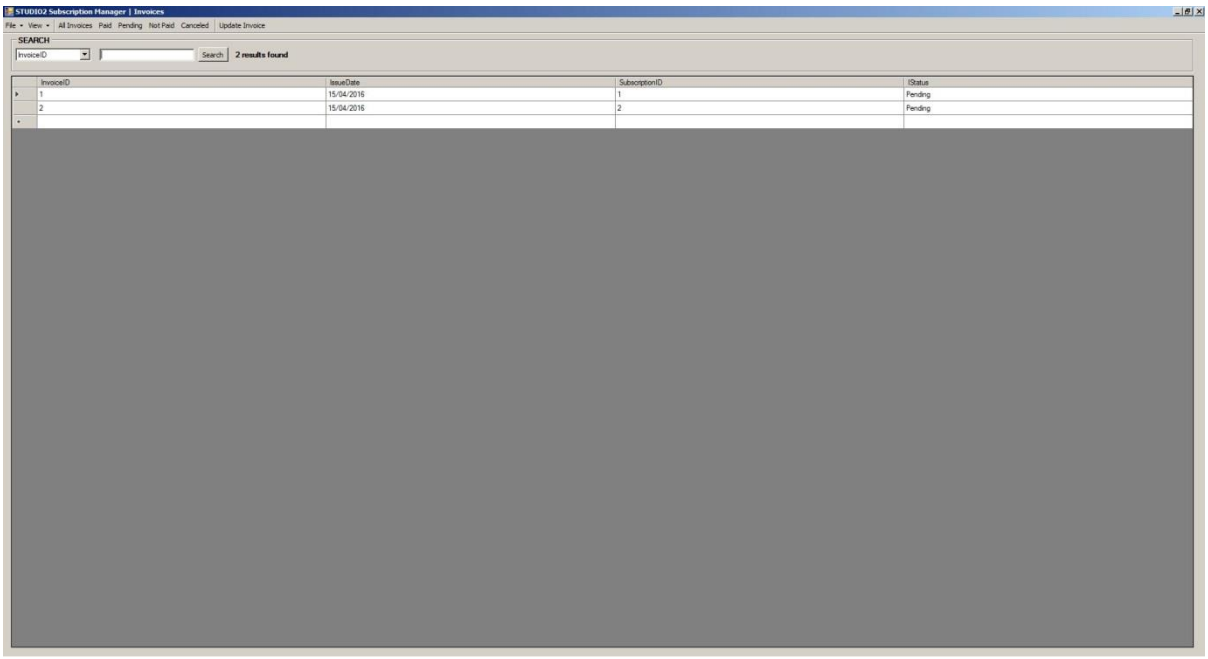
Test 4-9: Click 'Invoices' ToolStripMenuItem



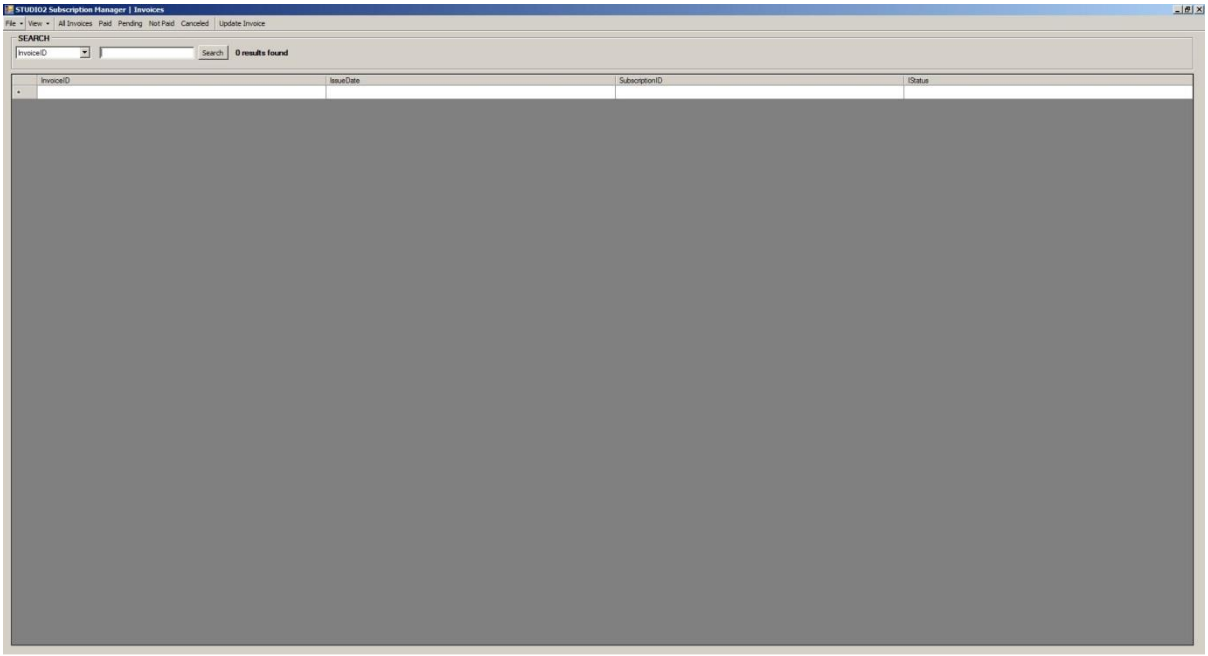
Test 4-10: Click 'All Invoices' ToolStripButton



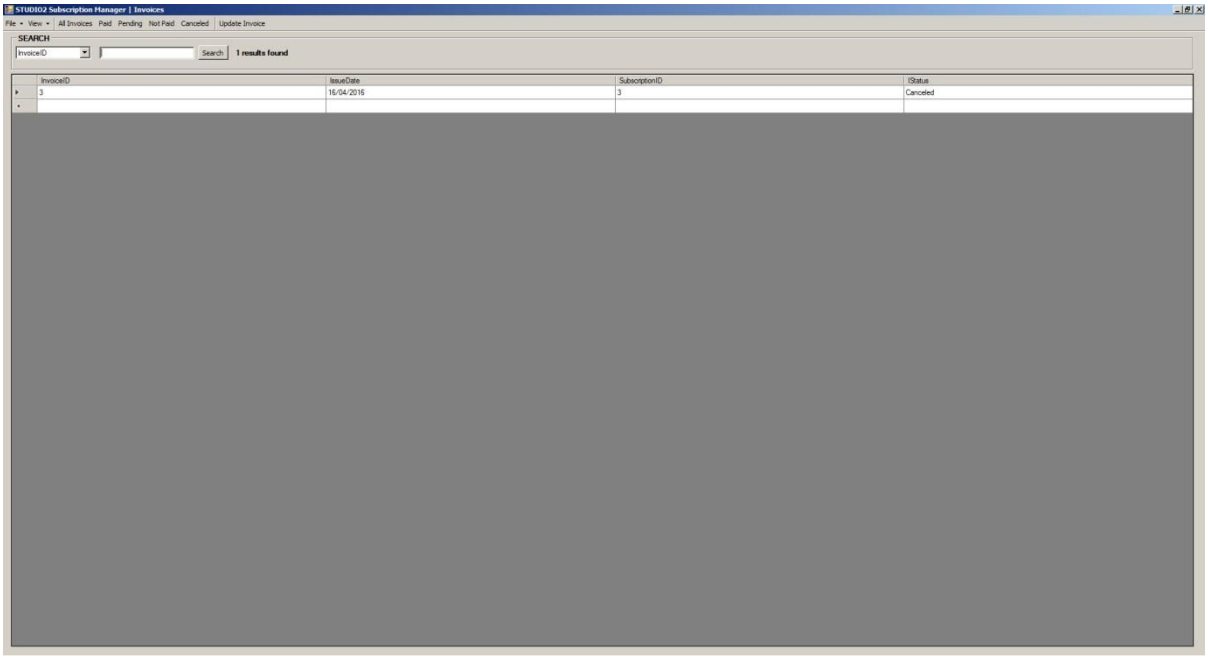
Test 4-11: Click 'Paid' ToolStripButton



Test 4-12: Click 'Pending' ToolStripButton

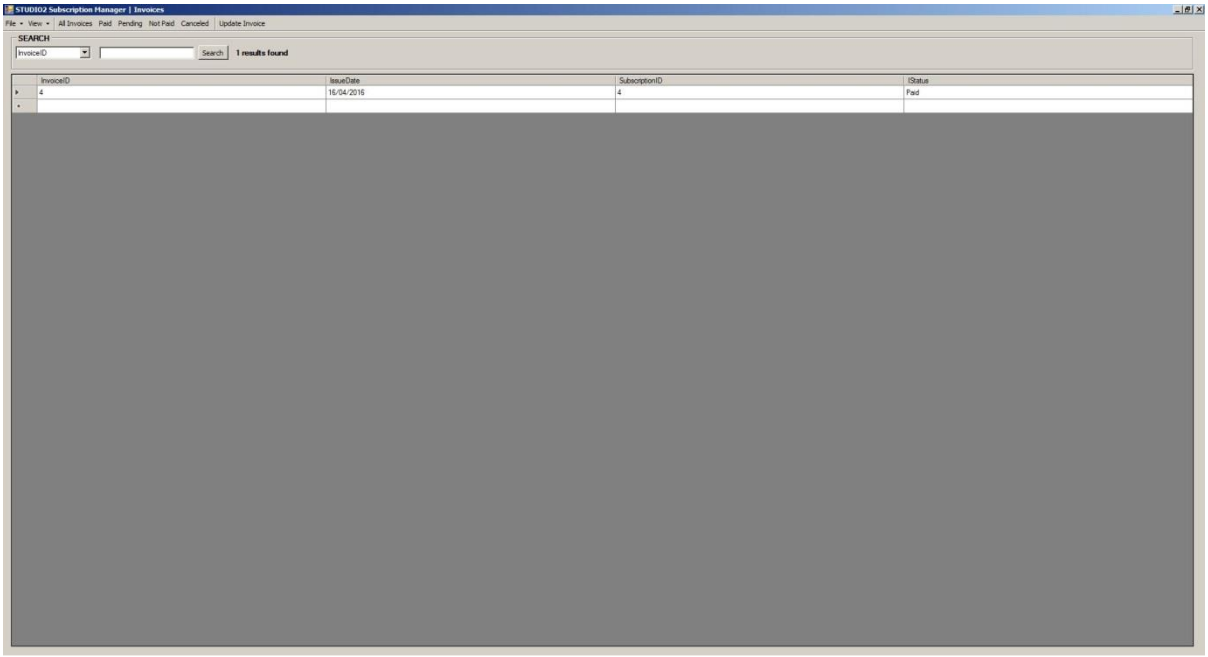


Test 4-13: Click 'Not Paid' ToolStripButton

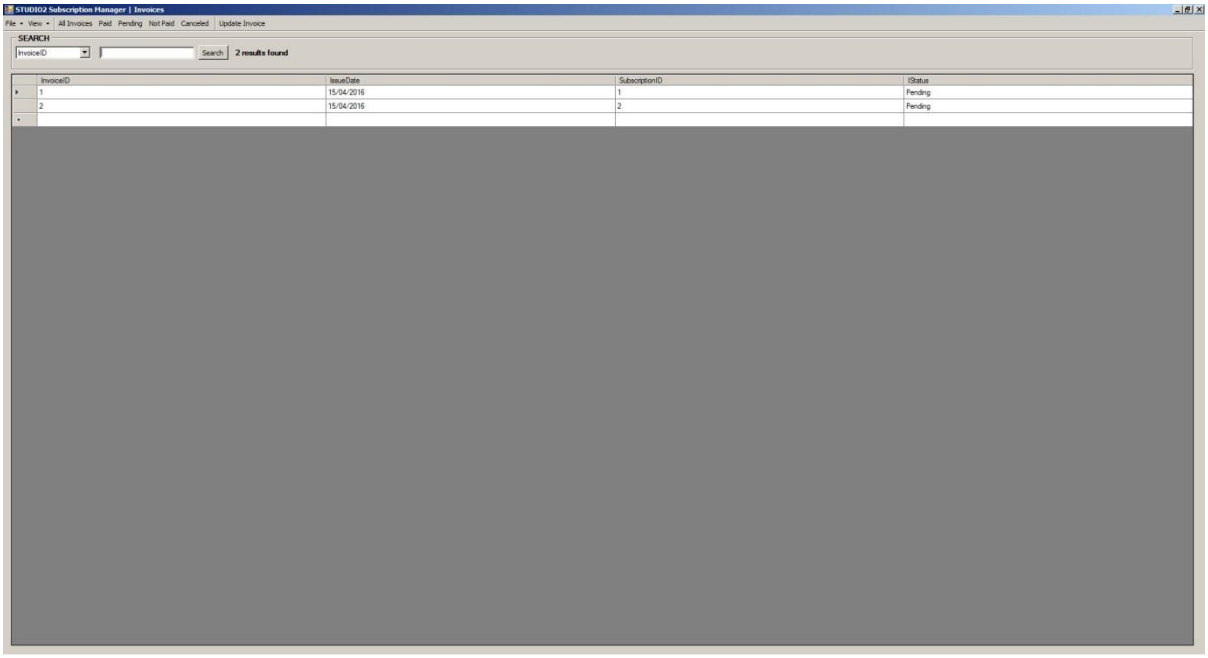


Test 4-14: Click 'Canceled' ToolStripButton

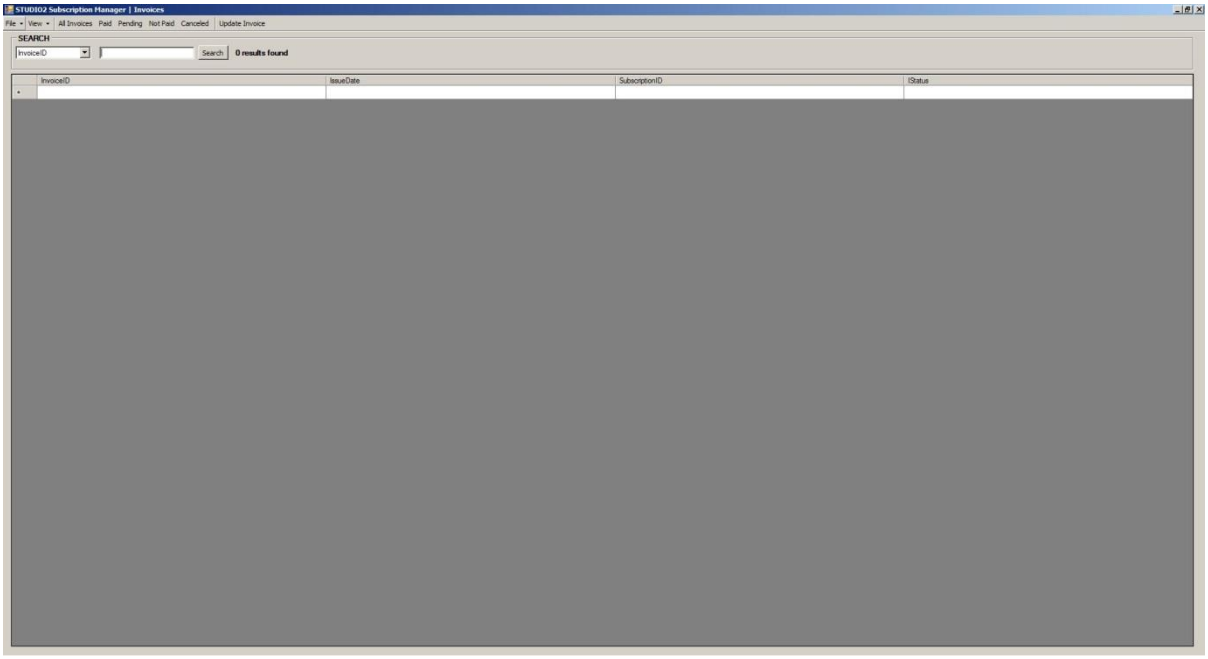
Test 4-15 to 4-18



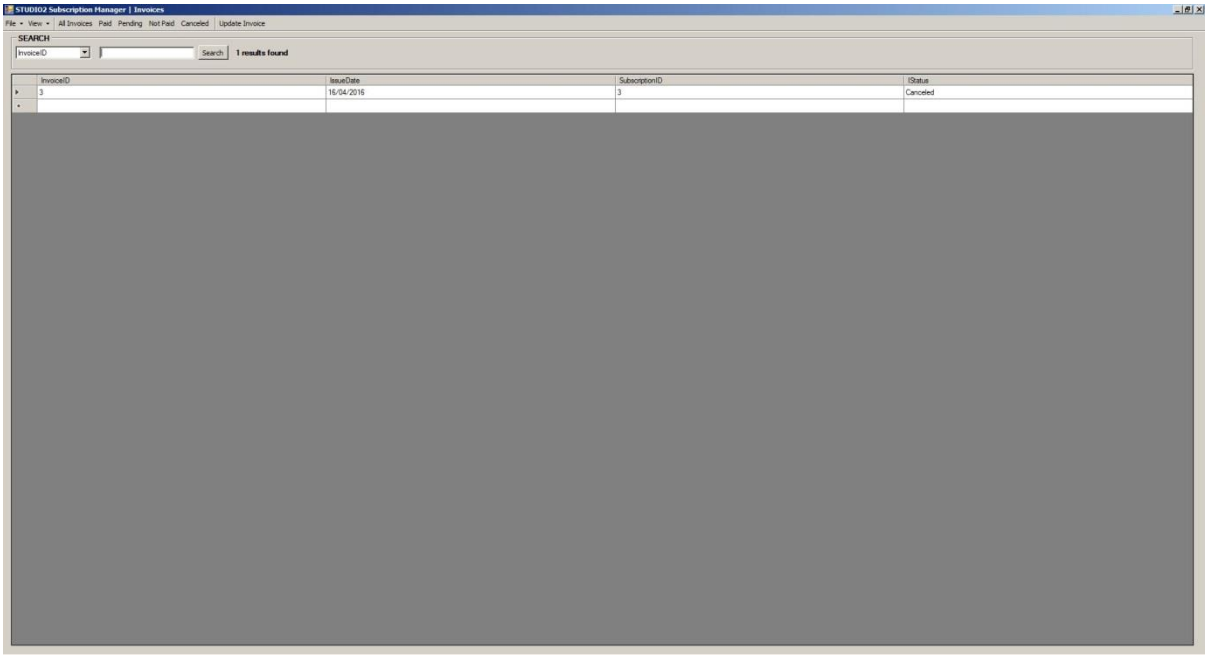
Test 4-15: Click 'Paid' ToolStripButton when grpUpdateInvoice GroupBox is visible



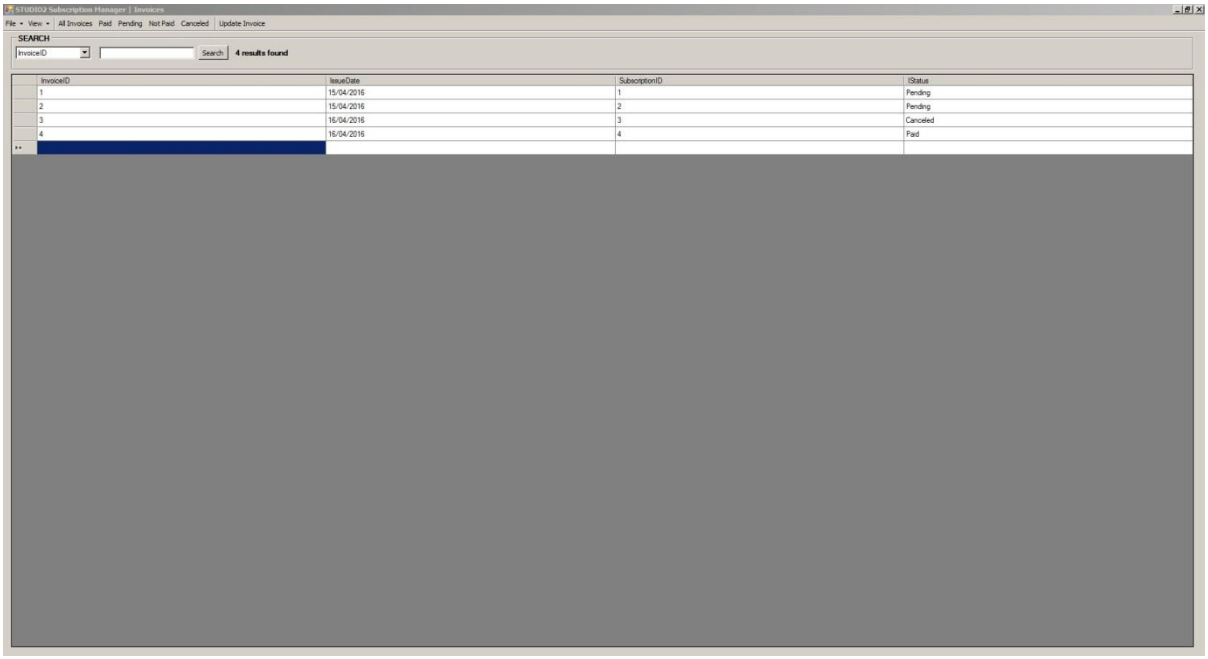
Test 4-16: Click 'Pending' ToolStripButton when grpUpdateInvoice GroupBox is visible



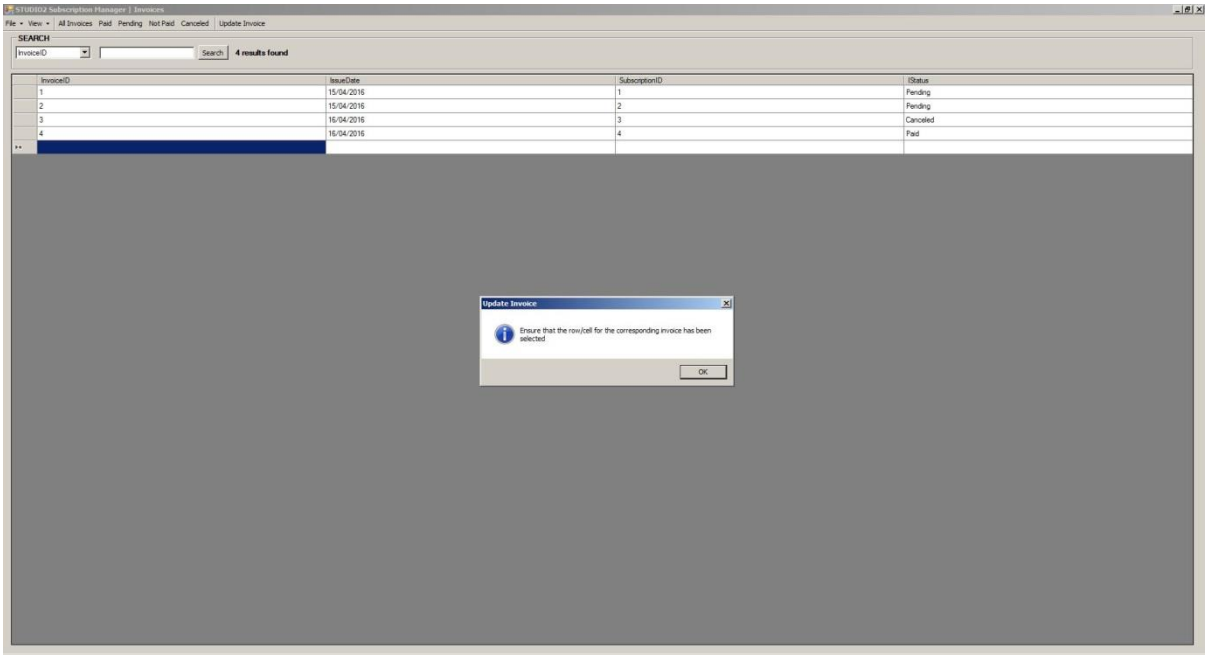
Test 4-17: Click 'Not Paid' ToolStripButton when grpUpdateInvoice GroupBox is visible



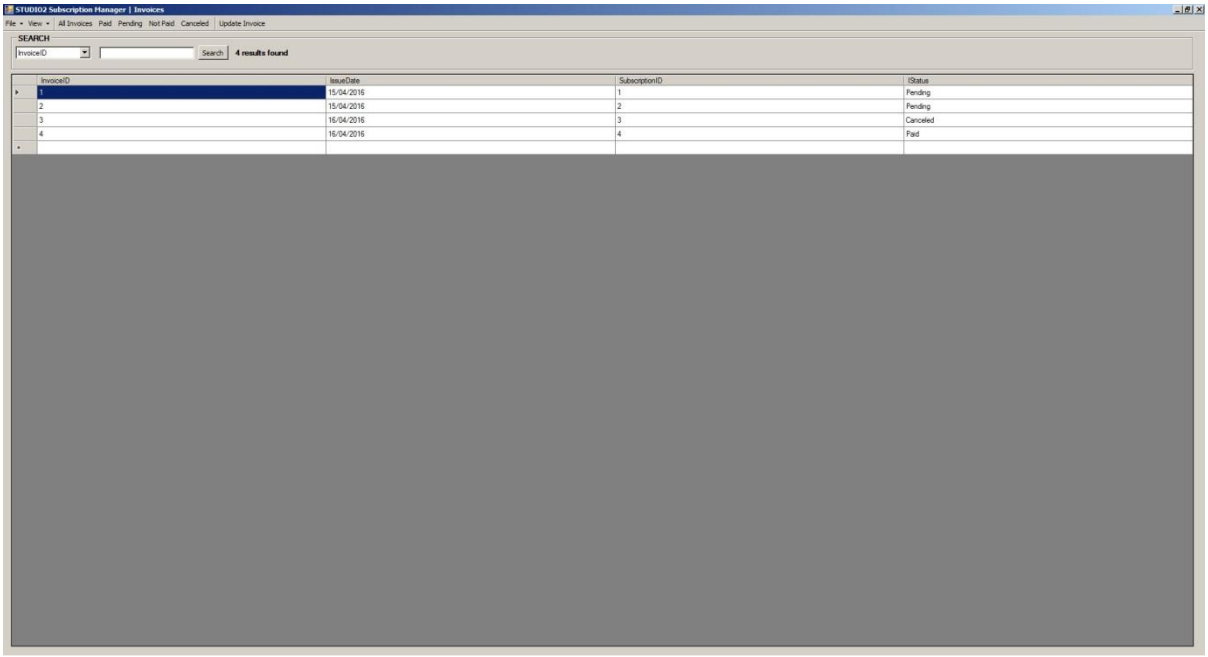
Test 4-18: Click 'Canceled' ToolStripButton when grpUpdateInvoice GroupBox is visible



Test 4-19: Click 'Update Invoice' ToolStripButton when an invalid cell/row has been selected

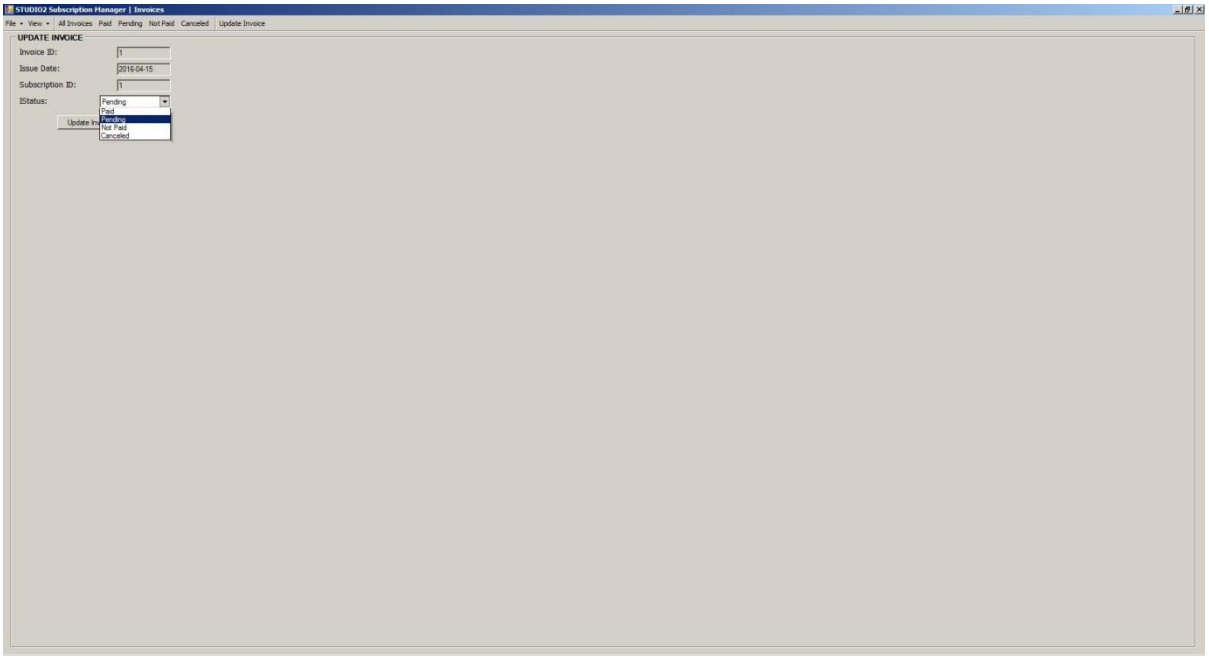


Test 4-19: MessageBox displays stating that a cell/row of the chosen invoice must be selected

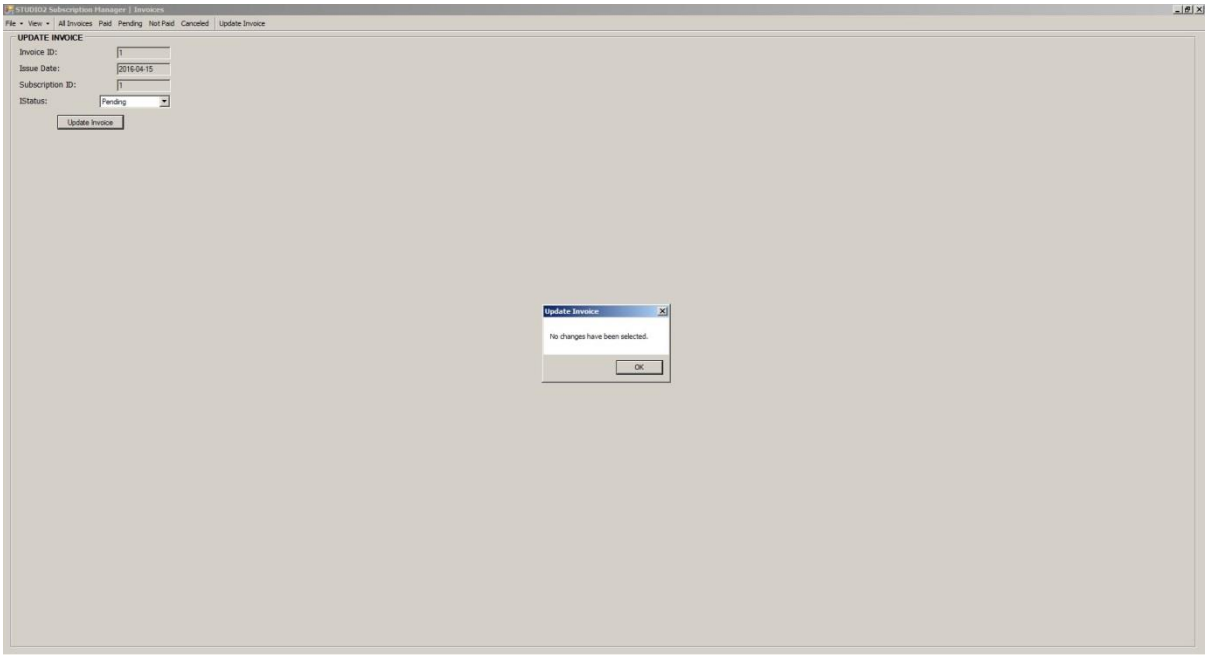


Test 4-20: Click 'Update Invoice' ToolStripButton when a valid cell/row has been selected

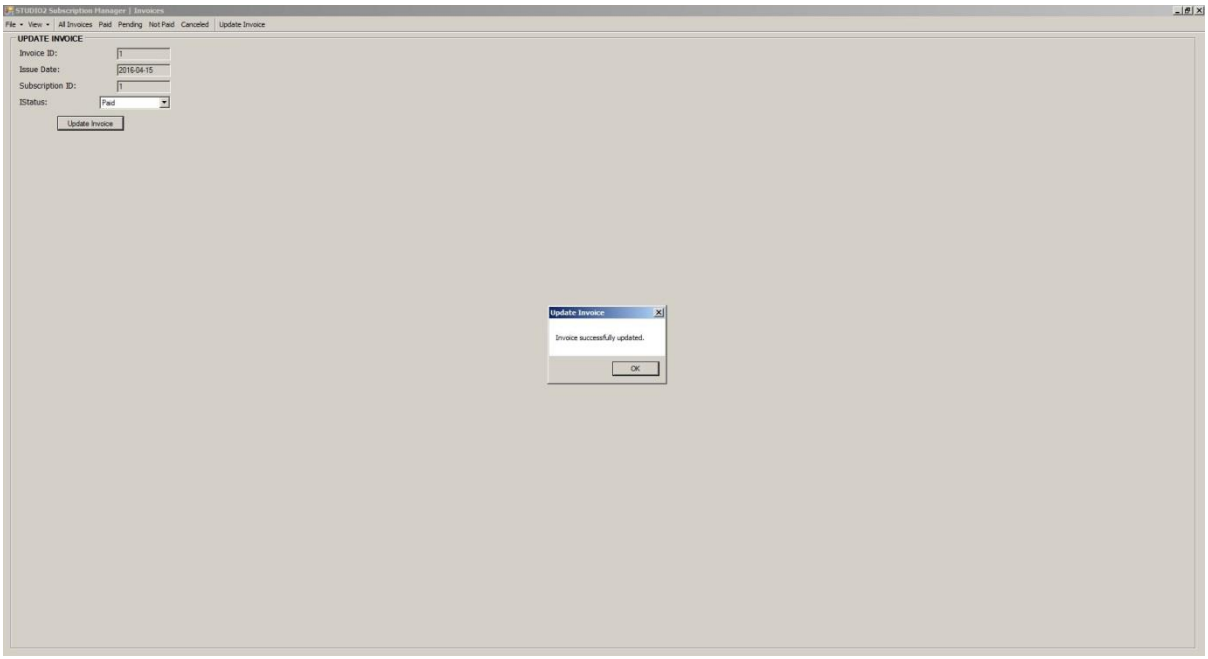
Test 4-20: grpUpdateInvoice GroupBox displays



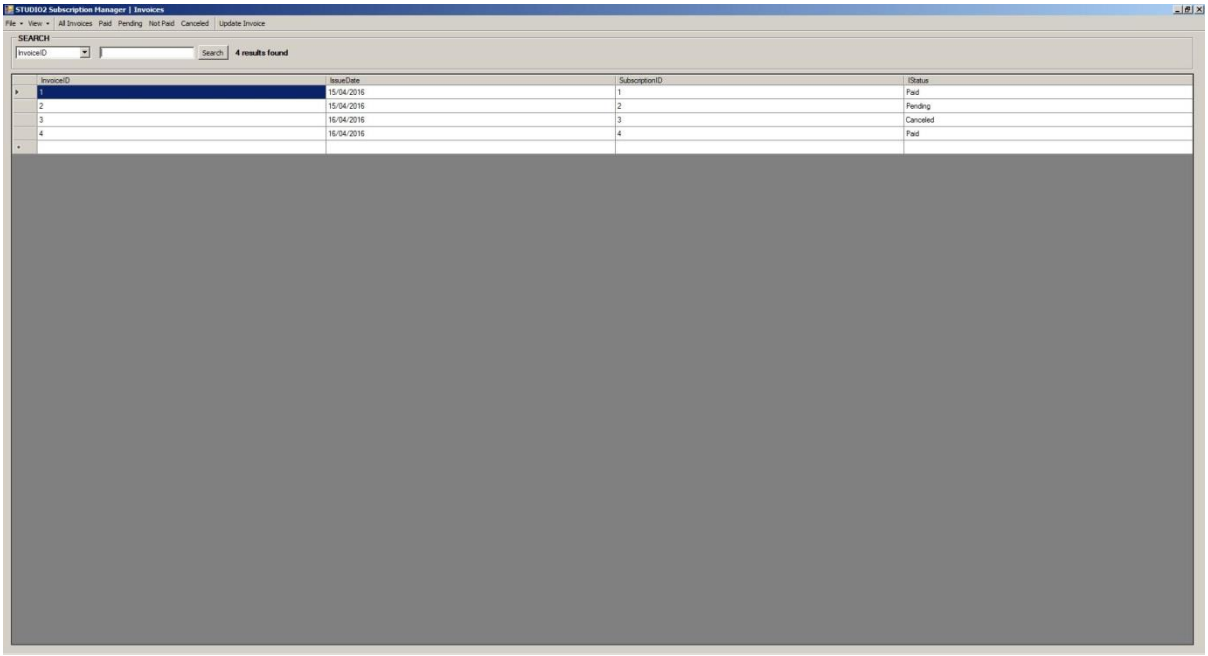
Test 4-21: Click cboIStatus ComboBox



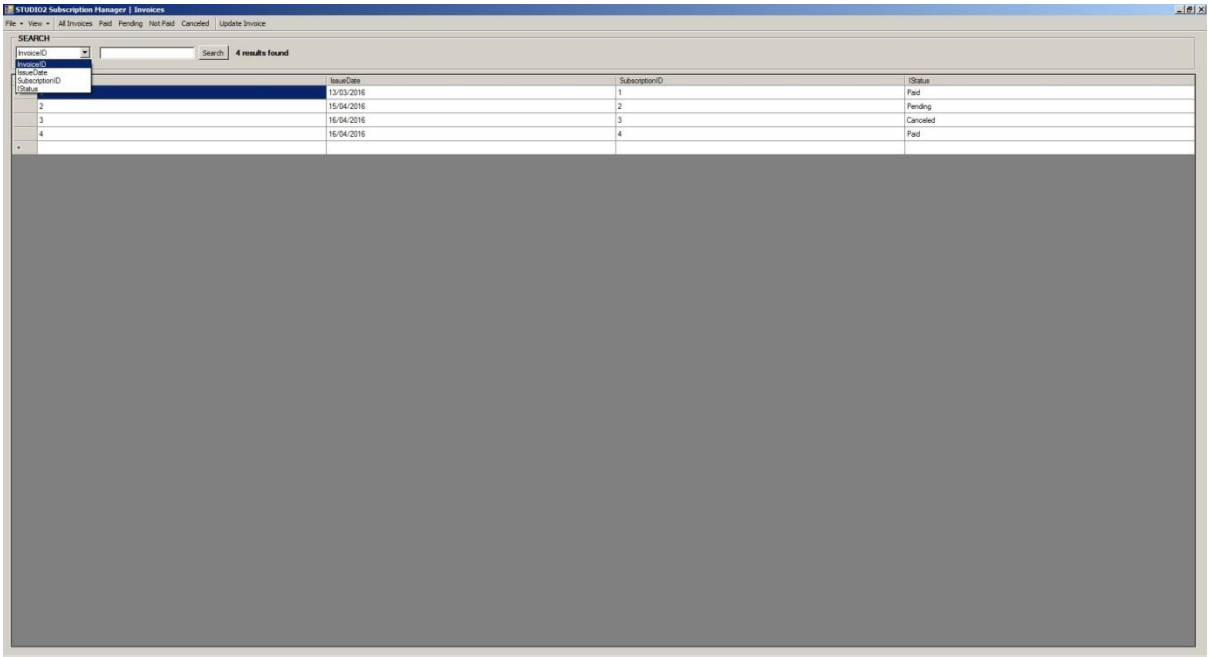
Test 4-22: Click 'Update Invoice' button when cboIStatus Text has not been changed to a different value



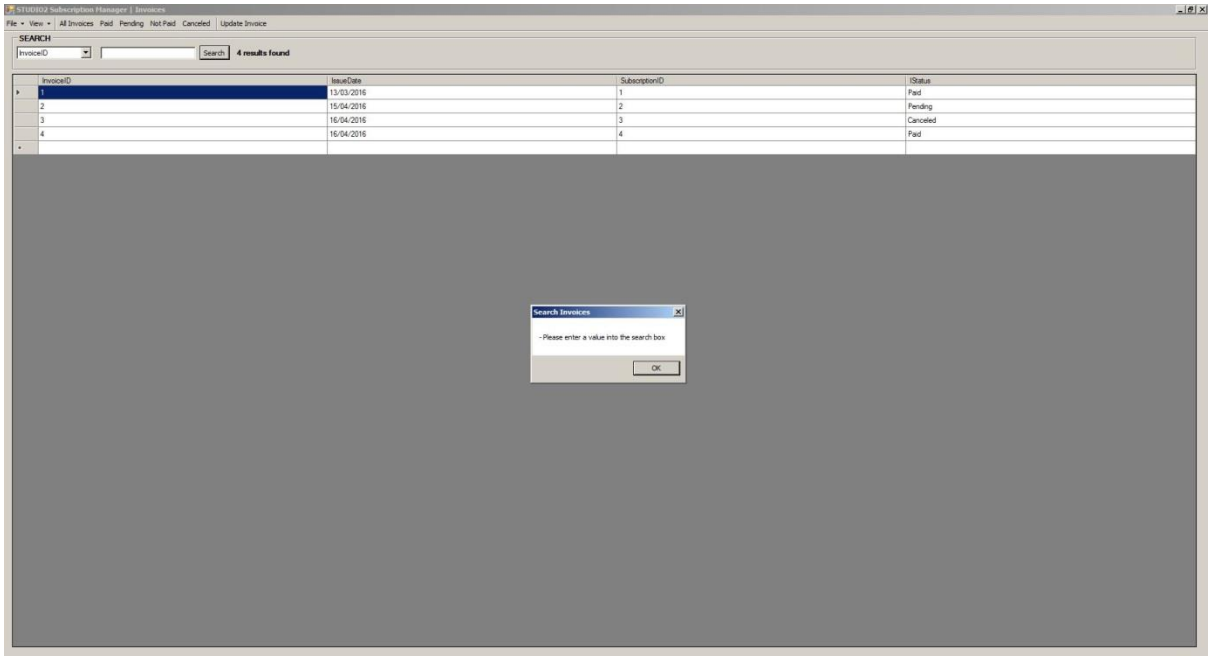
Test 4-23: Click 'Update Invoice' button when cboIStatus Text has been changed to a different value



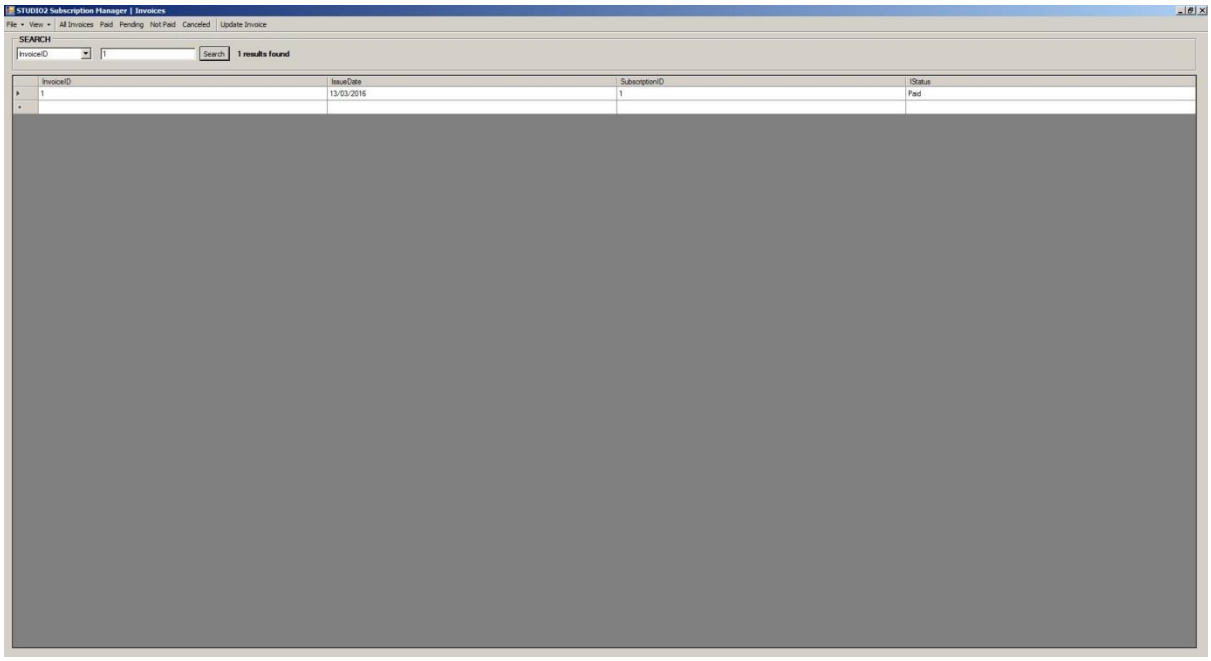
Test 4-23: Displays dgvSQLOutput with updated invoice



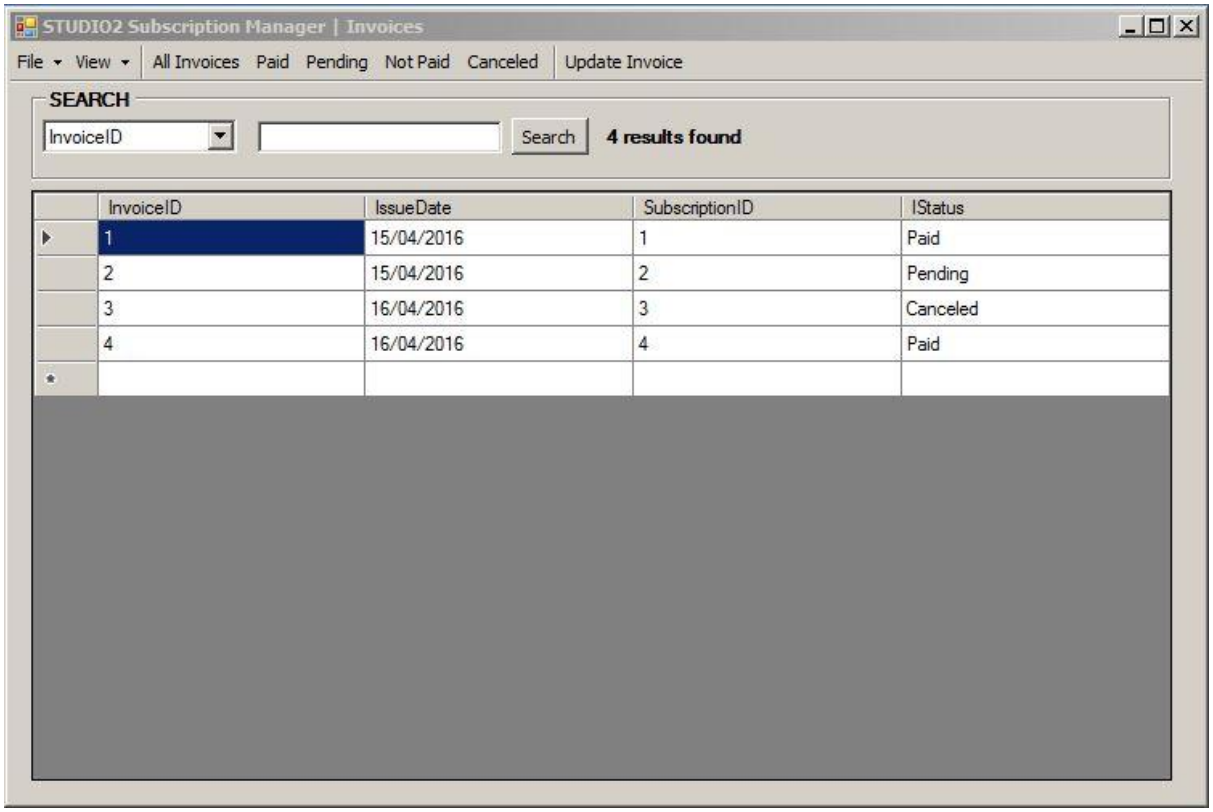
Test 4-24: Click cboSearchField ComboBox



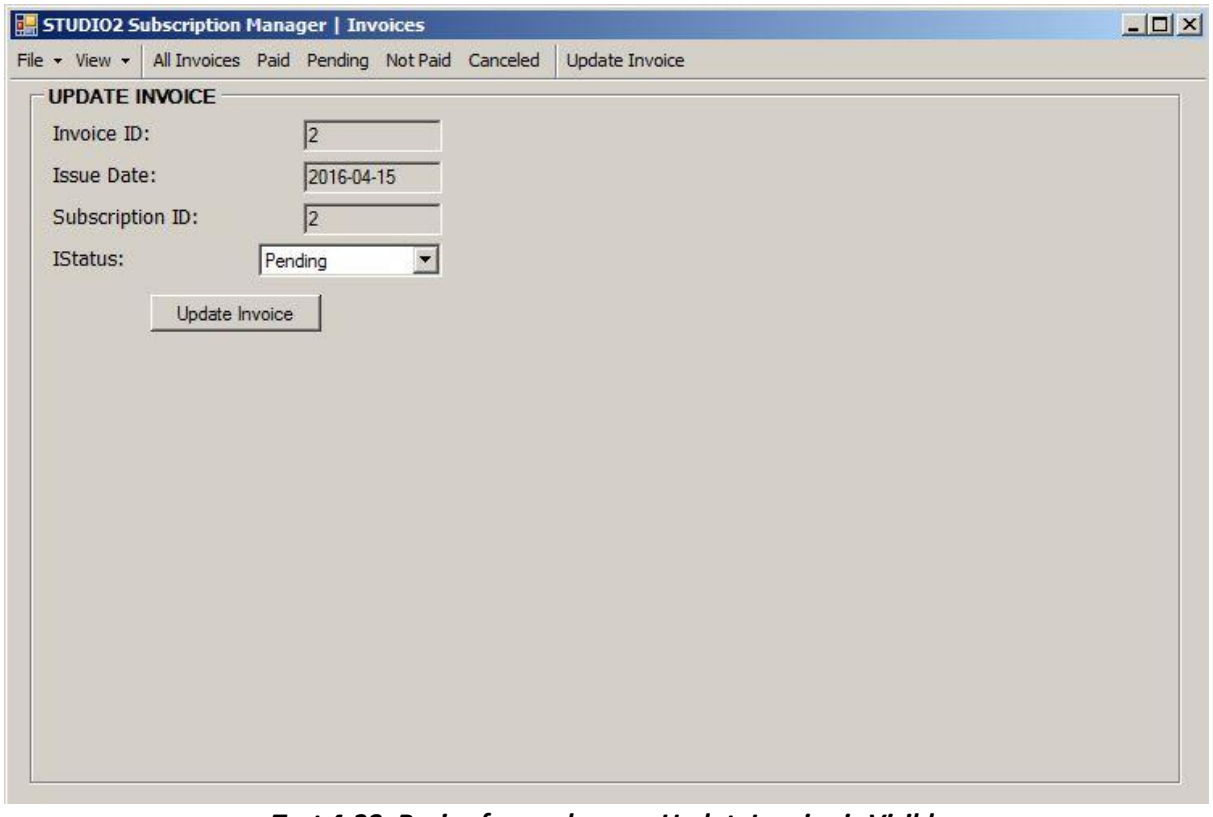
Test 4-25: Click 'Search' Button when no text has been entered into txtSearch TextBox



Test 4-26: Click 'Search' Button when '1' has been entered into txtSearch TextBox and cboField text is 'InvoiceID'

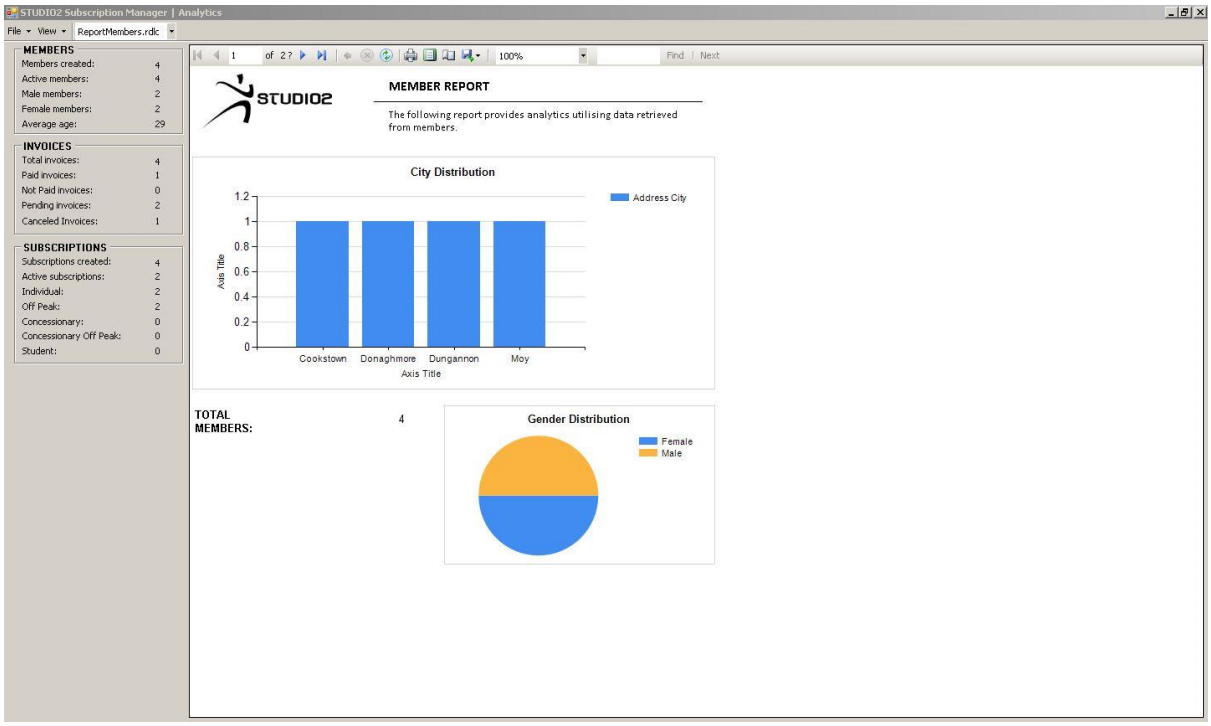


Test 4-27: Resize form when dgvSQLOutput is Visible

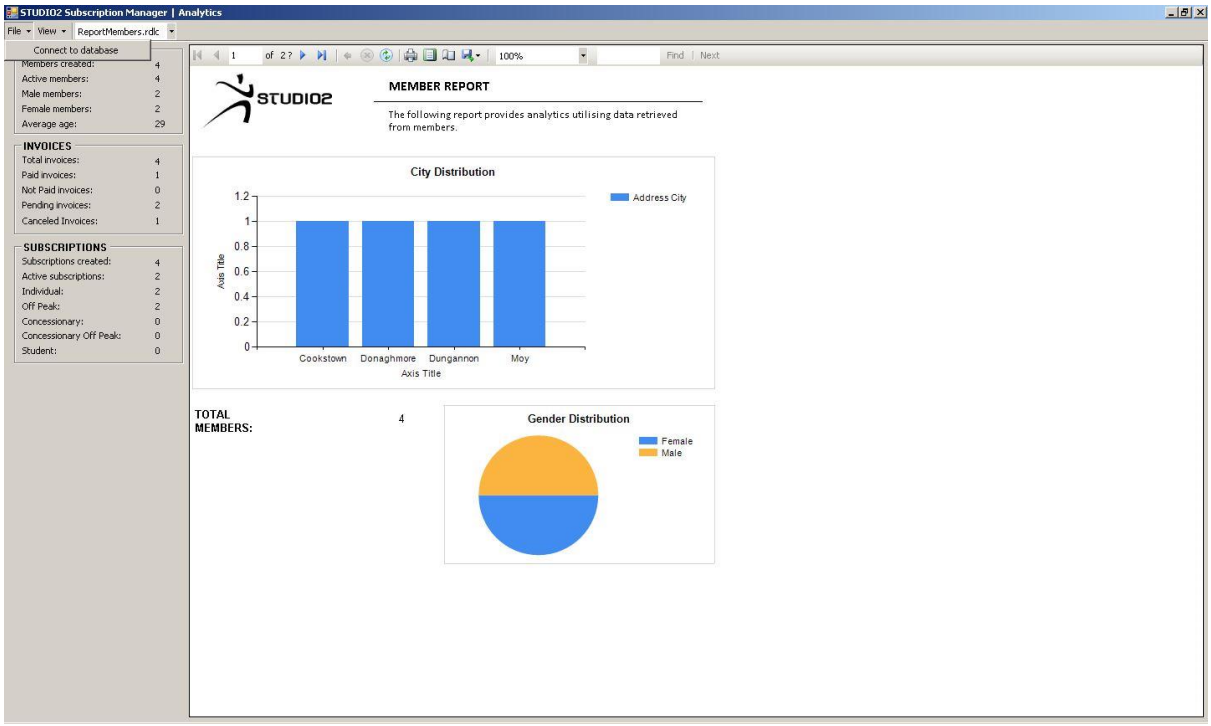


Test 4-28: Resize form when grpUpdateInvoice is Visible

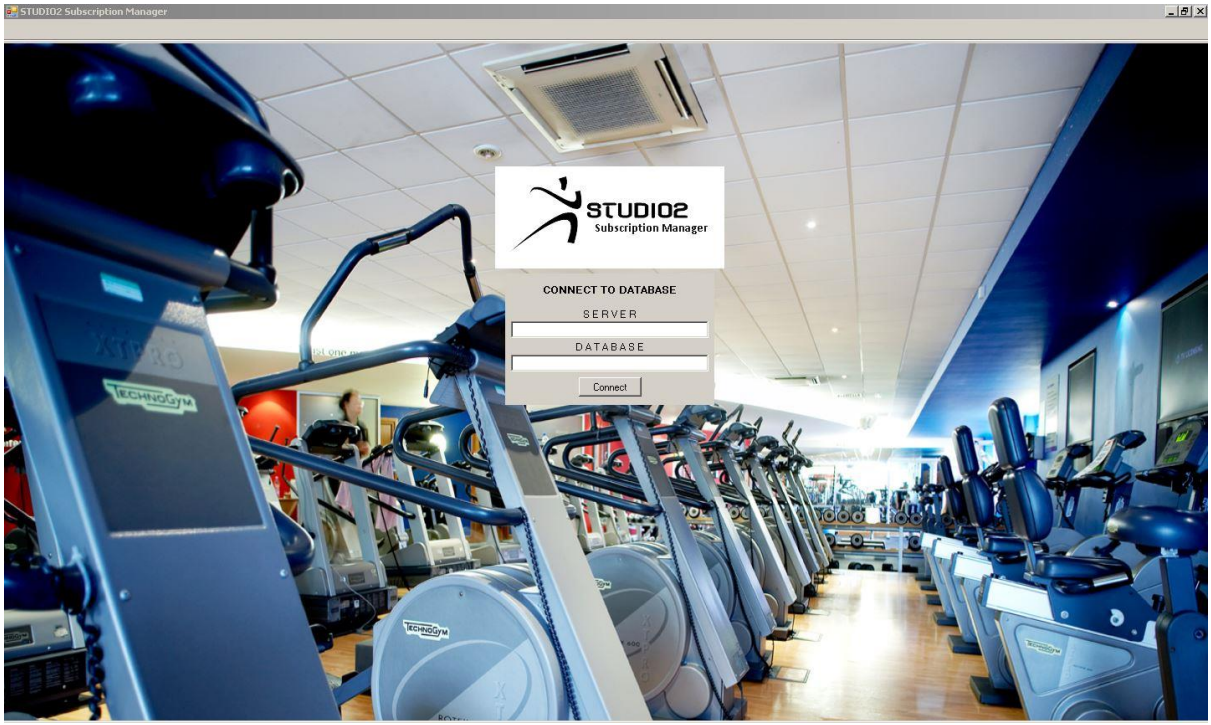
ANALYTICS FORM



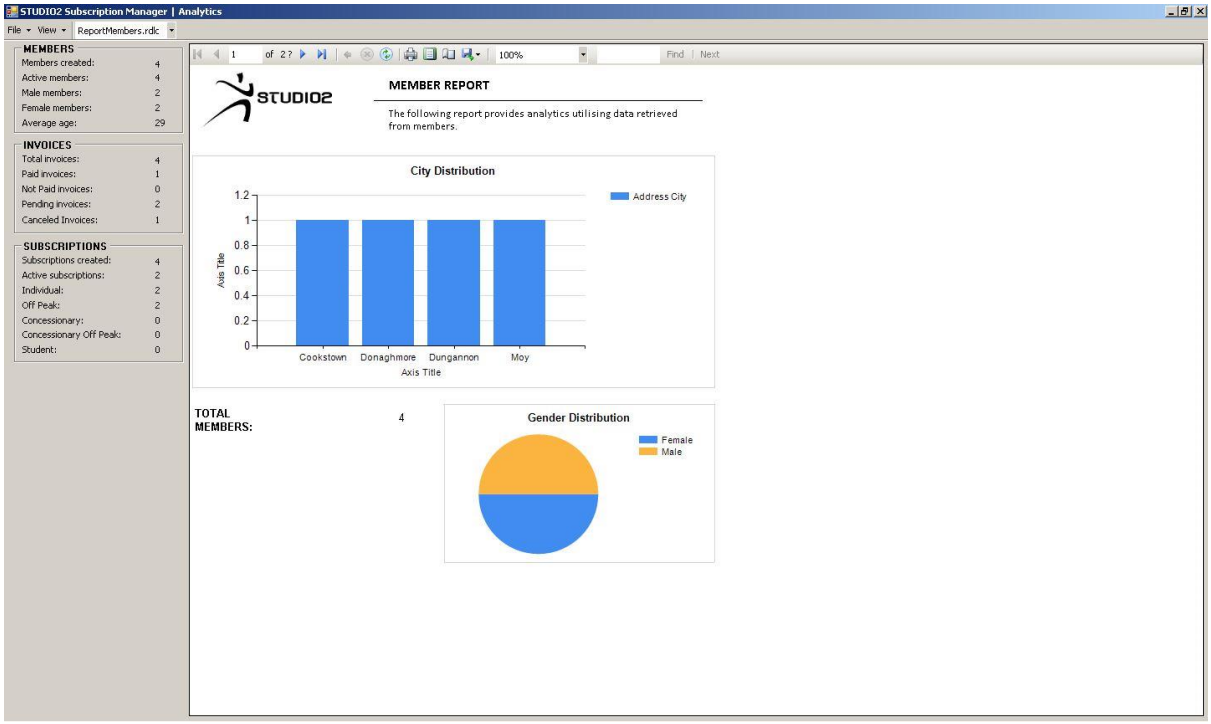
Test 5-1: Load 'Analytics' Form



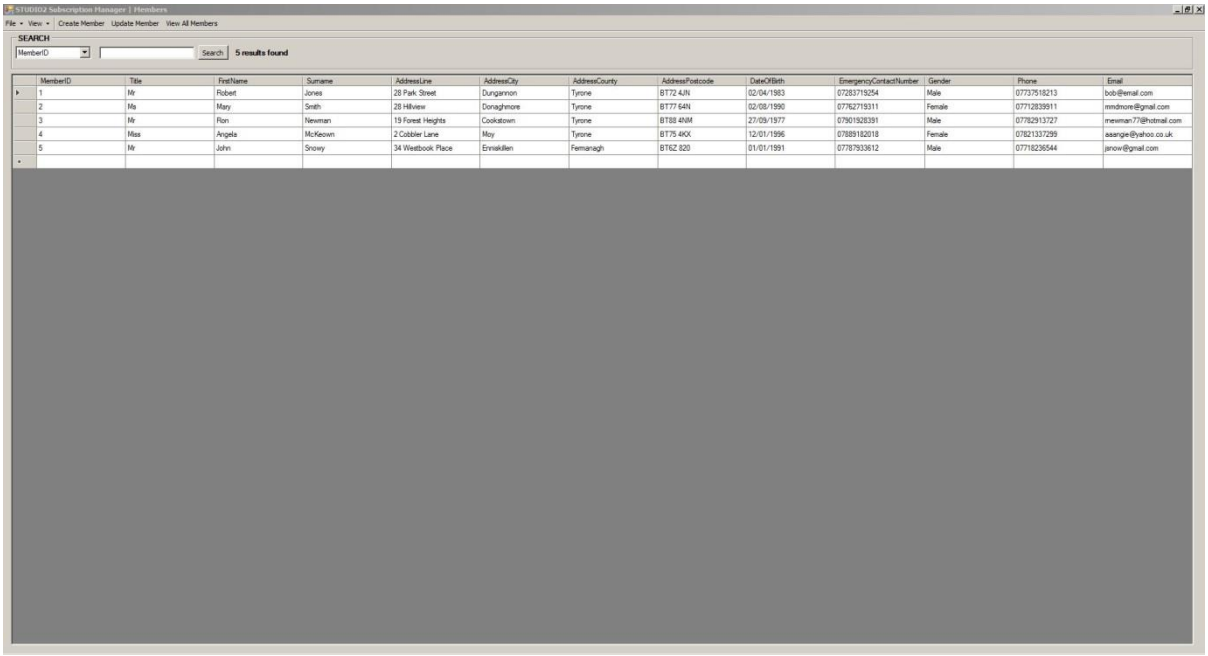
Test 5-2: Click 'File' ToolStripDropDownButton



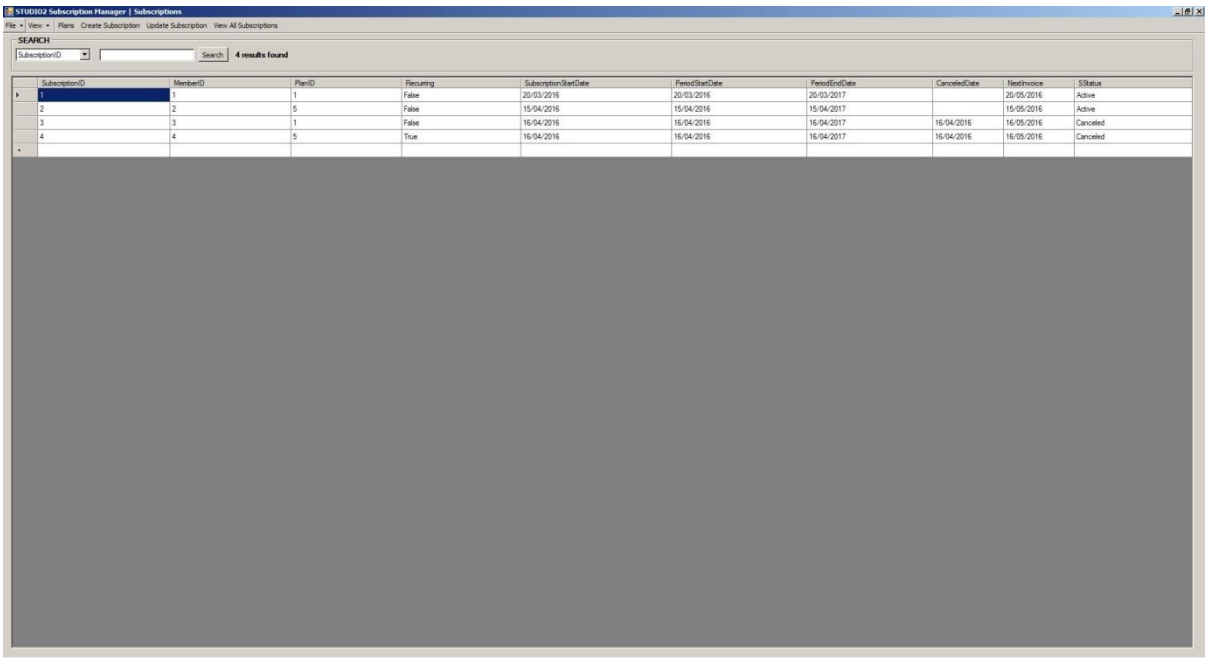
Test 5-3: Click 'Connect to database' ToolStripMenuItem



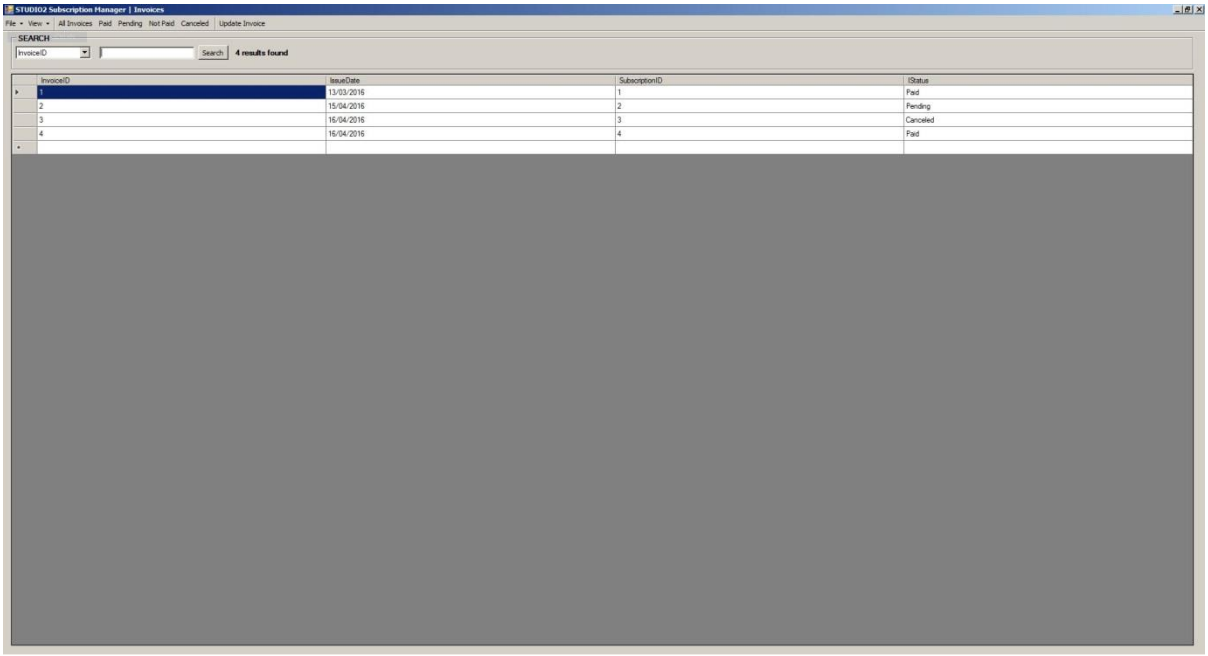
Test 5-4: Click 'Analytics' ToolStripMenuItem



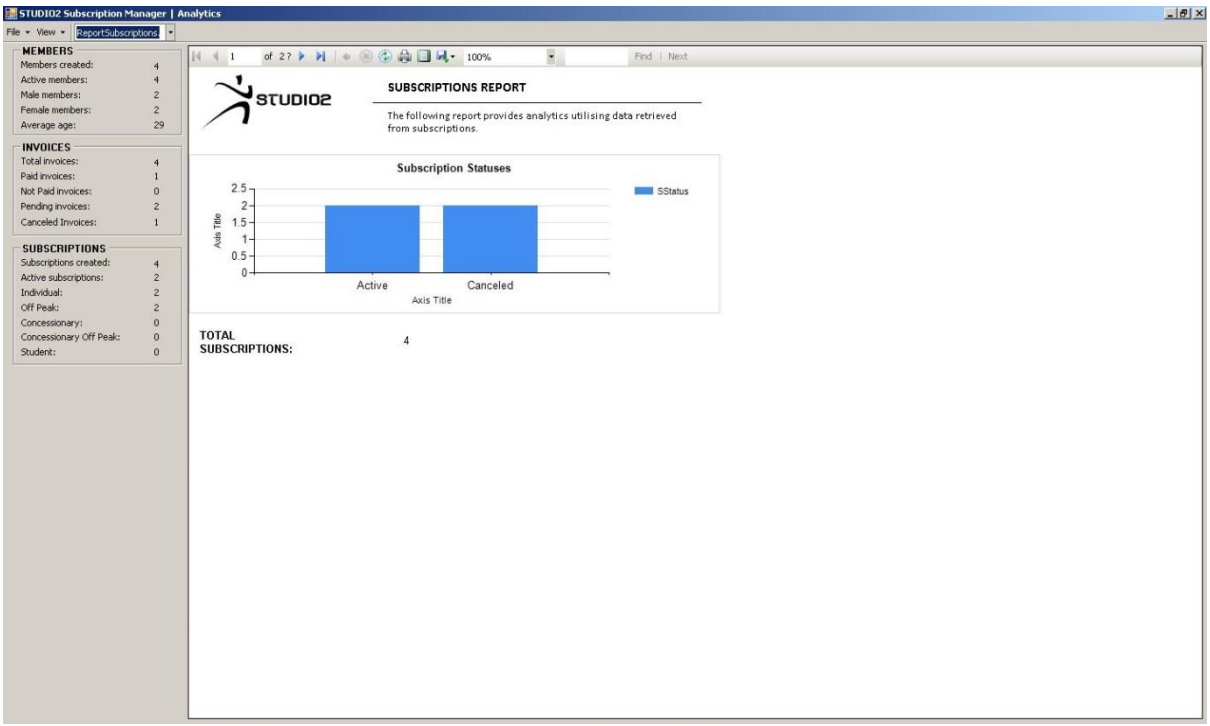
Test 5-5: Click 'Members' ToolStripMenuItem



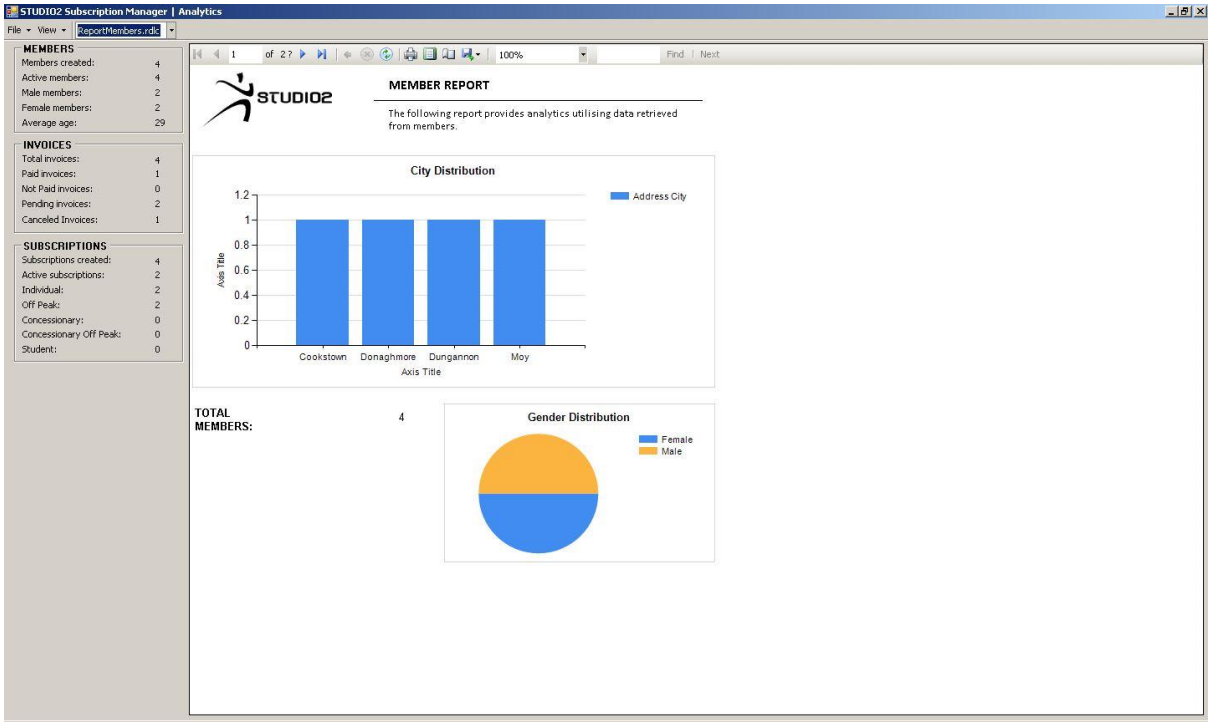
Test 5-6: Click 'Subscriptions' ToolStripMenuItem



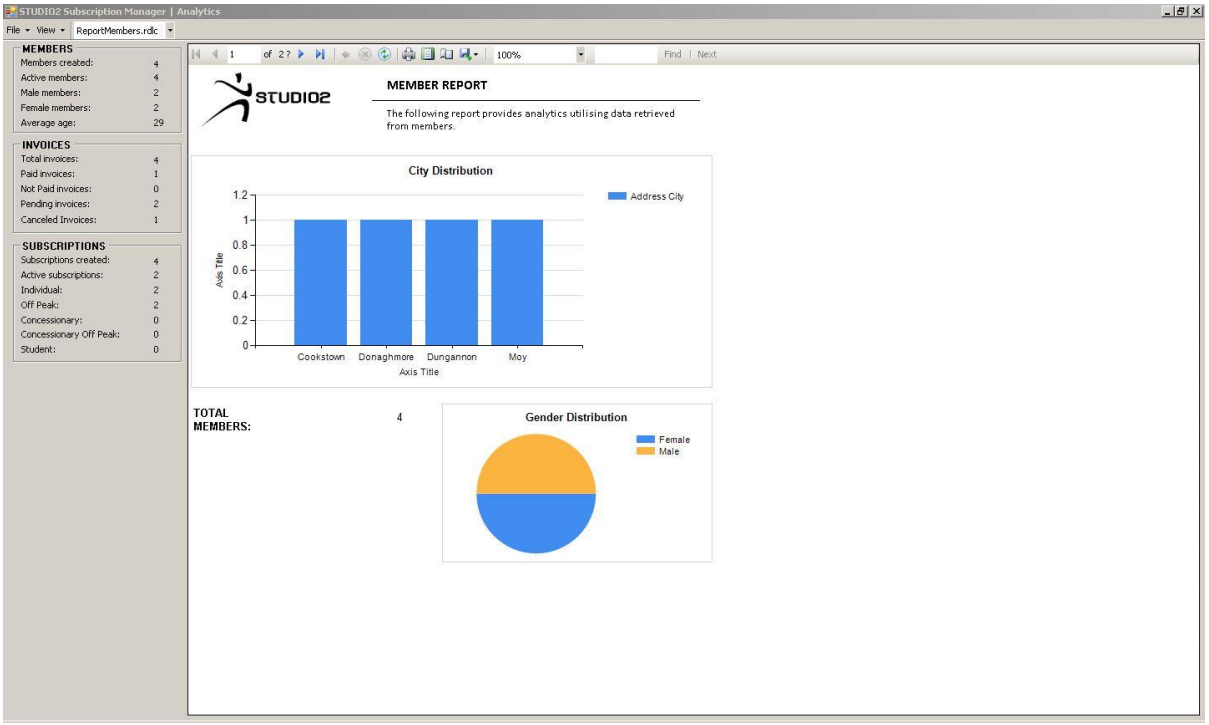
Test 5-7: Click 'Invoices' ToolStripMenuItem



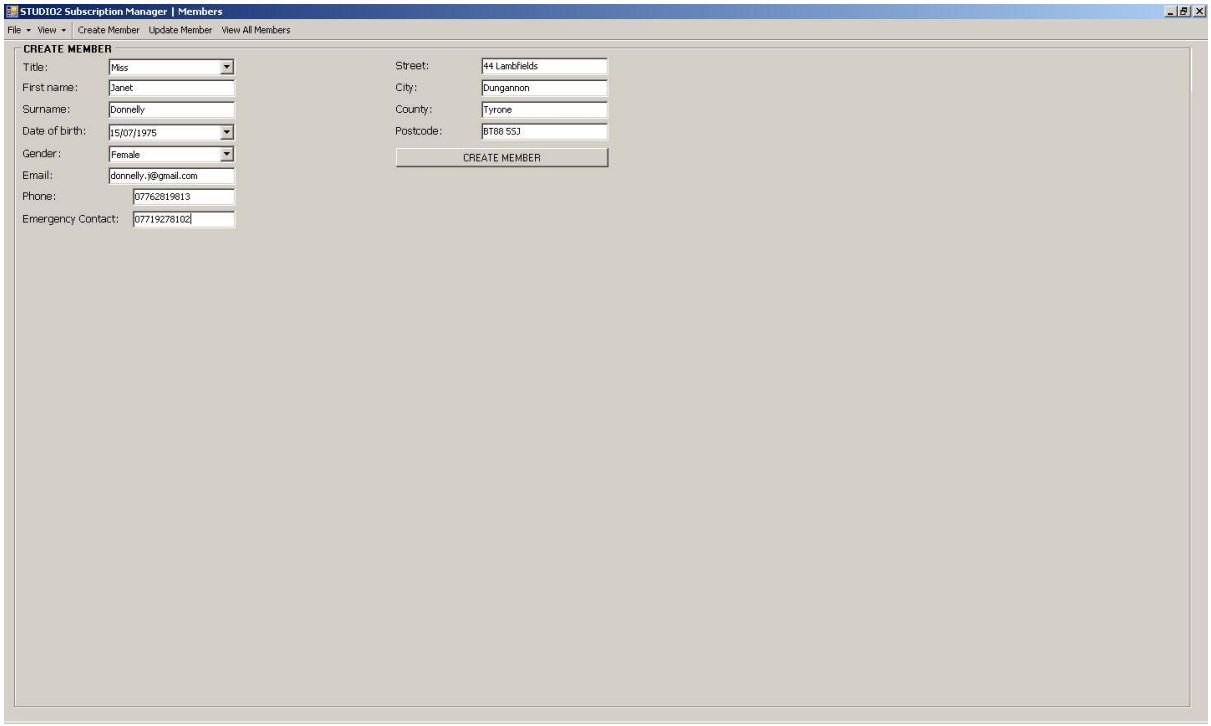
Test 5-8: Select "ReportSubscriptions.rdlc" in ToolStripMenuItemComboBox



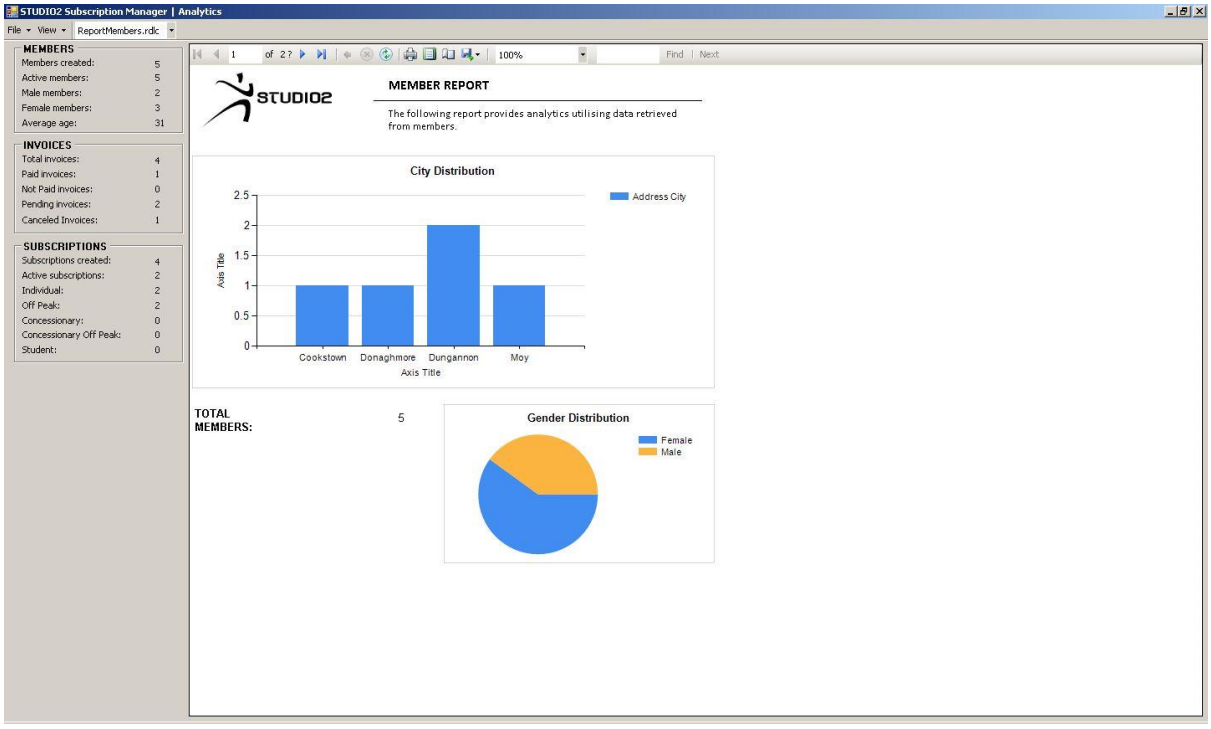
Test 5-9: Select "ReportMembers.rdlc" in ToolStripMenuItemComboBox



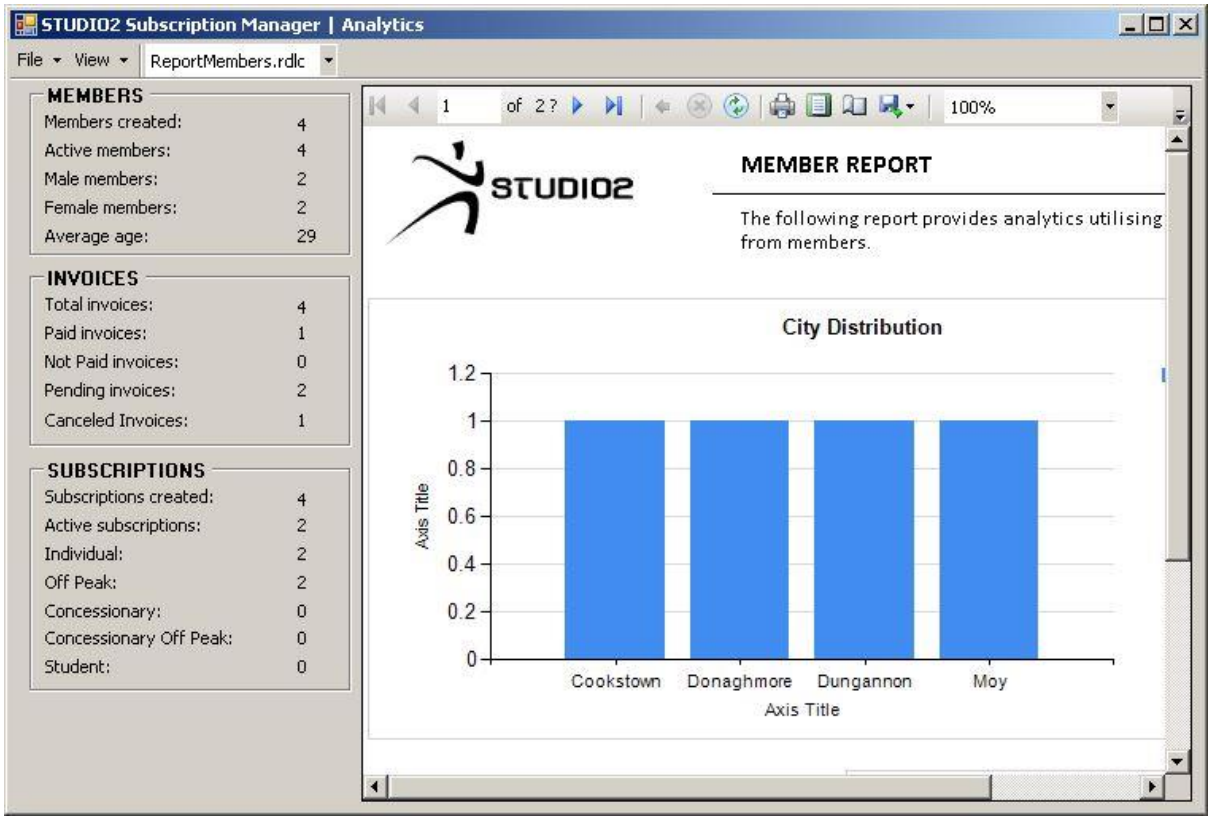
Test 5-10: Create new member in Member Form then move to Analytics Form to check that Member Report Viewer updates with new data



Test 5-10: Creating member record



Test 5-10: Report Viewer updated



Test 5-11: Resize form

E V A L U A T I O N

Evaluation of User Requirements

The most vital part of any project whether it is developing a software system or building a house, is to ensure that the core requirements outlined by the client are fulfilled. Across all types of business, the products which meet the desires of the customer will inevitably be the most successful. This success will be evident from the high levels of customer satisfaction and the achievement of healthy sales targets and profit margins.

Functional Requirements

- *SQL database must contain data for members, subscriptions, subscription plans and invoices*
    - I successfully created the SQL database containing the Member, Subscription, Invoice and Plan tables, effectively making the centre of the entire system around which everything else revolves
- *System must allow user to connect to a locally hosted SQL database*
    - By utilising connection strings within Visual Studio, I was able to build a GUI interface which allows the member to enter the Server and Database details of the SQL database, thus allowing them to connect to it. **(Test 1-2**)
- *System must allow members to be created*
    - The user is provided with the means to easily create a new member record which is stored into the Member table of the SQL database. Details of the member are entered into TextBoxes, ComboBoxes and a DateTimePicker control meaning that the member is not required to write any form of SQL query **(Test 2-13)**
- *System must allow members details to be updated*
    - Similar to the creation of a member, with the exception that an UPDATE SQL query is utilised instead of an INSERT SQL query. **(Test 2-20)**
- *System must display members in a table*
    - All records from the Member table are displayed in a DataGridView when the user loads the Members Form **(Test 2-1)**
- *System must allow user to search for member records which match a search query*
    - User is provided with the means of selecting a field from the Member table through the use of a ComboBox and then they enter the value which they are searching for in a TextBox. Any matches found will then display the entire record of the match. **(Test 2-26)**
- *System must allow subscriptions to be created*
    - The user is provided with the means to easily create a new subscription record which is stored into the Subscription table of the SQL database. A Button is provided which redirects the user to the Member Form to select the member for whom the subscription is to be created for. The plan is then selected from a ComboBox and the user simply clicks a button to create a subscription **(Test 3-24)**

- *System must allow subscriptions to be canceled*
    - When updating a subscription, a cancelation Button is displayed enabling the user to cancel a subscription at any time. **(Test 3-33 / Test 3-34)**
- *System must display subscriptions in a table*
    - All records from the Subscription table are displayed in a DataGridView when the user loads the Subscriptions Form **(Test 3-1)**
- *System must allow user to search for subscription records which match a search query*
    - User is provided with the means of selecting a field from the Subscription table through the use of a ComboBox and then they enter the value which they are searching for in a TextBox. Any matches found will then display the whole record of the match. **(Test 3-38)**
- *System must allow invoices to be created*
    - Invoices are automatically generated by the system when the user creates a new subscription. They are also generated during the initial connection to the database for any subscriptions which require a new invoice to be issued and when a new subscription cycle is started **(Test 3-24 / Test 1-3)**
- *System must display invoices in a table*
    - All records from the Invoice table are displayed in a DataGridView when the user loads the Invoices Form **(Test 4-1)**
- *System must allow user to search for invoice records which match a search query*
    - User is provided with the means of selecting a field from the Invoice table through the use of a ComboBox and then they enter the value which they are searching for in a TextBox. Any matches found will then display the whole record of the match. **(Test 4-26)**
- *System must display subscription plans in a table*
    - All records from the Plan table are displayed in a DataGridView when the user clicks the “Plans” ToolStripMenuItem in the Subscriptions Form **(Test 3-11)**
- *System must allow prices of subscription plans to be changed*
    - ComboBox is provided allowing user to select the chosen plan. The new price is then entered into a TextBox and a Button clicked updating the corresponding plan record with a new Amount value **(Test 3-16)**
- *System must automatically identify when invoices need to be generated and carry out this task*
    - During the initial connection to the database, the system issues invoices for any subscriptions with a due payment and when a new subscription cycle is started **(Test 1-3)**
- *System must automatically identify invoices which have not been paid and cancel the associated subscription*
    - During the initial connection to the database, system checks for any pending invoices that are more than 7 days old and consequently sets their status to ‘Not Paid’ and suspends the subscription **(Test 1-3)**
- *System must allow user to print out records from any table*
    - In Members, Subscriptions and Invoices Forms, a ToolStripMenuItem is provided which when clicked, allows the user to print the records shown in the DataGridView

visible. When a search query has been executed, it also allows the search query and the results to be printed. **(Test 2-4 / Test 3-4 / Test 3-6 / Test 4-4**)

- *System must provide detailed analytical data in the form of graphs and statistics*
    - Analytics are provided in Report Viewers in the Analytics Form and are also provided within GroupBoxes separate to the Report Viewers. As a result of time constraints, I was only able to integrate simple analytics **(Test 5-1)**

Non-Functional Requirements

- *A user friendly Graphical User Interface (GUI) should be created using Windows Forms*
    - In developing the system, the high-level interface used by the user is of critical importance as it is the middle layer that allows them to interact with the database. For this reason, I placed particular emphasis on creating a clean and simple interface that can be easily understood and used effectively. On top of this, I incorporated a responsive design to the interface to ensure that the system would display properly on a monitor of any resolution (however it is not designed to display properly on mobile devices). **(Test 1-9 / Test 2-27 / Test 3-39 / Test 4-24 / Test 5-11)**
- *Try Catch blocks should be utilised to catch errors and prevent the application from unexpectedly crashing*
    - I did not use Try Catch blocks for every block of code within the system however I instead focused on using them in areas prone to error. This was for the most part inside the methods of each Data Access Layer class where the system interacts with the SQL database and thus is most susceptible to encountering problems. In doing so the stability of the system is improved and it is less likely that it will crash during run-time.
- *Code should be commented properly to allow for maintenance and maximum reusability*
    - The code has been commented throughout in order that any future developers can easily access and understand the workings of the system should they decide to perform any maintenance or improvements to the system. Had the system been deployed in a real world scenario, this specific project would have been a module of the overall STUDIO2 system and thus the use of comments would provide the other developers with ease in integrating the different modules together.
- *Application should load and perform operations within a reasonable time*
    - The application itself loads fairly quickly and executes SQL queries incredibly fast due to the fact that the SQL server is local and not web based. I made use of reusing of code where it was suitable and when I had the time to do so however I believe that any perfective maintenance performed in the future would find areas where replicated code could be condensed. A good example of this would be when the user navigates to a different form form as to do this I repeated the same blocks of code in each form instead of when it may have been possible to create a class to handle the creation of new form instances and hiding of the current form.
- *Appropriate measures should be taken to provide validation*
    - Some validation was added to the system such as displaying a MessageBox when the user attempts to search for something without entering anything into the search box or when the user enters an age less than 16 when creating / updating a member.

However, I did not implement any other validation in the create member / update member fields in the application which is the area where it would be most important as there are multiple types of fields such as names, mobile numbers and addresses. Although it is highly unlikely and illogical to do so since the user has access to the SQL database through SQL Server Management Studio, the lack of validation in these areas makes the database susceptible to SQL injection. However since the details are entered by staff as opposed to the user inputting member details themselves, this is not a particularly serious problem.

The finished application typically uses between 15MB and 50MB (when viewing analytical reports) making it suitable to run on any PC, regardless of whether it is low or high spec. This will mean that STUDIO2 will not be required to purchase additional hardware in order to run the application. The SQL server itself also takes up a small amount of RAM however as the database grows, it is likely that the RAM allocated to the SQL server would also increase. This issue could be rectified by simply purchasing more RAM to install into the computer or perhaps by purchasing a standalone server. Though it is important to take note that the application has been designed to operate with a local SQL server and so has not been tested using a SQL server connected via intranet or internet, meaning that there is the possibility the application would not function properly. In addition, as was also addressed in the user requirements, the .NET framework is still required in order to run the application.

Evaluation of Approach to Solution

Generally, I used a Waterfall approach for the project. At the early stages I focused on creating user requirements (Analysis stage). I then used these to develop the data fields which would be required for the database and created storyboards based around these (Design stage). During the development stage, I implemented many changes to the original plans resulting in cycling back to the design stage in order to re-evaluate my data model and the different tables and fields present in the database. In particular, I made fairly significant changes to the GUI compared to what was originally planned with the storyboards. The implementation of various changes during development meant that in some ways an Agile approach was also utilised. Occasionally when problems were encountered, I co-operated with peers to overcome these so it could be suggested that an XP approach was used to some extent as well. In terms of the project overall, Waterfall was generally the approach used as is evidenced by how for the most part, the project generally followed a structured approach in which each stage was only started after the previous completed i.e. design after analysis, testing after development and such.

Evaluation of Project Plan

Many different obstacles arose during the development of the system. One of the most critical was the necessity to allocate time throughout the course of the project to focus on software systems development theory. I had not properly considered the extent to which this would impede on the project and thus this impacted on the allocated time frames on my Gantt chart.

Since I was only introduced to SQL this year, and in particular, integrating C# and SQL together, extra research was often required in order to fully understand the low-level processes required to make

the system function as intended. This inexperience in using SQL and C# together meant that unexpected errors were often encountered during development ranging from minor issues such as forgetting to open a connection to the SQL database when executing a SQL query programmatically from the application , to formatting queries incorrectly which resulted in SQL errors. However by encountering and resolving these problems, I was able to learn from my mistakes and quickly rectify any similar problems that arose at later stages. It also equipped me with the knowledge to assist peers who faced similar problems.

Due to only being made aware of the Report Viewer at a late stage of development, I had to completely change my Analytics form to incorporate the Report Viewer. The lack of many resources regarding to the Report Viewer compared to SQL or C#, meant that quite a lot of time had to be spent attempting to achieve different things by trial and error. This along with time and the lateness with which I was made aware of it meant that I was unable to utilise the Report Viewer to its full capability.

Another issue encountered was the Programming Environment. Quite often I would have to spend 10-20 minutes trying to launch the application and in some cases, even more time had to be spent trying to connect my USB drive. This meant that by the time I started doing any actual work, as much as 15-20 minutes could have passed, and as this often occurred, I encountered a notable loss of time.

Evaluation of Testing Procedures

The strategy I employed for testing was to continually test whatever element of the system I was working on while I was developing it. I chose to do this to ensure that when it came to the system testing, the system would actually compile and tests could be carried out.

The testing allowed me to identify various areas of the system which required amendment. For example I encountered a logical error in which when a members details were updated, their date of birth was always set to 01/01/1991. There was no Exception error being returned indicating that the issue was not related to the execution of the query. I checked to see if the wrong value was being passed into the UpdateMember() dateOfBirth argument however the value from the dtpUpDateOfBirth was being used as intended. To determine where exactly the code was going wrong, I inserted breaking points into the UpdateMember() method in Member_DAL.cs as this is the method which handles the execution of the SQL query. In doing so I was able to determine with certainty that the problem was not caused by any sort of SQL error as before the command was executed the string containing the query showed ‘01/01/1991’. So I checked the statement I used to build the query string and it was here that I found that I had left ‘01/01/1991’ as a placeholder value, likely to act as a test harness while I was working on age validation, instead of using the dateOfBirth argument passed into the method. The process of using breaking points and checking exceptions to determine exactly what a problem was, allowed me to take remedial action on any issues that arose during the course of testing.

For each test I also provided screenshots as evidence to prove that the test was conducted and to show that any remedial action taken rectified the issues which were highlighted. Having to document each test, provide screenshots and insert these screenshots into the documentation

meant that a sizeable amount of time had to be spent on testing. However with the system being built through a Waterfall approach it is necessary that extensive testing be done to produce a stable high quality system, since this is a core principle of Waterfall.

Evaluation of Own Performance

Personally, I believe that the aspect of this project which I had to manage the most critically was time. The time frames outlined in the Gantt chart were only predictions and thus did not take into consideration any unexpected obstacles that would arise. Since a Waterfall approach is what I used instead of Scrum or XP, I had to tackle any obstacles encountered myself rather than with a scrum master when Scrum. This caused some minor setbacks which when added up did have a fairly sizeable impact on the development. In addition to these and taking into consideration work to be done for other subjects, only a certain amount of time could be made available to focus on the project and also meant that I had to be careful not to constantly prioritise the project over other subjects work.

Throughout the entire project, I developed many valuable skills. One of the most important skills was a fundamental understanding of SQL which now that I have, can be built upon in the future and would be a skill that could potentially be used across a wide range of jobs from data analysis to marketing. The continued use of C# and Visual Studio has also helped to improve my proficiency in programming in C# and helped me find ways to take advantage of the wide array of tools made available by Visual Studio to build a high quality system in an efficient manner. In order to improve the readability of the code both for development purposes and potentially for perfective maintenance in the future if the project was created as part of a real world scenario, I provided brief comments throughout the code explaining the functions of different blocks of code. I also utilised industry standards when naming objects and variables so as to reduce the need for a large amount of commenting and again make the code more readable both for myself and for anyone else who goes on to use it.

From the initiation stage to the completion of the project, I continually backed up both my documentation and the actual application. In doing so I minimised the risk of an unexpected event occurring such as the loss of my USB drive or the corruption of a group of files. Had something such as this occurred and the data lost was from tasks executed on the critical path, the time scale of the project would have to have been extended which in turn would serve to increase the overall cost of the project, as is evidenced through the use of the project manager's 'Iron Triangle'.

Evaluation of Solution

Strengths

Within the finished solution there are areas which I believe were of particular strength. The implementation of all the core user requirements means that the client's desires have been fulfilled. In addition, the system produced, through extensive testing, appears to be relatively stable. For the GUI, I differentiated quite far from the original storyboards, but in doing so I believe that the improved GUI provides a cleaner interface and thus, will not only provide the user with a better experience but also assist them in operating the system in an efficient matter. I also implemented a

responsive design into my GUI so that when the Form is resized, controls inside the Form will resize and relocate appropriately to ensure that they remain visible to the user and to maximise the space used by them, particularly in regards to the DataGridViews. The finished application typically uses from 13MB to 50MB of RAM, making it suitable for any computer system which has .NET framework installed.

Weaknesses

There are parts of the system which in my opinion could have been better. As a result of time constraints and inexperience in the utilisation of the Report Viewer, I was only able to create simple reports as opposed to the advanced reports utilising complex SQL queries which I had originally planned to do at the beginning of the project. The simplicity of the reports means that if STUDIO2 wished to properly analyse the data, they would have to send the data to an external company or data analyst to provide them with the analytics, which would likely impose a notable expense on the business. In turn, the absence of an advanced analytics module depreciates the overall value of the system to the client, which may impact negatively on the customer's satisfaction and the likelihood of repeat sales. The system also lacks the ability to create new plans which is something which could very possibly be wanted by STUDIO2 in the future. In particular the ability to create new plans may be used for special promotion purposes such as trial subscriptions.

Enhancements

There is no notification system in place to notify either staff or the members of STUDIO2 with an active subscription of when an invoice has been issued or a subscription is near its expiration/recurring date. However considering that the system I have developed is actually only a module of a much more advanced system, it could be argued that notifications could be a standalone module within the system for notifications about subscriptions, classes and rooms. If it were implemented, the use of an email service may provide an effective and inexpensive means of sending notifications or perhaps if STUDIO2 were to grow large enough, a mobile companion app of some sort could be a further option. Another aspect that I believe would enhance the system is to completely encapsulate all code within various Try Catch blocks and to then create a system to log any errors which occur, possibly into a text file, so that any bugs or errors can be automatically recorded and reported to the developer. One part of the system that I was slightly unsure about was how a change in the price of a plan should affect any active subscriptions on this plan. Does the member pay the new price or the old price when their next payment is due? For ease of management, I chose to arrange it so that the member is only entitled to pay whatever the current price of the plan is. However should STUDIO2 ever wish to have it that members pay the same price which they were originally quoted, possibly until their subscription renewal date, this could be accomplished by adding a price field into the subscription record which retrieves the price from the Amount field of the selected plan record when the subscription period starts. When a member record is deleted, due to the relational nature of the database, all records corresponding to the MemberID of the record must also be deleted. Due to the nature of the relational database model, when deleting a member, in order to retain the referential integrity of the system, child records containing the MemberID as a foreign key must also be deleted. Because of this, when accounts are performed by STUDIO2 they may be unable to find any reference to a specific subscription/invoice and this could affect their ability to perform tax returns at the end of the financial year due to being

unable to account for why there is a significant amount of extra cash recorded. This problem could however be fixed by adding a separate archival database which has the sole purpose of the storing records that have been deleted from the main database thus meaning that referential integrity is retained and records are not lost.

Developer Diary

02/11/15	Prepared coursework document
03/11/15	Wrote out background information and began focusing on Problem Identification
04/11/15	Continued working on problem identification
06/11/15	Begun focusing on developing fields for normalisation
09/11/15	Continued with normalisation
10/11/15	Continued with normalisation
11/11/15	Continued with normalisation
12/11/15	Continued with normalisation
13/11/15	Continued with normalisation
16/11/15	Researched Gantt charts
17/11/15	Begun building Gantt chart
18/11/15	Finished Gantt chart
19/11/15	Started storyboards
20/11/15	Continued with storyboards
23/11/15	Continued with storyboards
24/11/15	Continued with storyboards
25/11/15	Continued with storyboards
26/11/15	Continued with storyboards
30/11/15	Began doing SQL examples to become familiar with the syntax
01/12/15	Continued using SQL
02/12/15	Continued using SQL
03/12/15	Continued using SQL
04/12/15	Continued using SQL
07/12/15	Continued using SQL
08/12/15	Started writing SQL for system database
09/12/15	Continued writing SQL for system database
10/12/15	Completed SQL for system database. Started developed by beginning the creation of Windows Forms GUI interface.
11/12/15	Continued with GUI
14/12/15	Continued with GUI
15/12/15	I decided that I was not happy with GUI originally planned in the storyboards and decided to make changes. Most notable of these was to dock the navigation ToolStrip to the top of all Forms.
16/12/15	Continued with GUI
17/12/15	With changes made to the GUI I also went back and changed my normalisation as I believed that it would not be able to perform as I intended
18/12/15	Continued focusing on improving normalisation
04/01/16	Continued with GUI

05/01/16	Completed GUI.
06/01/16	Researched how to integrate SQL database with C# application.
08/01/16	SQL database successfully integrated with C# application. Because of this, SQL statements can be performed programmatically within the application
11/01/16	Setup up a 'Test' table in database to test within application how to perform SQL statements. Encountered issues with this. However I later found the issue was that the connection to the database was never opened and thus the SQL query could not be executed
12/01/16	With the test SQL successful, I moved onto enabling the application to create a new member. Again errors were encountered.
13/01/16	Continued to encounter errors with creating a new member record.
14/01/16	Upon further inspection of the string used as the SQL query, I found that one of the fields was missing quotation marks and when added, this rectified the issue and allowed a member to be created.
15/01/16	Focused on updating of members.
18/01/16	Encountered errors when updating members. Continued working on updating of members
19/01/16	Found problem with updating members and have now completed updating members
20/01/16	Began working on adding a way of letting the user enter the name of the SQL server and database, as these details will vary depending on the machine the application is used on and the name of the database. Incorrect details mean that the connection string for connecting to the database will be incorrect.
25/01/16	Continued focusing on adding a way to manually connect to the database through the application.
26/01/16	Successfully completed manual connection strings. Connection strings no longer need to be hardcoded when changed
27/01/16	Added ability to search through member records and display results in the DataGridView by using an SQLDataAdapter to retrieve and matching records and then fill these into a DataSet which is returned back to the Members Form
28/01/16	Search functionality added to Subscriptions and Invoices form by copying the code used for the Members form search and making minor changes
29/01/16	Added KeyDown event to search TextBox in each Form so that a search can be performed when the user hits enter in the TextBox, providing a bit more convenience to the user than clicking the search button.
05/02/16	Added button to Members Form to make DataGridView visible and display all records rather than having to navigate to new instance of the form. Also did the same for the subscription form.
08/02/16	Added button to view all invoices using the same code as yesterday.
09/02/16	Functionality added to invoice filtering buttons in Invoice Form.
11/02/16	Decided to implement a responsive design to enable the application to function properly on a monitor of any size
12/02/16	Continued working on adding a responsive design
15/02/16	Finished with responsive design.
16/02/16	Fixed issue involving the user being able to create a member with an age below 16
17/02/16	Found out about ReportViewers. Did some research into what they can be used for and how to use them
22/02/16	Started adding code to retrieve analytics for Analytics Form

23/02/16	Continued with analytics
24/02/16	Analytics complete. Also added a ReportViewer to Analytics Form to display printable reports
25/02/16	Continued using ReportViewer but had difficulties with it so will go back to it at another stage.
26/02/16	Added a GroupBox to Subscriptions Form to house a DataGridView displaying records from Plan table
27/02/16	Focused on adding functionality to change Amount value of plan records
28/02/16	Could not get regex for validating the new price entered to function properly so I continued with implementing the rest of the necessary code
29/02/16	Used a different regex and this seems to have worked much better than the previous. I then finished writing the code to get the price updating to work
03/03/16	Focused on creating subscriptions. Researched the use of a SqlDataReader to parse in data from Plan table. In order to create a subscription, [Plan].PlanID, [Plan].Months and [Plan].Interval are required. Initially encountered errors attempting to put the data retrieved from the SqlDataReader into a DataTable. I found that the SqlDataReader operates similar to the StreamReader class (which I used in my AS coursework) allowing me to solve the problem by simply using a while loop and the .Read() function of the SqlDataReader to parse in data.
04/03/16	Added some minor validation to Subscriptions form. Developed code to update/cancel subscriptions. In canceling a subscription I have made it that if a pending invoice is present it is deleted however I am unsure whether it is best to completely delete the record instead of simply using a 'Canceled' value in the [Invoice].IStatus property. Ended up simply changing the IStatus.
07/03/16	Encountering error when loading program on a different machine for the first time. It may be something to do with the table adapters not updating after changing App.config. They may need to be refreshed after executing the connection to the server.
08/03/16	Added code to delete a member however I encountered SQL errors. This was due to the referential integrity of the database and so I was required to delete child records such as subscriptions linked to the members MemberID then delete the member
09/03/16	Added a way to update invoices so that the IStatus can be changed i.e. when the invoice is paid
10/03/16	Continued with updating invoices
11/03/16	Started adding print functionality to each Form except Analytics. The method I used proved to be ineffective however as it only displayed the visible portion of the DataGridView present. Because of this I had to research a different way of doing it.
14/04/16	Found a different way of printing DataGridViews and implemented this instead. This proved to be much more reliable than the old method.
15/04/16	Went back to trying to get ReportViewer properly working. However I still could not find a way of getting it to display the values I wanted.
21/03/16	Continued with ReportViewer but still no luck. I have decided to simply keep it simple so as to not waste any more time on it.
22/03/16	Working on adding information to system approach section of documentation
02/04/16	Added code for handling the automatic issuing of new invoices and subscriptions
03/04/16	Coded automatic handling of invoices and subscriptions which must be suspended. Added responsive form design by having controls resize/relocate whenever the form is resized. Implemented this on all forms. Encountered issue with connection string not updating unless application is restarted after updating it in Start form again. Application

	may require a restart when App.config is altered but will look into this later.
05/04/16	Continued with system approach documentation
06/04/16	Minor additions made to documentation
08/04/16	Started test plan
09/04/16	Continued with test plan
10/04/16	Continued with test plan
11/04/16	Continued with test plan
12/04/16	Started test evidence
13/04/16	Continued with test evidence
14/04/16	Implemented printing for Members, Invoices and Subscriptions forms
16/04/16	Continued with test evidence
18/04/16	Continued with test evidence
19/04/16	Continued with test evidence
21/04/16	Started evaluation
22/04/16	Continued with evaluation
23/04/16	Continued with evaluation
25/04/16	Project for the most part completed. Looked over documentation and made various different minor amendments and fixed grammatical issues
26/04/16	Reviewed documentation for any missing areas