School of Computing Electrical & Applied Technology

ISCG6421 – GUI Programming
Semester 1, 2022

Assignment 1

Submission Time:	**5 p.m.**
Checkpoint 1:	**Thursday 31st March 2022**
Checkpoint 2:	**Thursday 14th April 2022**
Final Submission:	**Wednesday 4th April 2022**
Course Weighting:	**40%**
Marks:	**100**

Contents

Assignment objectives:

There are two objectives for this assignment:

1. To demonstrate the use of the C# language for the creation of an effective database front end by developing forms which:
    a. Visually present database information.
    b. Demonstrate appropriate user and programmatic navigation.
    c. Allow the user to enter new data and change existing data.
2. To demonstrate the testing, debugging and documentation of a C# application by:
    a. Using the available debugging tools to debug a program.
    b. Developing test cases appropriate to a GUI approach.

Introduction

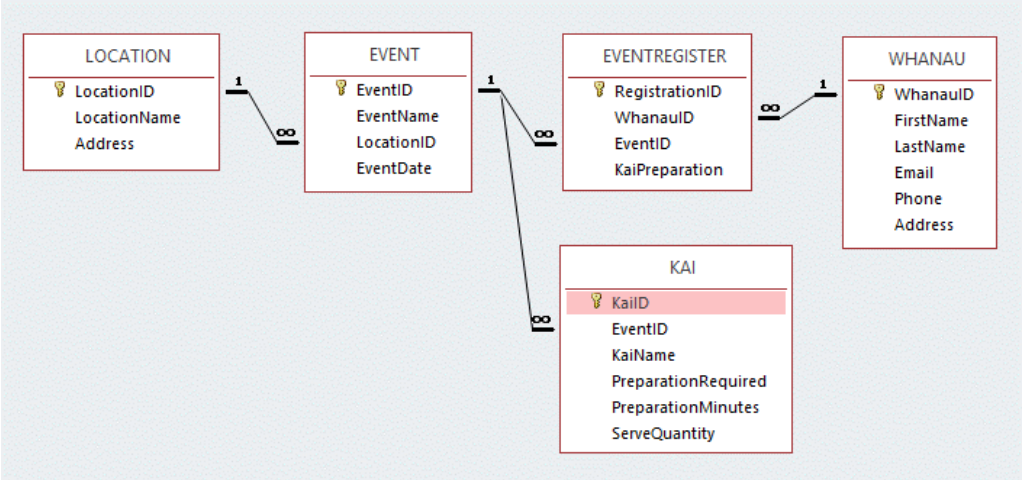
This assignment is based on a case study of the organisation of events at Unitec, involving the preparation and sharing of kai. As a result of the case study a request was made for an application that can be used by event organisers to simplify the planning and management of events.

The database have been normalized and designed for you in Microsoft Access. You, the programmer, must create a WinForms application using C# and .NET Framework, which will allow the users to maintain the records stored in the database and produce the reports that they need.

Database

You are provided with an MS Access database for this assignment named **kai-oordinate.mdb**. Please leave the structure of this database as it is and do not add any tables or queries. The database has the following tables (note that the primary keys are underlined, and foreign keys are indicated by a '*' symbol):

KAI (KaiID, EventID*, KaiName, PreparationRequired, PreparationMinutes, ServeQuantity)
EVENT (EventID, EventName, LocationID*, EventDate)
LOCATION (LocationID, LocationName, Address)
WHANAU (WhanauID, FirstName, LastName, Email, Phone, Address)
EVENTREGISTER (RegistrationID, WhanauID*, EventID*, KaiPreparation)



Guide to the Assignment

- You are to write the code in C# using **Visual Studio .NET Community 2019 or 2022**
- You are to use the **OleDb** objects to interact with the **MS Access** database tables.
- Make sure that your C# project database connection points to **C:\Temp** and make sure the **MS Access** database file is placed in **C:\Temp**
- You are to use the shared dataset model (see the **Shared Data, Wallace Events** and **Glendene Cat Care** projects, as discussed in class, and notes on databases in C#).
- You are to use programming logic (master-detail relationships in conjunction with 'for-each' loops) to create the reports – please do **not** use Crystal Reports.
- Your interface should conform to the standards discussed in class and – please note the interface in **GlendeneCatCare** has limitations and should be used as a starting point but not as an exemplar.
- Use the metadata provided in **MS Access** database tables as the data dictionary from which you can make decisions on what is valid or invalid data.
- Your test cases must include all of the tests that are listed as required in the assignment document and your tests must be repeatable and independent. You may use the data in the MS Access tables as the basis for your loaded test data.
    - Select your input test data to make your tests as independent as possible. For example: if you are testing that a program deletes an event record then use, as test data, an event that exists in the loaded test data NOT an event that was input in an earlier test.
    - Remember to verify the expected results of the tests with other tests if possible. For example, if you have a test that adds a new record, then use the 'view' facility in the program to check that the record has been added.
    - The numbering of your tests MUST correspond with the numbers in this document. If you create your interface so that a test is redundant (e.g. you use radio buttons for selection) you must still include the test in your test cases along with an explanation as to why it is redundant. Check all instances of this circumstance with the lecturer.
- You also required to add two extra features of your choice to the user interface or reports which should further improve it use.
- You must submit a work-in-progress version of your assignment for each of the checkpoint on Moodle. Minimum requirements implemented are listed below in the Checkpoints section.
- You must submit any questions and/or queries regarding this assignment to the lecturer before the Question Submission Deadline in the course schedule.

Test Cases

Please using the following format for your test cases and leave the actual outcomes column blank:

Requirement to test	Test Data Input	Expected Outcomes	Actual Outcomes

Checkpoints

There is an expectation that you will have made reasonable progress towards completing your assignment at each checkpoint submission. Your progress shown at each may be taken into consideration during the marking of your assignment.

Expectations:

Checkpoint 1:

Approximately half of the assignment should be completed. This may be represented as:

- All user interfaces created and event methods registered. No functional code written.
  OR
- Some user interfaces created and related event methods registered. Some functional code written.
  OR
- At least 1 user interface created and related event methods registered and functional code written.

Checkpoint 2:

Almost all of the assignment should be completed. This may be represented as:

- All user interfaces created and event methods registered. Most functional code written. Most testing completed.
  OR
- Application completed. No testing completed.
  OR
- Assignment completed except for printing, an area of difficulty or extra features.

Design Guide

Fonts		FontAwesome Button icons	
Field	Font type	Button	Icon char
Button text	Tahoma 12pt Bold	Kai	DrumstickBite
Label	MS Sans Serif, 14pt	Events	CalendarAlt
Text box	MS Sans Serif, 14pt	Whānau	Users
List box	MS Sans Serif, 14pt	Locations	MapMarkedAlt
Combo box	MS Sans Serif, 14pt	Registration	ClipboardList
Numeric UpDown	MS Sans Serif, 14pt	Report	Print
DateTime picker	MS Sans Serif, 14pt	Exit	Times
		Up	CaretSquareUp
		Down	CaretSquareDown
		Add	Plus
		Update	Edit
		Delete	TrashAlt
		Return	SignOutAlt
		Save	Save
		Cancel	Times
		Generate Report	Print

Main form icon: **icon.ico**
Main form banner logo: **cover_[size].png**
Form background colour: **(6, 73, 41)**
Button foreground colour (enabled): **ControlText/Black**
Button background colour (disabled): **ControlDark**

All buttons used in this application are IconButton from the FontAwesome.Sharp Nuget package, rather than the default button from WindowsForms. You will need to add the package to your project and restart for the IconButton option to show in your ToolBox if you are not getting it.

Please note: All GUI design images in this document are intended for use as a guide and not guaranteed to be accurate to the design requirements. Where the design requirements differ from image guides, follow the design requirements.

Application Requirements

Write a Windows application in C# that allows the user to do the tasks specified below and to provide reasonably comprehensive testing documentation that proves that the application performs all of the tasks. Please note that these requirements are non-negotiable.

Task 1: Main Menu

- This task allows the user to access all the forms and the reports in the system and exit the application.
- When run, this task will create a data module object (containing the OleDb objects used with the MS Access database tables) the address of which will be passed on to all the other forms via a parameter.

Required tests:

1.1. Main menu is displayed when the program is run.
1.2. Kai management form is displayed when the Kai button is clicked.
1.3. Event management form is displayed when the Event button is clicked.
1.4. Whanau management form is displayed when the Whanau button is clicked.
1.5. Location management form is displayed when the Location button is clicked.
1.6. Registration management form is displayed when the Registration button is clicked.
1.7. Report form is displayed when the Report button is clicked.
1.8. The application exits when the Exit button is clicked.

Main Menu Design

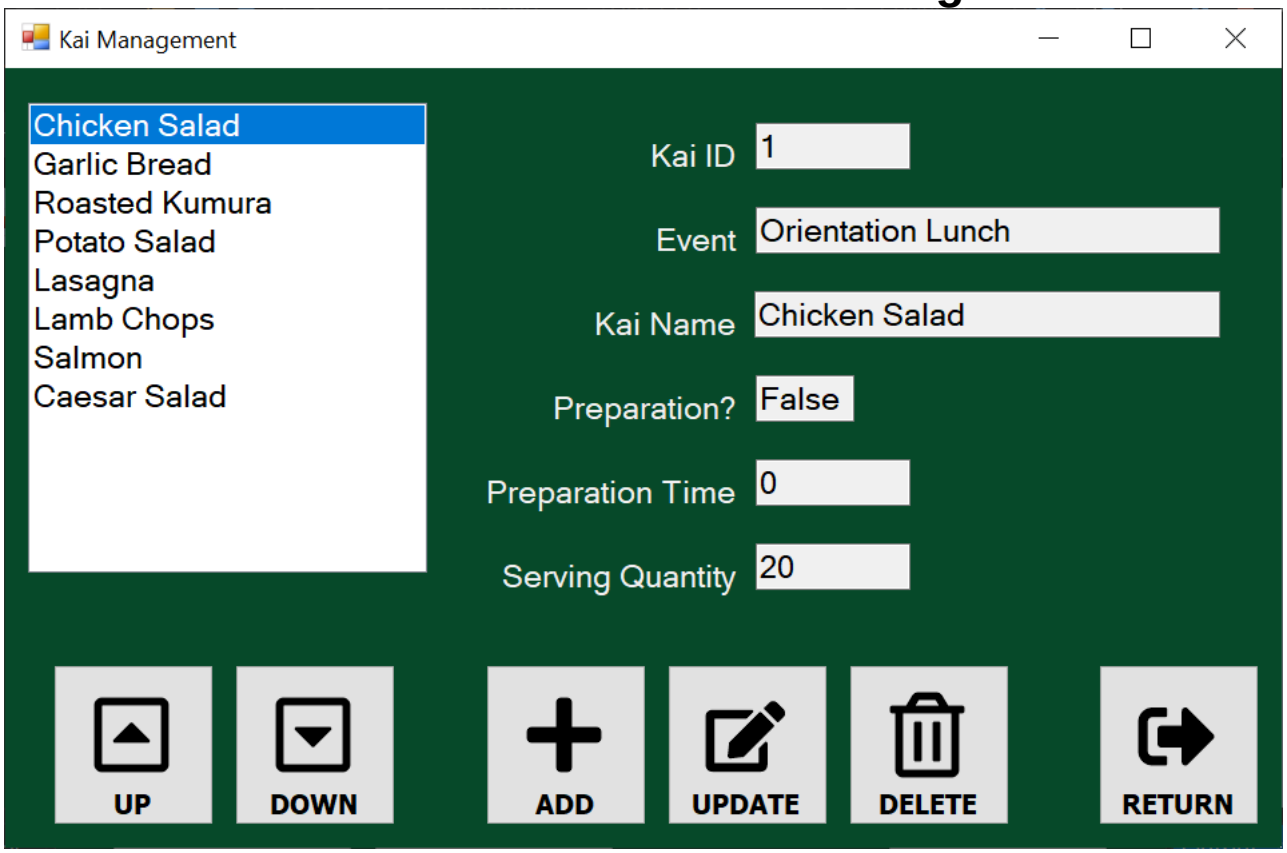
Task 2: Kai Maintenance

1. Display the kai names in a list.
2. Allow the user to navigate between kai records in the list.
3. Display a selected item's details (kai id, event name, kai name, preparation required, preparation minutes, serving quantity) in a read-only format.
4. If the user clicks on the **Add** button, then
    a. The **Up**, **Down**, **Return**, **Update** and **Delete** buttons are disabled, and the list should disappear.
    b. A panel should appear (showing **Save** and **Cancel** buttons) allowing the user to enter values for the new kai's name, event (combo box), preparation requirement (checkbox), preparation time (if required, numeric UpDown picker), serving quantity (numeric UpDown).
    c. If the user enters valid data for all fields and clicks on the **Save** button, then a new kai record is saved in the database and the message "Kai added successfully" is displayed.
    d. If the user clicks on the **Cancel** button then panel should disappear, the **Previous**, **Next**, **Return**, **Update** and **Delete** buttons are enabled, and the list should reappear.
5. If the user clicks on the **Update** button, then
    a. The **Previous**, **Next**, **Return**, **Add** and **Delete** buttons are disabled and the list should disappear.
    b. A panel should appear (showing **Save Changes** and **Cancel** buttons) allowing the user to change the values for the kai's name, event name, preparation requirement, preparation minutes, or serving quantity.
    c. If the user makes valid changes to any of the allowable fields and clicks on the **Save Changes** button, then the kai record is updated in the database and the message "Kai updated successfully" is displayed.
    d. If the user clicks on the **Cancel** button then the panel should disappear, the **Previous**, **Next**, **Return**, **Add Arena** and **Delete Arena** buttons should be enabled and the list should reappear.
6. If the user clicks on the **Delete** button and the selected record has no event, then the user is asked to confirm the deletion – if the user confirms the deletion then the kai is deleted from the database and the message "Kai deleted successfully" is displayed.
7. If the user clicks on the **Delete** button and the selected kai has an event, then the error message "You may only delete kai that have no event relation" is displayed.
8. If the user clicks on the **Return** button, then focus is returned to the Main Menu.

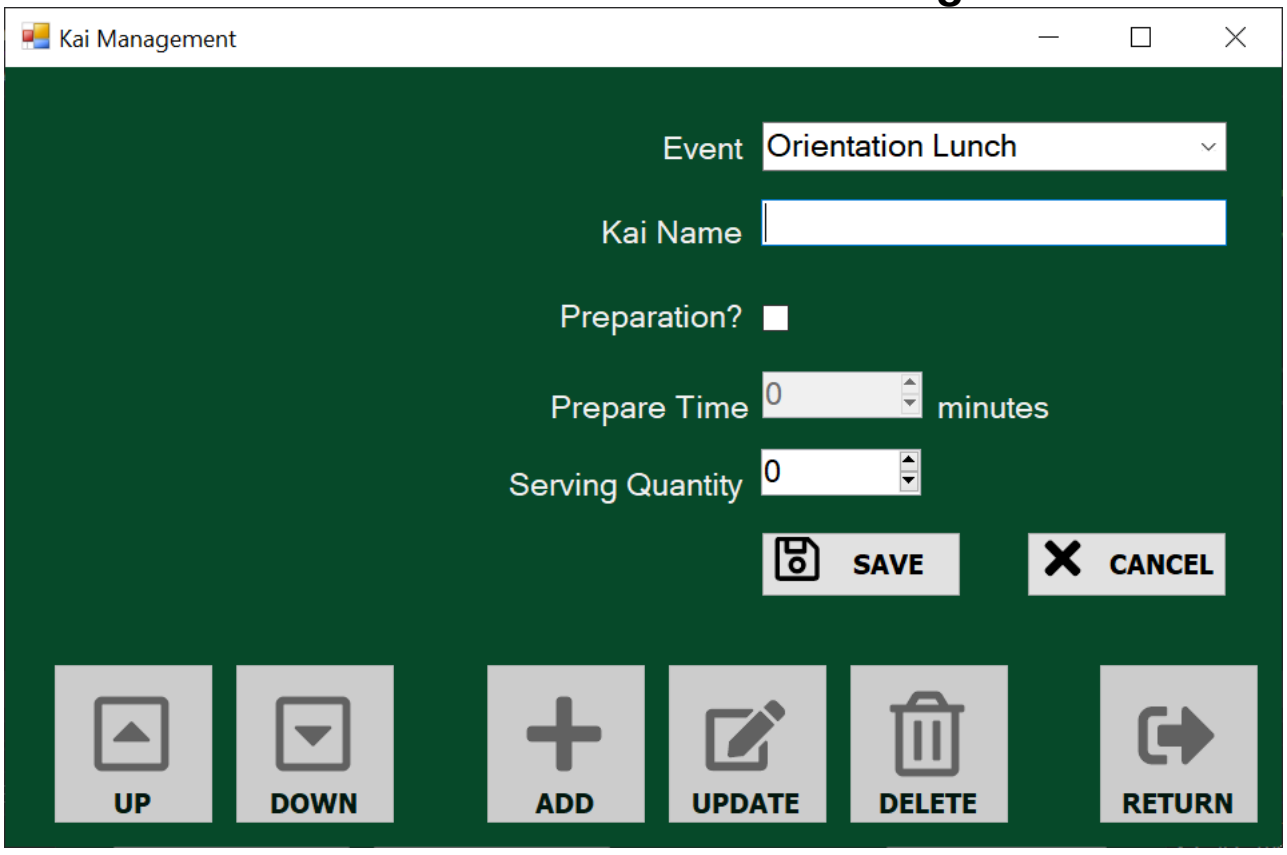
Required tests:

2.1. Add a new kai record with valid data in all fields.
2.2. Update an existing kai's name with valid data.
2.3. Delete an existing kai that has no event.
2.4. Attempt to add a new kai with an invalid quantity but with valid data in all other fields.
2.5. Attempt to update an existing record with a blank name.
2.6. Attempt to delete an existing arena that has events.

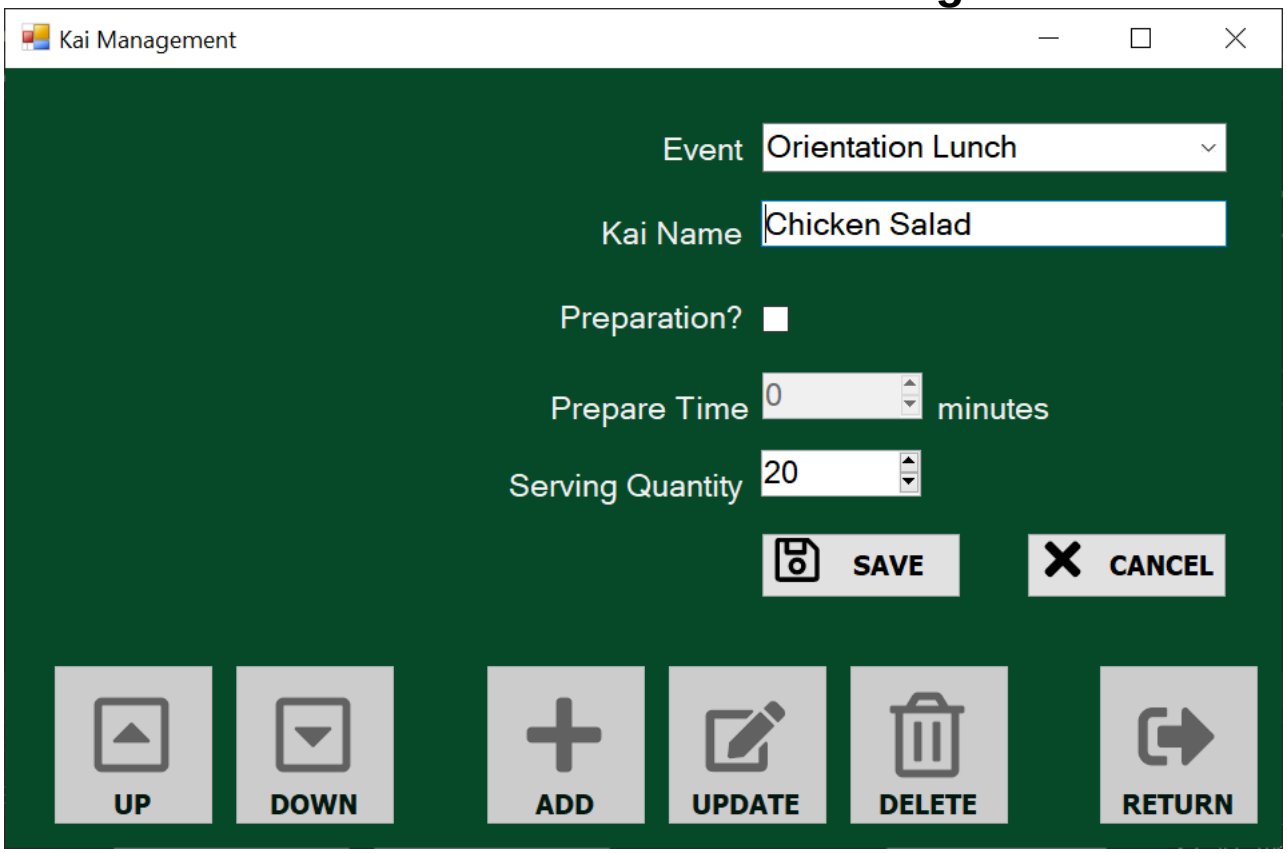
Kai Maintenance: Screen Design 1



Kai Maintenance: Screen Design 2



Kai Maintenance: Screen Design 3



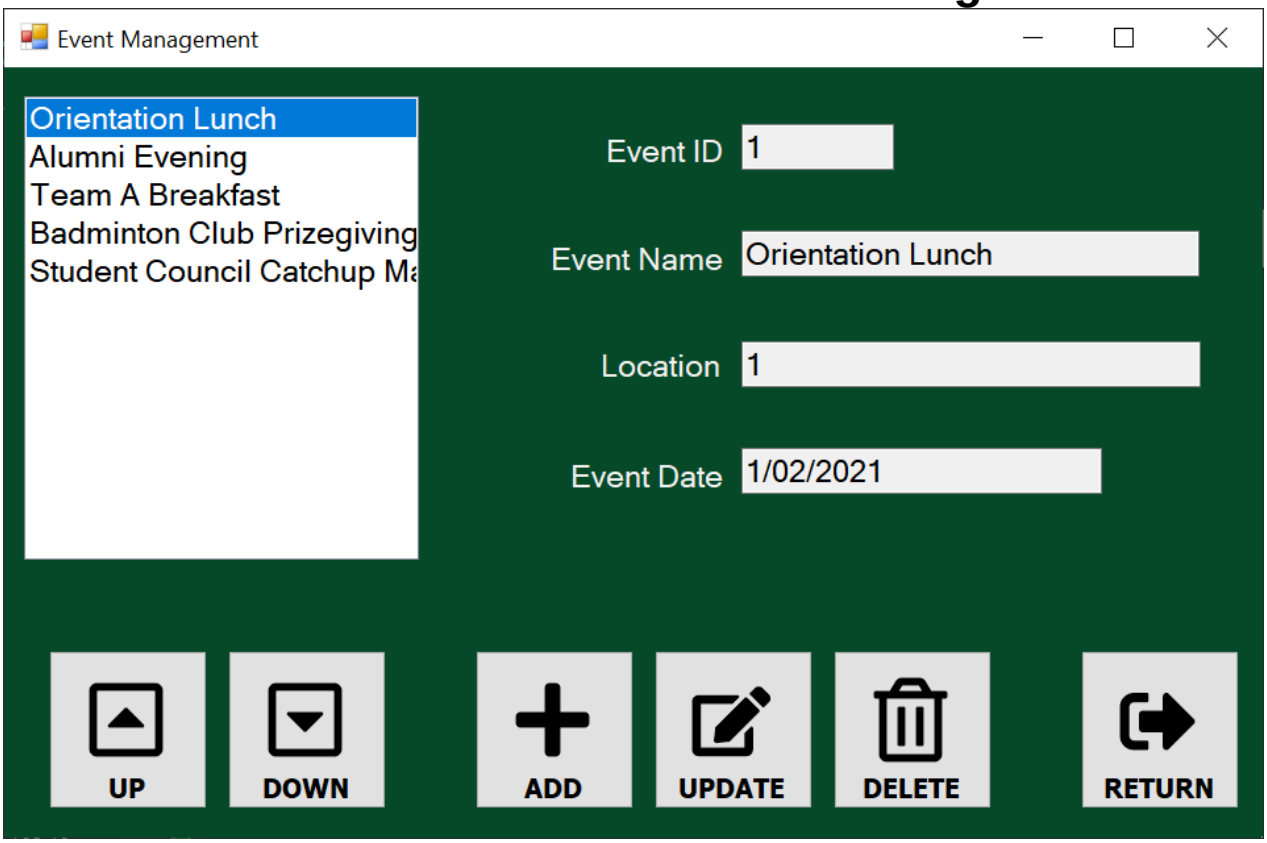
Task 3: Event Maintenance

1. Display the Event names in a list.
2. Allows the user to navigate between events in the list.
3. Display a selected event's details (event id, event name, location id, event date) in a read-only format.
4. If the user clicks on the **Add** button, then
    a. The **Previous**, **Next**, **Return**, **Update, Delete** buttons are disabled and the list should disappear.
    b. A panel should appear (showing **Save** and **Cancel** buttons) allowing the user to enter values for the new event's name, location (using combo box), date (using a dateTime picker).
    c. If the user enters valid data for all fields and clicks on the **Save** button, then a new record is saved in the database and the message "Event added successfully" is displayed.
    d. If the user clicks on the **Cancel** button then the panel should disappear, the **Previous**, **Next**, **Return**, **Update**, **Delete** buttons are enabled and the list should reappear.
5. If the user clicks on the **Update** button, then
    a. The **Previous**, **Next**, **Return**, **Add**, **Delete** buttons are disabled and the list should disappear.
    b. A panel should appear (showing **Save** and **Cancel** buttons) allowing the user to change the value for the event's name, location and date (using a dateTime picker).
    c. If the user makes valid changes to any of the allowable fields and clicks on the **Save** button, then the record is updated in the database and the message "Event updated successfully" is displayed.
    d. If the user clicks on the **Cancel** button then panel should disappear, the **Previous**, **Next**, **Return**, **Add**, **Delete** buttons are enabled and the list should reappear.
6. If the user clicks on the **Delete** button and the selected event has no kai, then the user is asked to confirm the deletion – if the user confirms the deletion then the record is deleted from the database and the message "Event deleted successfully" is displayed.
7. If the user clicks on the **Delete** button and the selected challenge has kai, then the error message "You may only delete an event that has no kai" is displayed.
8. If the user clicks on the **Return** button, then focus is returned to the Main Menu.

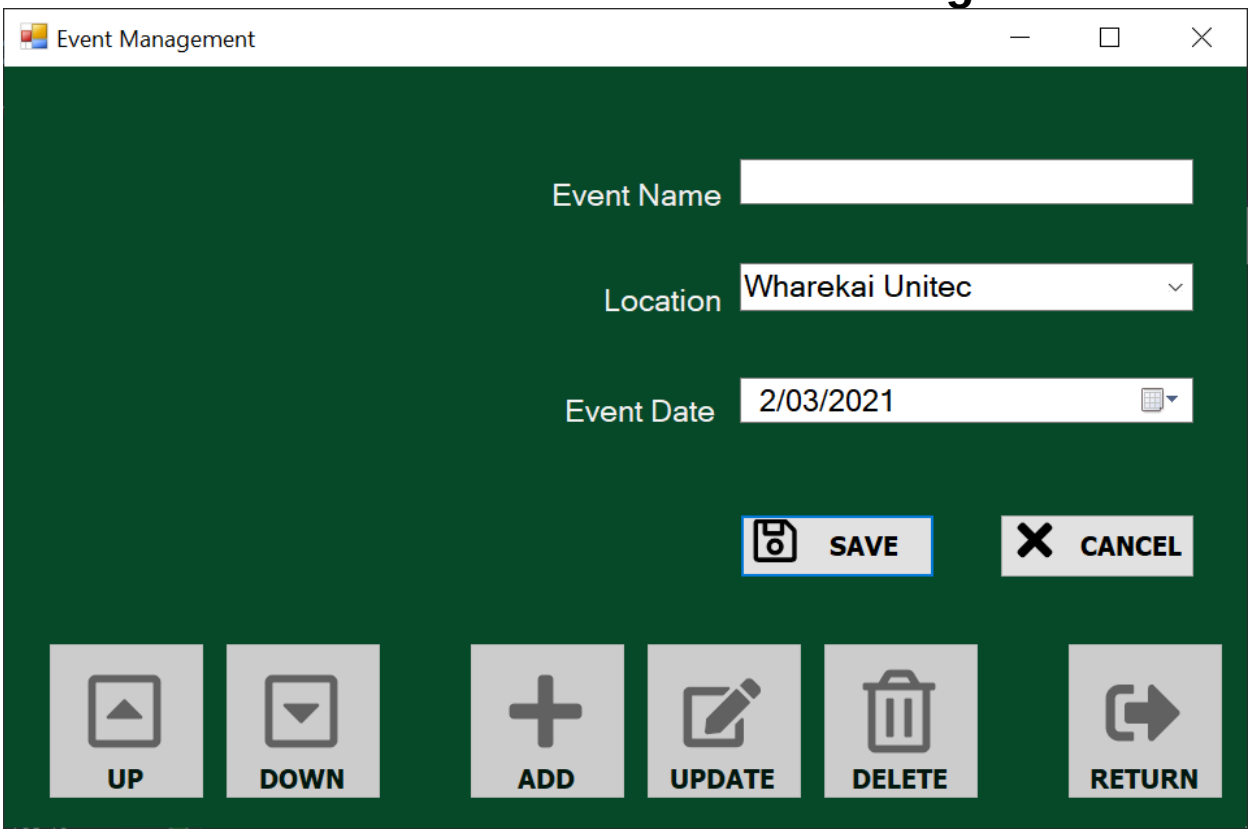
Required tests:

3.1. Add a new event with valid data in all fields.
3.2. Update an existing event date with valid data.
3.3. Update an existing event location with valid data.
3.4. Delete an existing event that has no kai.
3.5. Attempt to delete an existing event that has at least one kai.

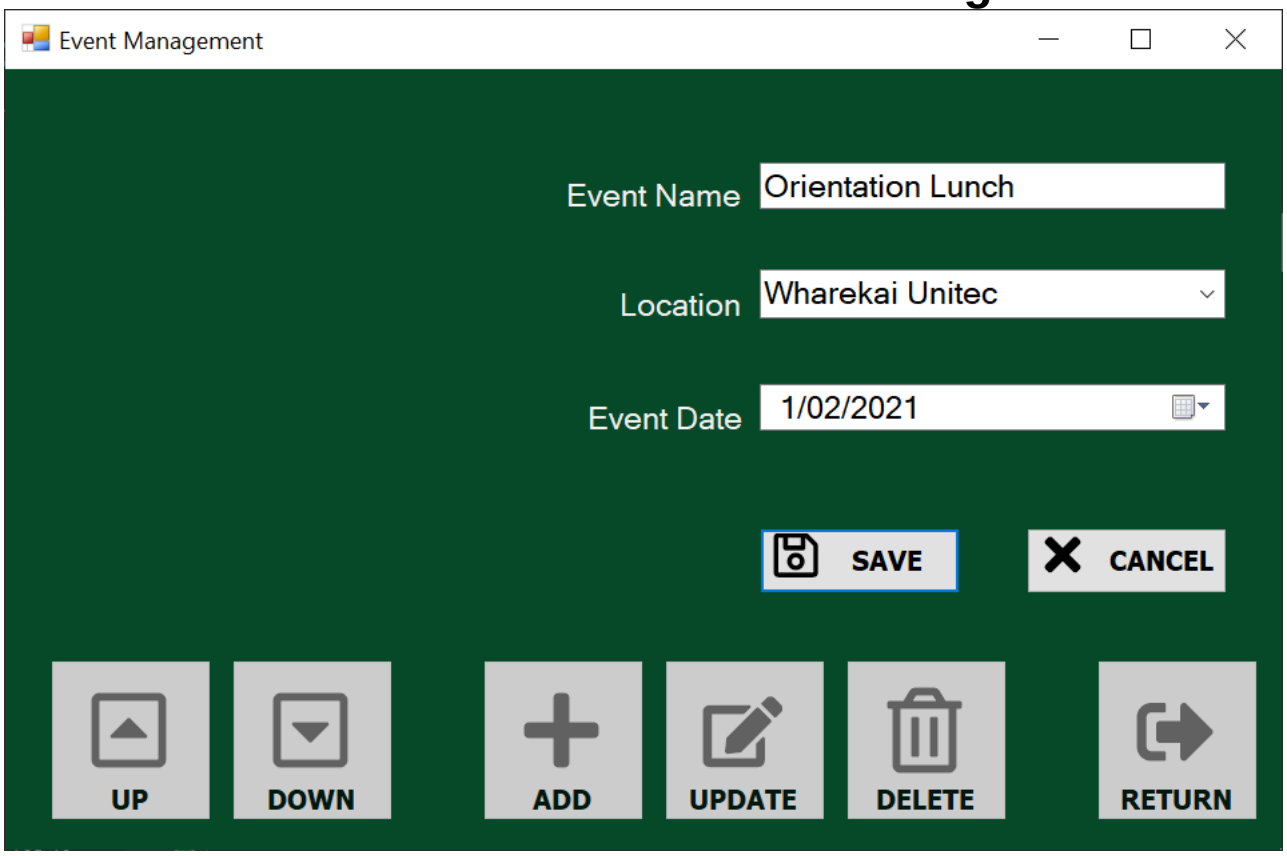
Event Maintenance: Screen Design 1



Event Maintenance: Screen Design 2



Event Maintenance: Screen Design 3



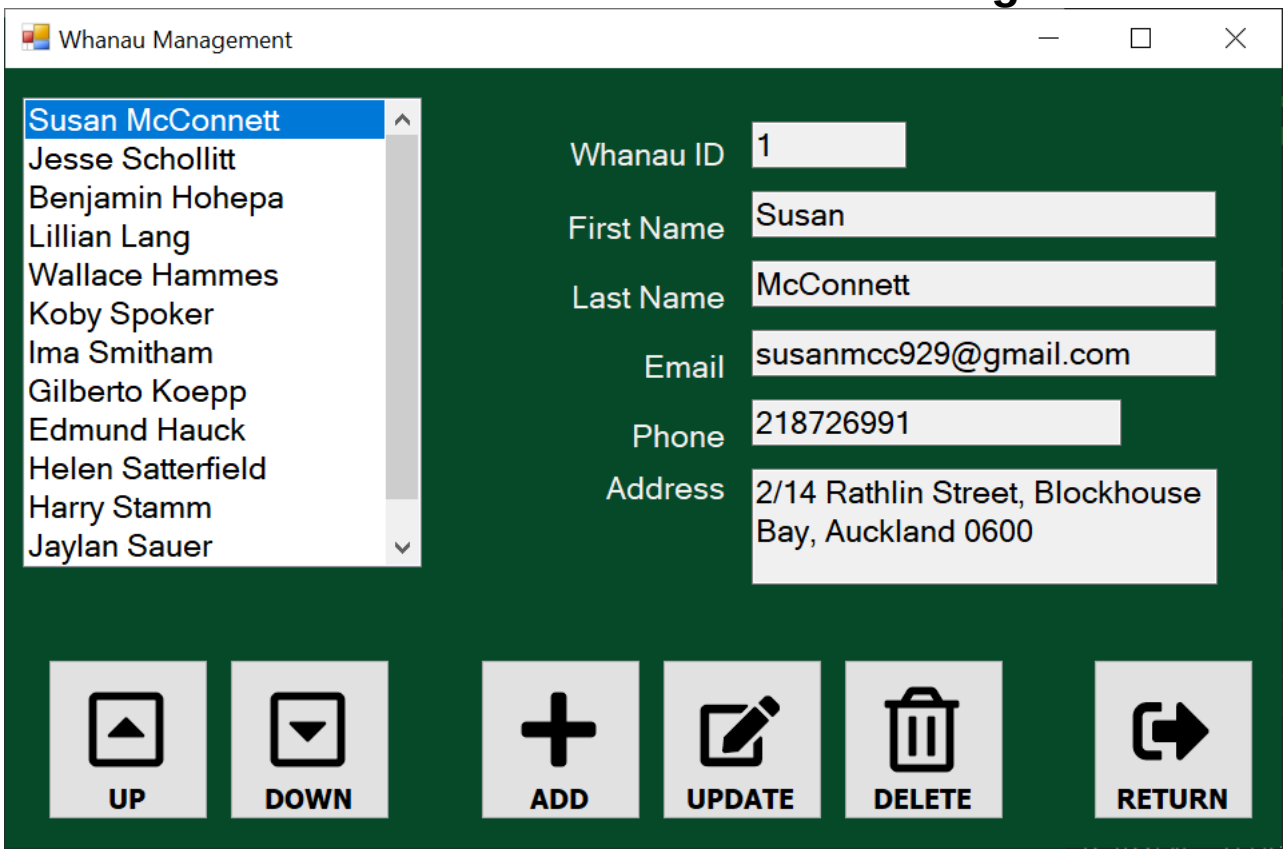
Task 4: Whānau Maintenance

1. Display the whānau names (first and last) in a list.
2. Allow the user to navigate between records in the list.
3. Display a selected record's details (whānau id, first name, last name, email, phone, and address) in a read-only format.
4. If the user clicks on the **Add** button, then
    a. The **Previous**, **Next**, **Return**, **Update** and **Delete** buttons are disabled and the list should disappear.
    b. A panel should appear (showing **Save** and **Cancel** buttons) allowing the user to enter values for the new record's first name, last name, email, phone number, and address.
    c. If the user enters valid data for all fields and clicks on the **Save** button, then a new record is saved in the database and the message "Whānau added successfully" is displayed.
    d. If the user clicks on the **Cancel** button then panel should disappear, the **Previous**, **Next**, **Return**, **Update** and **Delete** buttons are enabled and the list should reappear.
5. If the user clicks on the **Update** button, then
    a. The **Previous**, **Next**, **Return**, **Add** and **Delete** buttons are disabled and the list should disappear.
    b. A panel should appear (showing **Save** and **Cancel** buttons) allowing the user to change the values for the record's first name, last name, email, phone number, and address.
    c. If the user makes valid changes to any of the allowable fields and clicks on the **Save** button, then the record is updated in the database and the message "Whānau updated successfully" is displayed.
    d. If the user clicks on the **Cancel** button then the panel should disappear, the **Previous**, **Next**, **Return**, **Add** and **Delete** buttons should be enabled and the list should reappear.
6. If the user clicks on the **Delete** button and the selected record has no event registrations, then the user is asked to confirm the deletion – if the user confirms the deletion then the record is deleted from the database and the message "Record deleted successfully" is displayed.
7. If the user clicks on the **Delete** button and the selected record has registrations, then the error message "You may only delete records that have no registrations" is displayed.
8. If the user clicks on the **Return** button, then focus is returned to the Main Menu.

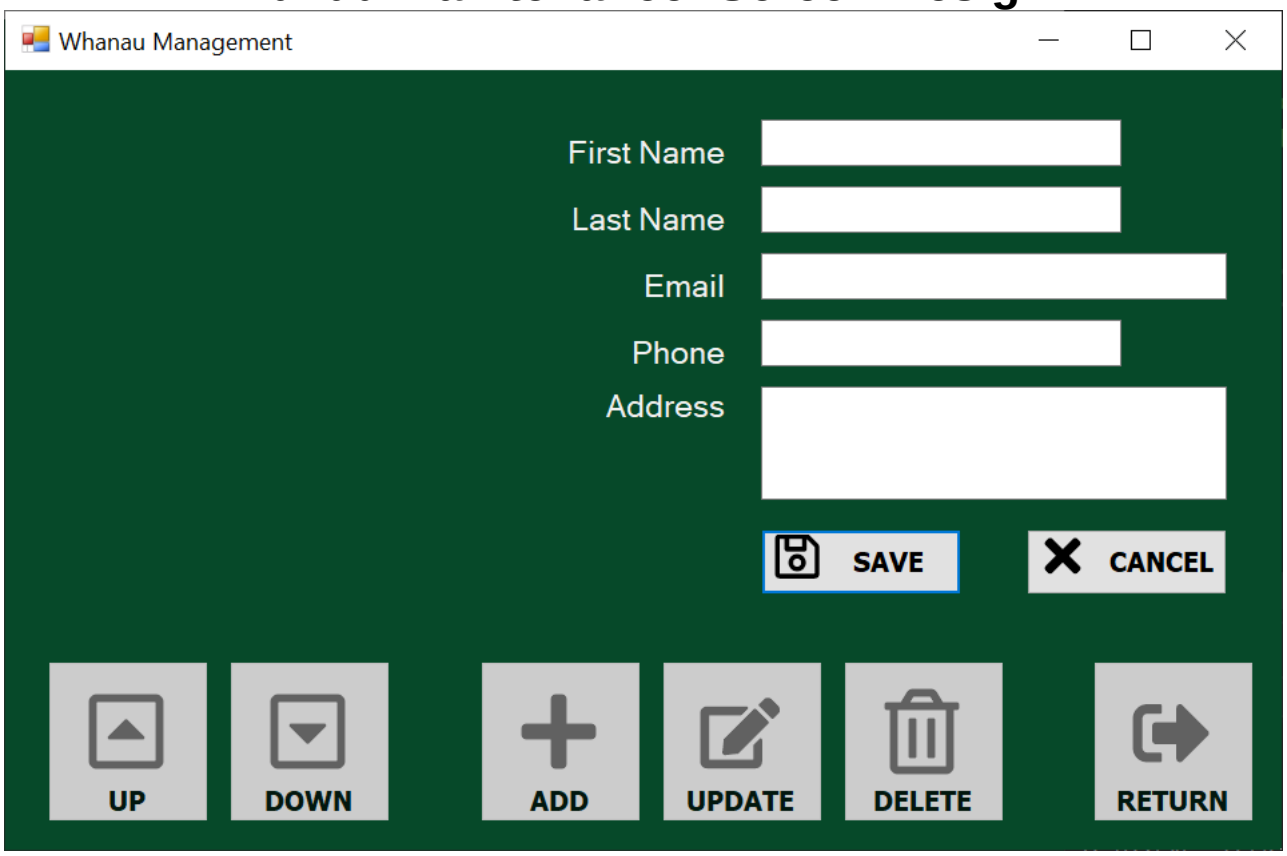
Required tests:

4.1. Add a new record with valid data in all fields.
4.2. Update an existing record's name with valid data.
4.3. Delete an existing record that has no entries.
4.4. Attempt to add a new record with an invalid first name but with valid data in all other fields.
4.5. Attempt to update an existing record with an invalid first name.
4.6. Attempt to delete an existing record that has registrations.

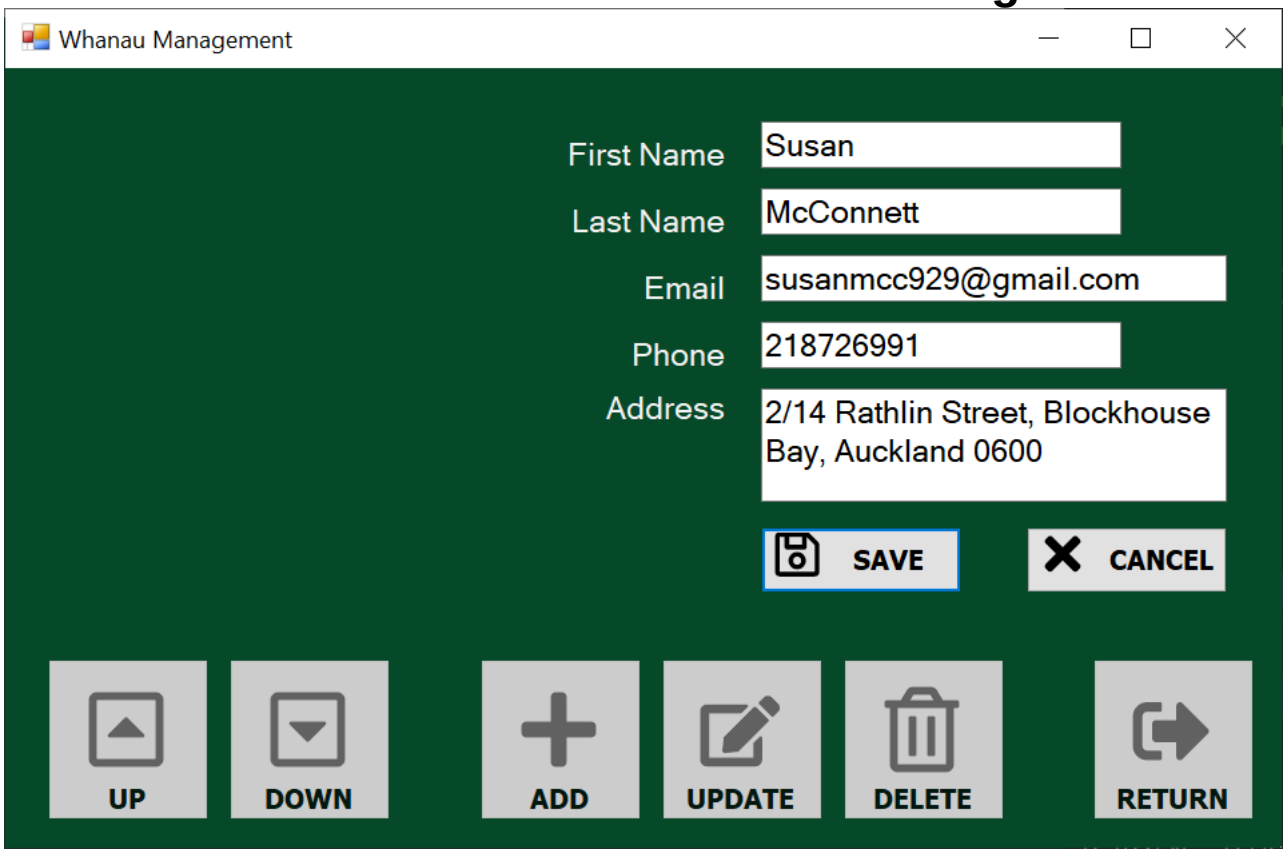
Whānau Maintenance: Screen Design 1



Whānau Maintenance: Screen Design 2



Whānau Maintenance: Screen Design 3



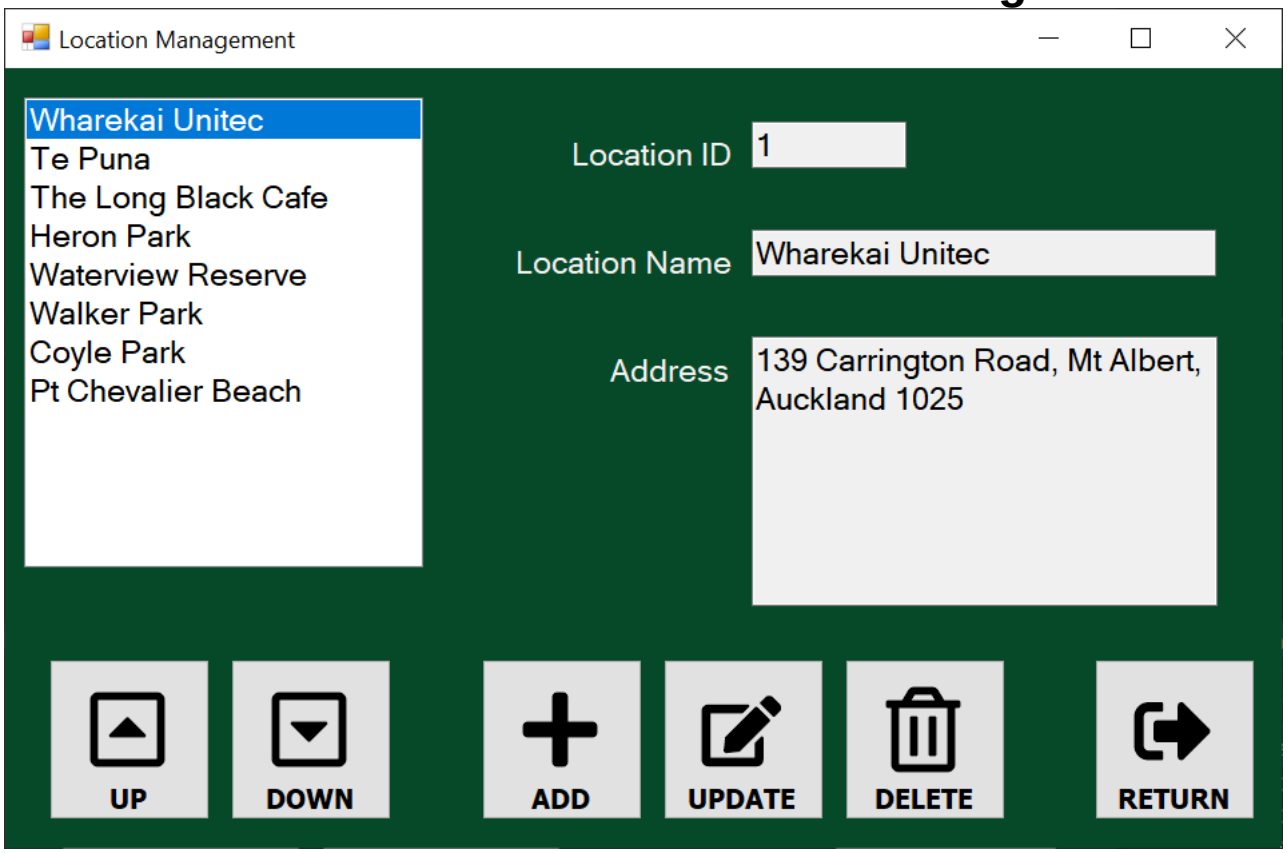
Task 5: Location Maintenance

1. Display the Location names in a list.
2. Allows the user to navigate between locations in the list.
3. Display a selected location's details (Location id, location name, address) in a read-only format.
4. If the user clicks on the **Add** button, then
    a. The **Previous**, **Next**, **Return**, **Update** and **Delete** buttons are disabled and the list should disappear.
    b. A panel should appear (showing **Save** and **Cancel** buttons) allowing the user to enter values for the new location name and address.
    c. If the user enters valid data for all fields and clicks on the **Save** button, then a new event record is saved in the database and the message "Location added successfully" is displayed.
    d. If the user clicks on the **Cancel** button then panel should disappear, the **Previous**, **Next**, **Return**, **Update** and **Delete** buttons are enabled and the list should reappear.
5. If the user clicks on the **Update** button, then
    a. The **Previous**, **Next**, **Return**, **Add** and **Delete** buttons are disabled and the list should disappear.
    b. A panel should appear (showing **Save** and **Cancel** buttons) allowing the user to change the values for the location name, and address.
    c. If the user makes valid changes to any of the allowable fields (location name, address) and clicks on the **Save** button, then the record is updated in the database and the message "Location updated successfully" is displayed.
    d. If the user clicks on the **Cancel** button then the panel should disappear, the **Previous**, **Next**, **Return**, **Add** and **Delete** buttons should be enabled and the list should reappear.
6. If the user clicks on the **Delete** button and the selected location has no events registered to it, then the user is asked to confirm the deletion – if the user confirms the deletion then the record is deleted from the database and the message "Location deleted successfully" is displayed.
7. If the user clicks on the **Delete** button and the selected location has events, then the error message "You may only delete locations that have no events" is displayed.
8. If the user clicks on the **Return** button, then focus is returned to the Main Menu.

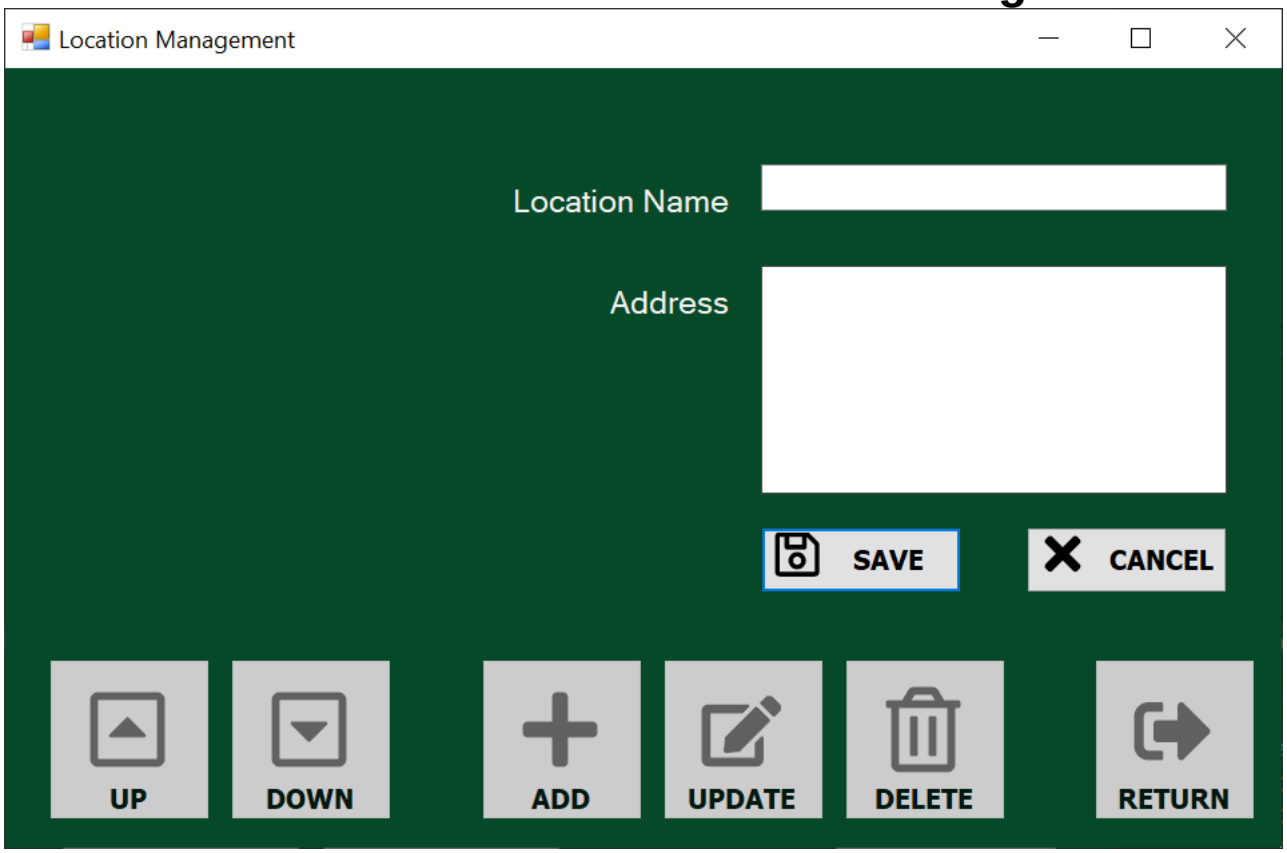
Required tests:

5.1. Add a new location with valid data in all fields.
5.2. Update an existing location's name with valid data.
5.3. Update an existing location's address with valid data.
5.4. Delete an existing location that has no events.
5.5. Attempt to add a new location with an invalid name but with valid data in all other fields.
5.6. Attempt to update an existing location with an invalid name.
5.7. Attempt to delete an existing location that has events.

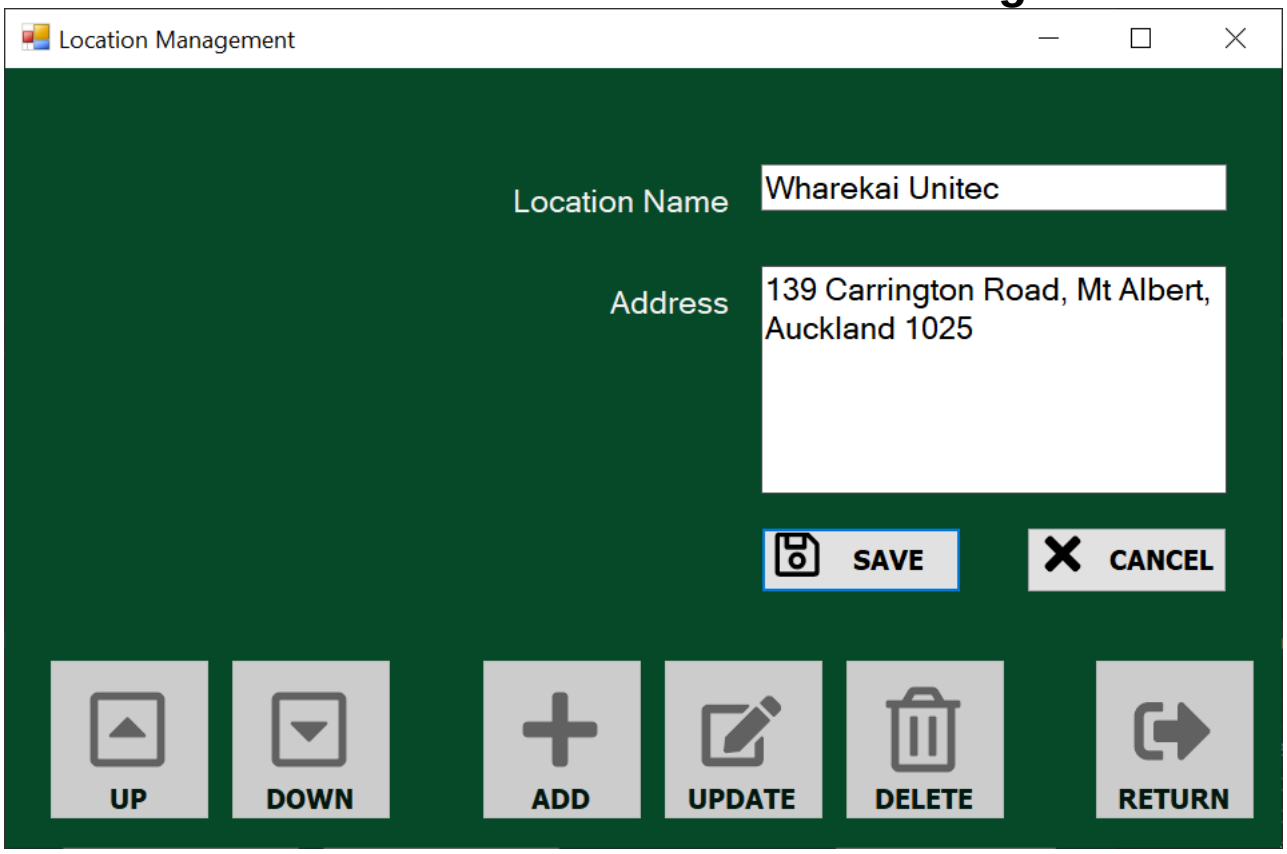
Location Maintenance: Screen Design 1



Location Maintenance: Screen Design 2



Location Maintenance: Screen Design 3



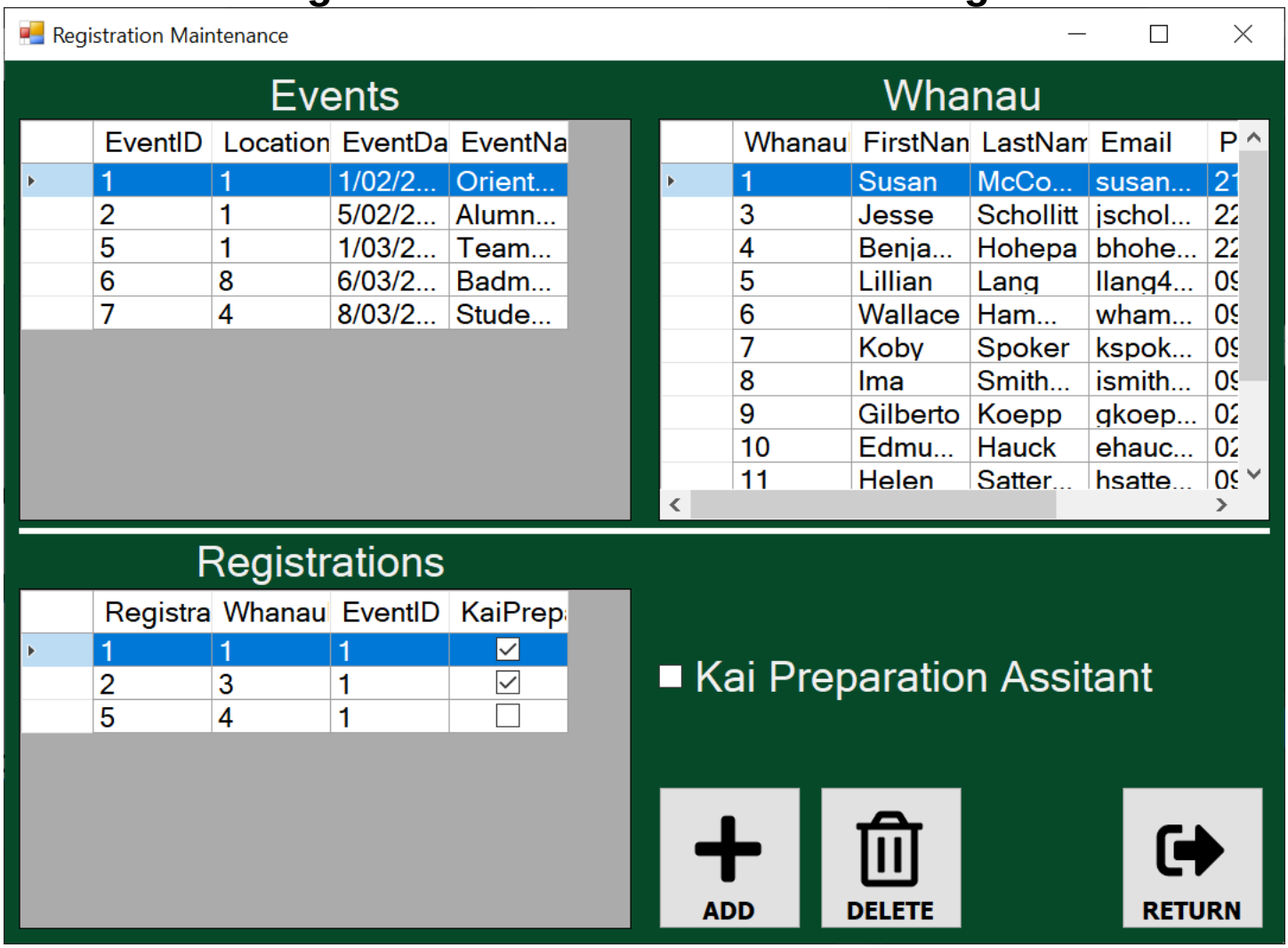
Task 6: Register Whānau to Events

1. Display the whānau's details (whanau id, first name, last name, email, phone, and address) in a data grid view.
2. Display the event's details (event id, event name, location id, date) in a data grid view.
3. Display the registration details (registration id, whanau id, event id, kaipreparation) of all of those whanau linked to the selected event in a data grid view.
4. If the user selects a whanau record and an event and clicks on the **Add Entry** button and the whanau is not already registered to the event, then the entry record is saved in the database and the message "Entry added successfully" is displayed.
5. If the user selects a whanau record and an event and clicks on the **Add Entry** button and the whanau is already registered, then the message "Whanau can only be registered to an event once." is displayed.
6. If the user clicks on the **Remove Entry** button, then the user is asked to confirm the removal – if the user confirms the removal then the entry is deleted from the database and the message "Entry removed successfully" is displayed.
7. If the user clicks on the **Return** button, then focus is returned to the Main Menu.

Required tests:

6.1. Add a whanau to an event
6.2. Add a whanau to an event with kai preparation ticked
6.3. Remove a whanau from an event
6.4. Attempt to add a whanau to an event they are already registered to

Register Maintenance: Screen Design



Task 7: Registration Report

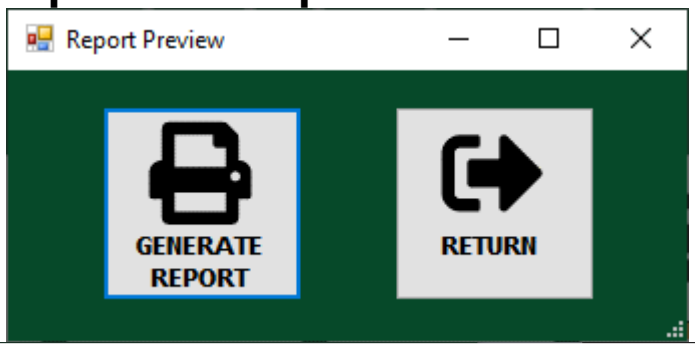
- This must allow the user to print (on a print form) a report that shows, for each event, the event id, event name, date, location, and address, along with the details of the whanau registered to the event (first name, last name, phone number, kaipreparation)
- One Event per page. Please note that Crystal Report is NOT to be used for this task and the report is to be produced using Master-Detail relationships in conjunction with 'for-each' loops. Please find below the logic that produces a display version this report – this logic needs to be adapted for producing the printed report:

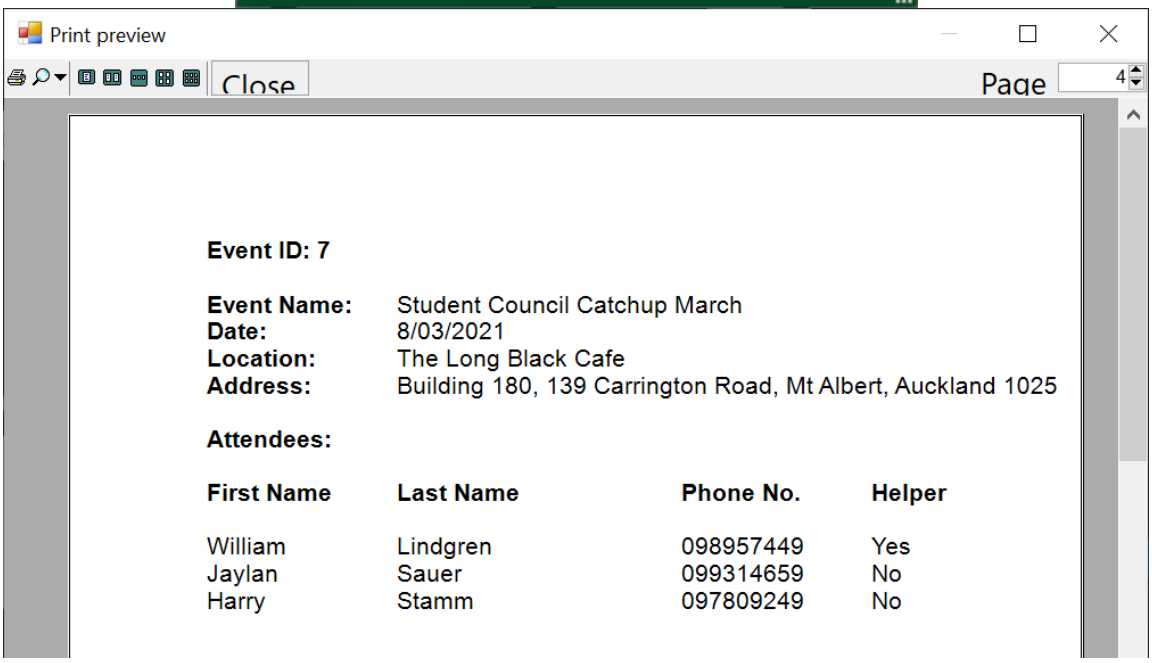
```
LOOP FOR EACH Event in the EVENT Table
        Get the registrations for the Event
        If Event has registrations, THEN
                OUTPUT Event details
                LOOP FOR EACH Registration (associated with Event ID) in the EVENTREGISTER table
                        OUTPUT Whanau details
                END LOOP
        END IF
END LOOP
```

Required tests:

7.1. Produce the report with all requested fields present.

Competitors Report: Screen Design





Task 8: Two Extra Features

- You need to add appropriate two extra features of your choice which should improve the usefulness of this application

Marking

It is expected that the code for each of the program tasks that you finish does not contain compilation errors (i.e. no marks will be awarded if the code does not compile). Your application must update the database every time the user changes the data. You are also expected to handle exceptions by using MessageBoxes and "**try and catch**" blocks.

Requirement	Marks
Task 1: Main Menu	10
Task 2: Kai Maintenance	10
Task 3: Events Maintenance	10
Task 4: Whanau Maintenance	10
Task 5: Location Maintenance	10
Task 6: Register Whanau to Event	10
Task 7: Events Report	15
Task 8: Two appropriate extra features are added	10
Testing Documentation (Minimum 25 Test Cases)	10
Naming conventions and correct internal documentation	5
Total:	100

Notes on marking

- Marks will be deducted for any task that is either not fully functional or not tested adequately.
- Check "Programming Standards for C Sharp Courses" at the end of this document. The standards for C# programming in this document MUST be followed. In particular, this includes putting meaningful comments at the beginning of **each and every** method in the standard format as given in the document.

Have a query? Want to improve your work?

You could:

- Visit Student Learning and Achievement for learning advice and support.
- Visit the Pacific Centre.
- Visit Te Noho Kotahitanga Marae (building 171)

Lecturer Assistance

If you have any questions about the assignment, then please email them to me. If you require assistance with your assignment, then please make an appointment outside of class time to see me. Please keep a record of all the work that you do on your assignment in a progress log (use the template provided on the next page).

Deadline for questions

Please note that there is a deadline for questions on the assignment at Checkpoint 2. Please make sure that you ask any questions regarding the assignment before this deadline.

Assignment 1 Progress Log

If you require any help while doing your assignment, please make sure that you bring along your up-to-date progress logs. Use the template below:

Date & Time:	
Name of current version of project:	
Task(s) undertaken:	
Source(s) of solution(s):	
Problem(s) encountered:	
Progress made:	
Name of new version of project:	
Time Taken:	

Deliverables

Soft Copy

The soft copy of the assignment is to be submitted in one ***.zip** (zip up your project while it is in the C:\Temp folder and please do not use any other formats such as **.rar**) file via Moodle and is to include:

- A soft copy of ALL files needed to compile and run your application from the **Visual Studio .NET Community 2017 or 2019** environment used in the Department of Computing labs. Please note that you must submit your assignment in **Visual Studio .NET Community 2017 or 2019**. 30 marks will be deducted if this is not done.
- Testing documentation.
- Progress Log

Please note that it is important to upload the correct version of your assignment onto Moodle. If you submit the wrong version onto Moodle, please notify me by email before the deadline date or late penalties may be incurred.

User Interface

Always provide the users with clear instructions explaining what they should do. Areas used for input must be labelled to explain what input is required. Use **hints** or **tool tips** to explain interface features. The user must be prevented from entering values or taking actions that the program is unable to deal with. All input should be validated (use the MS Access database tables to see what is allowed in each field) and to see which fields are required); any errors found should be reported back to the user with an error message which clearly and politely explains how to correct the error. The user should be unable to proceed without correcting invalid input.

Assignment Delivery

Electronic submission of all of the necessary files is required for ALL assignments and must be submitted prior to the due date and time. Assignments submitted after the due date and time without having received an extension through Affected Performance Consideration (APC) will be penalised according to the following:

- 10% of marks deducted if submitted within 24hrs of the deadline
- 20% of marks deducted if submitted after 24hrs and up to 48hrs of the deadline
- 30% of marks deducted if submitted after 48hrs and up to 72hrs of the deadline
- No marks will be awarded for an assignment that is submitted later than 72hrs after the deadline.

For the purposes of academic integrity, students who haven't demonstrated progress work in the class time can be asked to demo/test their working code and explain logic to the lecturer individually after assignment submission.

Affected Performance Consideration

A student, who due to circumstances beyond his or her control, misses a test, final exam or an assignment deadline or considers his or her performance in a test, final exam or an assignment to have been adversely affected, should complete the Affected Performance Consideration (APC) form available from Student Central.

When requesting an APC for an assignment, the APC application form must be submitted (along with work completed to-date) within the time frame of the extension requested; i.e. if the Doctor's certificate is for one (1) day, then the APC application form and work completed must be submitted within one (1) day.

Assistance to other Students

Students themselves can be an excellent resource to assist the learning of fellow students, but there are issues that arise in assessments that relate to the type and amount of assistance given by students to other students. It is important to recognise what types of assistance are beneficial to another's learning and also what types of assistance are unacceptable in an assessment.

Beneficial Assistance

- Study Groups.
- Discussion.
- Sharing reading material.
- Testing another student's programming work using the executable code and giving them the results of that testing.

Unacceptable Assistance

- Working together on one copy of the assessment and submitting it as own work.
- Giving another student your work.
- Copying someone else's work. This includes work done by someone not on the course.
- Changing or correcting another student's work.
- Copying from books, Internet etc. and submitting it as own work. Anything taken directly from another source must be acknowledged correctly: show the source alongside the quotation.

For the purposes of academic integrity, students who haven't demonstrated progress work in the class time (and/or no check point submission) can be asked to demo/test their working code and explain logic to the lecturer individually after assignment submission.

Programming Standards for C Sharp Courses

Internal Documentation

- Your code is such that other programmers can read it without struggling and your users are not left guessing as to what to do.
- Each class file (including form classes) will begin with comments explaining the purpose of the class, the author and the date written.
- Each method will start with comments that explain what the method does.
- Any code which does not have an obvious meaning or which uses a specialized technique is to be commented. Use blank lines and further comments to identify where parts of a task begin within a method.
- Code will use meaningful variable, class and method names. Components which have event handler code for any of their events must have meaningful names. Components which have properties assigned to in code must also have meaningful names. A naming convention that identifies the type of component involved is recommended. e.g. btnExit, txtStartDate
    - Note: Since there is no in-code use of labels in the assignment, label variable names will **not** be checked for compliance with the variable naming conventions.

Layout

- Code will be laid out in the style of the example below, using indentation steps of 4 spaces. Blocks using { and } will use the layout shown here:

```
///<Summary> method: btnLeapYear_Click
///Check if a date falls in a leap year
///</Summary>
private void btnLeapYear_Click(object sender, System.EventArgs e)
{
    DateTime aDate = getDate();
    if ( DateTime.IsLeapYear(aDate.Year) )
    {
        label6.Text = aDate.Year.ToString() + " IS a leap year";
    }
    else
    {
        label6.Text = aDate.Year.ToString()
                        + " is NOT a leap year";
    }
}
```

- Parentheses and spaces will be used to make the meaning clear in arithmetic expressions and conditions:

```
sum = (n1 / n2) + n3;
```

not

```
sum = n1 / n2 + n3;
```

not

```
sum=n1/n2+n3;
```

- In general, each method will perform a single simple task.

- In the final version of your project please delete all sections of code that have been ‘commented out’