|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Qualification details** | | | | | | | | | | | | | | | |
| **Training Package Code and Title** | | ICT - Information and Communications Technology (Release 7.2) | | | | | | | | | | | | | |
| **Qualification National Code and Title** | | ICT40120 Certificate IV in Information Technology (Release 2) | | | | | | | | | | | | **State code** | BFF9 |
| **Assessment Title** | | Assessment Task Three Individual Project | | | | | | | | | | | | | |
| **Unit National Code & Title** | | ICTPRG440 Apply introductory programming skills in different languages | | | | | | | | | | | | | |
| ICTPRG437 Build a user interface | | | | | | | | | | | | | |
| ICTICT435 Create technical documentation | | | | | | | | | | | | | |
| **Date Due** | | Week Nineteen | | | | | **Date Received** | | | | |  | | | |
| **Student Name** | | Jervin Alejandro | | | | | | | | | **Student ID** | | | | 30009690 |
| **Student Declaration** | | I declare that the evidence submitted is my own work:  Jervin.Alejandro  ………………………………………….. | | | | | | | | | | | | | |
| **Assessor Name** | |  | | | | | | | | | | | | | |
| **Assessment Decision** | |  | Satisfactory | | | | |  | Not Yet Satisfactory | | | | | | |
| **Assessor Signature** | |  | | | | | | | **Date** | | | | |  | |
| **Is student eligible for reassessment (Re-sit)?** | |  | | No |  | Yes | | | **Reassessment Date** | | | | | Week Twenty | |
| **Feedback to student** | | | | | | | | | | | | | | | |
| Via Blackboard (LMS) – Please check [Grade] section. | | | | | | | | | | | | | | | |
| **Feedback from student** | | | | | | | | | | | | | | | |
| Via Blackboard (LMS) – Please use [Comment] section during submission. | | | | | | | | | | | | | | | |
| **Student signature** | Jervin.Alejandro | | | | | | | | | **Date** | | |  | | |

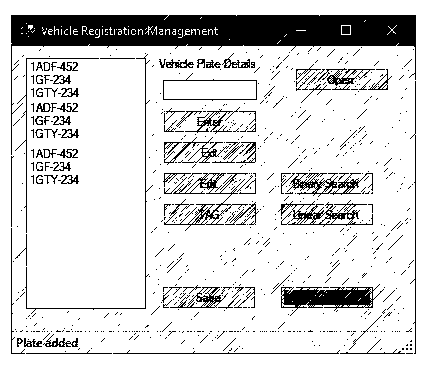
|  |  |
| --- | --- |
| **Assessment Instructions** | |
| **TO THE ASSESSOR** | |
| Type of Assessment | Individual Project |
| Duration of Assessment | 5 Class Sessions (Week 15 - 19) |
| Location of Assessment | Classroom |
| Conditions | Assessor to ensure that the noise levels, natural interactions and time variances are maintained as it would be in the Software Development industry.  Learners are required to complete the required tasks in class and submit the required documentation electronically via Blackboard  In order to verify the authenticity of the student’s assessment, you may ask the student to again produce an answer to an existing question.  Ensure you observe students writing code and using the features of the IDE (debug, trace, error correction, intelliSense, etc) in the classroom. |
| Elements and Criteria | As detailed in the assessment plan  You are required to make sure that all students meet the elements, performance criteria and oral communication items as outlined in the provided checklist and marking guide. |
| **TO THE STUDENT** | |
| Purpose of Assessment | You are required to show you can:  ICTPRG440 Apply introductory programming skills in different languages   * Demonstrate your skills by creating and manipulating data structures. * Create a forms based application that utilises binary search and sort algorithms. * Debug and test programming code using organisational standards * Record and document testing session.   ICTPRG437 Build a user interface   * Demonstrate your knowledge by researching prototyping tools and application development languages. * Investigating organizational guideline, policies and procedures.   ICTICT435 Create technical documentation   * Demonstrate your knowledge of technical document styles and design. * Investigate organisational policies, procedures and standards that cover document design. * Document scripts for internal and external stakeholders. * Collaborate and discuss ideas and requirements with client.   The student must demonstrate the ability to complete programming, testing and documentation tasks outlined in this assessment and is expected to use systematic analytical processes and effect time management to meet the goals/deadlines outlined in the DAP.  You are required to meet the elements, performance criteria and oral communication items as outlined in the provided checklist. |
| Allowable Materials | Blackboard (Topic by topic) will include the following: Weekly Readings, Class notes, and Weekly Activities.  Web links and example code can be downloaded from the Blackboard portal |
| Required Resources | Computer with:   * Internet Access to MSDN * Internet Access to www.citems.com.au/ * Access to MS Visual Studio * Access to MS Office or similar * Access to Blackboard (LMS) |
| Reasonable Adjustment | In some circumstances, adjustments to assessments may be made for you. If you require support for literacy and numeracy issues; support for hearing, sight or mobility issues; change to assessment times/venues; use of special or adaptive technology; considerations relating to age, gender and cultural beliefs; format of assessment materials; or presence of a scribe you need to inform your lecturer. |
| Assessment Submission | All questions and activities must be attempted.  Use of research tools and peers in formulating answers are acceptable – but work submitted must be your own work.  Final project documentation is to be uploaded to the appropriate area in the Blackboard course created for this unit.  If you are marked as NYS (Not Yet Satisfactory) on your first attempt, you will be provided with another opportunity to re-attempt the assessment. |
| Project Contents | This project consists of the following tasks:  User Interface Design  Q1. Design User Interface and Client Sign off  Prototype Development  Q2 – Q14 Create methods and code to satisfy the User and Programming requirements.  Technical Documentation  Q15 – Q18 Create Document Template and Training Resource.  Debug and Testing  Q19 User Interface testing  Q20 Logic debug |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Assessor Marking Guide | | Satisfactory | | **Comment** |
| **Questions** | | YES NO | |  |
| Q1 | UI Design Specifications satisfies the Clients requirements and has been approved by the Assessor. |  |  |  |
| Q2 | The global data structure is a List<> of type string. |  |  |  |
| Q3 | Open file into List, ensure the program opens the correct file name and data is in clear text. |  |  |  |
| Q4 | Enter new data, data is added to the List<>, error trapping if Null or empty, Uses real data in demonstration. |  |  |  |
| Q5 | Leave (delete data), data is removed from the list, error trapping id Null or empty. User can enter data into text box or select a record form the list box. |  |  |  |
| Q6 | Edit single date item, data is selected from the listbox and updated, then the button writes the data back to the List<>. A duplicate record must not be created. Error trapping if Null or empty. |  |  |  |
| Q7 | Reset clear all data in List, all data is removed from the list and the textbox and list box |  |  |  |
| Q8 | Display a single data item, A List Box click method allows the user to select a record that will be displayed in the text box |  |  |  |
| Q9 | Display and Sort, the data is sorted using the built-in method. This method MUST be used inside the Add, Edit, and Leave methods. |  |  |  |
| Q10 | Binary Search, Input search data is used to search the List<> using the built-in method. Error trapping if Null or empty. Text box is clear if not found. User feedback is required for found and not found outcomes |  |  |  |
| Q11 | Linear Search, Linear search code must be implemented (cannot use any built-in methods) Error trapping if Null or empty. Text box is clear if not found. User feedback is required for found and not found outcomes |  |  |  |
| Q12 | Close / Save data to file, Ensure the program saves the correct file name and data is in clear text. |  |  |  |
| Q13 | Tag single data item, A selected data item from the List Box is identified with a prefix “Z” character. The List<> is re-sorted and displayed. |  |  |  |
| Q14 | Tool Tips and Status Strip, All error messages and user feedback is displayed in a status strip at the bottom of the GUI form. All buttons, textbox and listbox have meaningful tool tips. Tool tips cannot be single words (ie Button or Edit Button) |  |  |  |
| Q15 | Template Specifications satisfies the Clients requirements and has been approved by the Assessor. |  |  |  |
| Q16 | Source code/comments document has correct information and file name and has been approved by the Assessor. |  |  |  |
| Q17 | Training materials satisfies the Clients requirements and has been approved by the Assessor. |  |  |  |
| Q18 | Technical Documentation Design Specifications and Training Resources have passed the approval and sign off |  |  |  |
| Q19 | UI testing has satisfied all the clients requires as listed in the UI testing document |  |  |  |
| Q20 | The Linear Search test satisfies all the criteria in the Code testing document |  |  |  |
|  | **Assessment Decision**  Satisfactory  Not Yet Satisfactory | | | |

# Scenario

You have accepted the role of a Contract Programmer for CITE Managed Services working to create a car park tracking and billing system for a company called Active Systems PTY. Your task is to create a prototype program to demonstrate to the client how the program can track vehicle registrations in a large city car park. In this task you will create a new Windows Forms Application which will demonstrate the use of a List<string> to collect and process the vehicle registration plate information. The program will take data from a camera which reads the registration plates of each vehicle that enters and leaves the car park. Your prototype will read and write data to a file which will simulate the camera, the basic operations will be manually operated using a series of buttons. If the prototype is successful, the process will be automated and integrated into a larger software solution. You will be required to fully test the final prototype and supply technical documentation and training materials.

The purpose of this assessment is to demonstrate the use of a List<> data structure to add, edit and delete data. The information implied by the following graphic needs to be read/written into a single text file when the program methods are invoked. The details of the project are listed in the client requirements and program requirements.



You should consult with the CITE representative (Your Lecturer) if you are unsure about any of the criteria or questions. Your primary research should focus on the resources on the Blackboard website, additional information can be collected from the Internet, ensure all sources are referenced. You must write your answers into the forms provided in this document and reports which conform to the client’s format and has been approved by the CITE representative (Your Lecturer). The following weekly schedule should be used to manage your workflow.

## Weekly Schedule

|  |  |  |  |
| --- | --- | --- | --- |
| TIMELINE | | TASK | DESCRIPTION |
| Week One  (week 15) | UI Design and Client Sign off | Question 1 | Create a UI design which reflects all the Client’s specifications. |
| Week Two  (week 16) | Prototype Development | Question 2 – 14 | Create the UI and associated Program Functionality using the Clients criteria |
| Week Three  (week 17) | Refine the Prototype | Question 2 - 14  Question 15, 16, 17 | Refine the prototype and add internal technical documentation.  Design and create external documentation |
| Week Four  (week 18) | Testing and Documentation | Question 18, 19, 20 | Debug and Test the Prototype. |
| Week Five  (week 19) | Finalise and Demonstration | All | Demonstrate Prototype and present all documentation. |

# User Interface Design

## Client Requirements

The Client would like the prototype application to incorporate the following User Interface (UI) requirements with a separate button for each of the following functions,

* Open text file and load data.
* Save data to text file.
* Add new rego plate.
* Delete an existing rego plate.
* Edit or update and existing rego plate.
* Linear Search for a specific rego plate.
* Binary Search for a specific rego plate.
* Tag a specific rego plate for future investigation.
* A reset button to remove all rego plate data from the List<>.

The UI will require a listbox or similar to display all the values from the List<> and a textbox for user input. The Client requires user feedback for all errors and input processing via a status strip at the bottom of the UI.

# Question One

### Instructions

Create a new word document using the following template: ensure it has all the sections as shown below, then fill in each of the sections of the UI Design Specification document.

List all the UI components (buttons, textbox, listbox, etc) and describe what action or event is associated with each component.

Create a content flow diagram for the ADD and EDIT button functions which shows the sequence of events associated with these actions when the user clicks the associated button.

Create and design the UI layout, insert a detailed image/picture/screen capture of the design, ensure all the features are identified and labelled. Consult and refer to the CITEMS guidelines for during the GUI design.

Once you have completed this task arrange for the completed UI Design Specification to be reviewed by the Lecturer/Assessor for approval, sign off and feedback before starting the prototype development.

ICTPRG437 E1, E2.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| User Interface Design Specification | | | | | | |
| User Interface Development Tool | | | InkScape and Visio | | | |
| Application Development IDE | | | Visual Studio | | | |
| Development Language | | | C# | | | |
| User Interface Components | | | | | | |
| List all UI components and their associated action/event (describe what each component does). | | | | | | |
| Component | Description of Action/Event | | | | | |
| Open Button | Allows the user to open saved text file. | | | | | |
| Save Button | Allows the user to save the entire list in a text file. | | | | | |
| Add Button | Allows the user to add registration plate to the list by typing them into the text field. | | | | | |
| Delete Button | Allows user to remove the selected registration plate from the list. | | | | | |
| Edit Button | Allows the user to edit the registration plate displayed in the list. | | | | | |
| Linear Search Button | Allows the user to locate a specific registration plate. | | | | | |
| Binary Search Button | Allows the user to locate a particular registration plate. However, the list must first be sorted. | | | | | |
| Tag Button | Allows the user to flag a registration plate for future inspection. | | | | | |
| Reset Button | Allows the user to erase all registered plates from the list. | | | | | |
| Content Flow Diagram | | | | | | |
| Create a content (user) flow diagram for the ADD and EDIT button functions.  Add Button  Edit Button | | | | | | |
| UI Design Layout | | | | | | |
| Insert the UI Diagram/Image here | | | | | | |
| Approval (Lecturer/Assessor use only) | | | | | | |
| Approver Name | | Title | | Signature | Date | Approved? |
|  | |  | |  |  |  |
|  | |  | |  |  |  |
| Lecturer Feedback | | | | | | |
|  | | | | | | |
| Recommendations | | | | | | |
|  | | | | | | |

# Prototype Development

The Client has listed the following programming requirements and functionality for the prototype. Use the previously approved UI design to create a Windows Application Form which satisfies these criteria. Ensure all programming code and comments adheres with CITEMS organisational standards and guidelines.

## Program Requirements

# Question Two

The prototype must use a List<> data structure of data type “string”.

# Question Three

**OPEN**: When the OPEN button is clicked the user can select different data from pre-saved text files. The text files must be named for difference scenarios where the file names are “demo\_01.txt”, “demo\_02.txt”, etc.

# Question Four

**ENTER**: To add a rego plate to the List<> the user will type the data value (rego plate info) into the TextBox and click the **ENTER** button. The data will be added to the List<> and the TextBox will be cleared, and the cursor will focus in the TextBox. If the TextBox is empty the program should raise an error message. All data added to the List<> must be “real”, you cannot demonstrate or submit using data like: Rego Plate Info: “aa”, “bb”, etc.

# Question Five

**LEAVE**: To remove rego plate: There are two options to remove a rego plate item from the List<>. Method One: double click a data item from the ListBox and click the OK button in the popup dialog box. The data item will be removed from the List<>. Method Two: the user will enter the rego plate information into the TextBox and click the DELETE button. The data will be removed from the List<> and the TextBox will be cleared, and the cursor will focus in the TextBox.

# Question Six

**EDIT**: To edit a rego plate click (select) a data item from the ListBox so that is appears in the TextBox. Alter the information and click the EDIT button. The updated information is written back to the List<> and the TextBox is cleared, and the cursor refocus in the TextBox. (you are not permitted to use an add/delete option)

# Question Seven

**RESET**: Add a RESET button to clear all the data items from the List<>. The ListBox and TextBox should also be cleared.

# Question Eight

**SINGLE DATA DISPLAY**: Create a single click method to do the following: when a data item is selected from the ListBox on the left, the information is displayed in the TextBox on the right.

# Question Nine

**DISPLAY and SORT**: All the rego plates should be displayed in the ListBox which is sorted alphabetically using the built-in List Sort method. The List<> must be sorted after every List<> process (add, edit, delete, etc). The rego plate information should be unique.

# Question Ten

**BINARY SEACH**: To find a specific rego plate the user will type the information into the TextBox and click the BINARY SEARCH button. If the rego plate is found, then a confirmation message should be displayed. If the rego plate is not found, then a message should be displayed, and the TextBox cleared, and the cursor refocused. The search code must use the built-in Binary search.

# Question Eleven

**LINEAR SEARCH**: Add a second search button that implements a linear search algorithm. To find a rego plate the user will type the information into the TextBox and click the LINEAR SEARCH button. If the rego plate is found, then a confirmation message should be displayed. If the rego plate is not found, then a message should be displayed, and the TextBox cleared, and the cursor refocused.

# Question Twelve

**CLOSE and SAVE**: Create a method to save the data, this method will open a save dialog box and allow the user to save all the data back to a text file. The dialog box must have a filter to display text files only. Add a SAVE button that can utilise the save method. Create a FORM closing method using the save method so all data from the List<> will be written back to a single text file called “demo\_##.txt” file which is auto incremented (ie demo\_01.txt, demo\_02.txt, etc).

# Question Thirteen

**TAG**: Create tag method and associated TAG button to mark a rego plate. When a rego plate is selected from the ListBox and “tagged” an additional character value “z” will be prefixed to the rego plate. The List<> will be re-sorted and displayed.

# Question Fourteen

**USER INTERFACE (UI)**: Add a tool tip text to each of the controls (TextBox, ListBox, Buttons). Add a status strip at the bottom of the form to display error messages and general user feedback.

# Technical Documentation

The Client has requested documentation to support the prototype project to be delivered at the final handover.

Furthermore, the Client will require training materials during the roll out period for all their front-line technical personnel. Use the following information to create each of the documents; ensure you have the Client’s approval before proceeding to the debugging and testing stage.

### Client Details

Company name: Acme Systems Pty

Company logo:



Note: the logo image can be downloaded from Bb.

# Question Fifteen

### General Documents

Design and create a suitable template document for the Source Code of the prototype with the following;

* Company Logo in top left-hand corner of page header. Size 1.25cm square.
* Company Name in top right-hand corner. Font Cooper Std Black and size 16.
* Page numbers in the middle of bottom footer. Default font and size.
* Date field in left hand corner of bottom footer. Default font and size.
* Body code font Courier New and size 10
* Save the template as “AcmeTemplate.dotx”

# Question Sixteen

Using the template from question 15, copy and paste all the code and comments into the template and save as “SourceCode.docx”. Review the layout and readability to ensure the document is ready for distribution and submission.

# Question Seventeen

### Training Materials

Create a 6-slide PowerPoint training presentation and associated notes. The presentation should explain the usage of the application and each of the controls. Ensure your presentation includes image(s) of the UI; use callout shapes to explain the function of all the buttons, etc. For each slide add suitable notes which can be used during the presentation. You will need to spell and grammar check all text on the slides and the slide notes.

Title Slide must have the following,

* Company logo in the middle, size 9.55cm square.
* Company name under logo, font Cooper Std Black and size 28.

Content slides must have the following.

* Client Logo in top left-hand corner of page header. Size 1.25cm square
* Client Name in top right-hand corner.

The slides must cover the following topics

Slide One: Title

Slide Two: Button layout.

Slide Three: Save and Load data.

Slide Four: UI layout and features.

Slide Five: Error message and confirmation notifications.

Slide Six: Questions.

# Question Eighteen

### Submission Requirements

Once you have completed questions 15, 16 & 17 use the following Technical Documentation Design Specifications checklist to determine whether you have completed all the requirements for the Technical Documentation.

Arrange for the completed checklist to be reviewed and assessed by the Lecturer/Assessor for approval, sign off and feedback before commencing the Debugging and Testing.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Technical Documentation Design Specifications | | | | | | |
| Template Criteria | | | | |  | |
| 1. Company Logo in top left-hand corner of page header. Size 1.25cm square. | | | | | Yes ☐ | No☐ |
| 1. Company Name in top right-hand corner. Font Cooper Std Black and size 16. | | | | | Yes ☐ | No☐ |
| 1. Page numbers in the middle of bottom footer. Default font and size. | | | | | Yes ☐ | No☐ |
| 1. Date field in left hand corner of bottom footer. Default font and size. | | | | | Yes ☐ | No☐ |
| 1. Body code font Courier New and size 10 | | | | | Yes ☐ | No☐ |
| 1. Was the template created using an industry standard software package | | | | | Yes ☐ | No☐ |
| 1. Is the Source Code template in the correct format “.dotx”? | | | | | Yes ☐ | No☐ |
| 1. Is the template document ready for publication and distribution? | | | | | Yes ☐ | No☐ |
| Feedback and General Comments | | | | | | |
|  | | | | | | |
| Source Code Document | | | | | | |
| 1. Is the C# code and comments is in the correct template,   Check the File > Info > Properties. | | | | | Yes ☐ | No☐ |
| 1. Is the document saved with the correct name and extension “SourceCode.docx”? | | | | | Yes ☐ | No☐ |
| 1. Is the Source Code document is ready for publication and distribution? | | | | | Yes ☐ | No☐ |
| Feedback and General Comments | | | | | | |
|  | | | | | | |
| Training Materials Documentation Criteria | | | | | | |
| 1. The Presentation has a minimum of 6 slides | | | | | | |
| Slide 1: PPT has a tile slide with Company information | | | | | Yes ☐ | No☐ |
| Slide 2: Button layout with suitable details and explanations using callouts (or similar) | | | | | Yes ☐ | No☐ |
| Slide 3: Details related to Save and Load of data, includes naming convention for files. | | | | | Yes ☐ | No☐ |
| Slide 4: User Interface with details of Input Text, Display of List and all Buttons | | | | | Yes ☐ | No☐ |
| Slide 5: Details of all error message and notifications | | | | | Yes ☐ | No☐ |
| Slide 6: Final slide with option for participants to ask questions | | | | | Yes ☐ | No☐ |
| 1. Does each slide have accompanying notes? | | | | | Yes ☐ | No☐ |
| 1. Are the notes spell checked? | | | | | Yes ☐ | No☐ |
| 1. Are the notes grammar checked? | | | | | Yes ☐ | No☐ |
| 1. Is the Training Materials ready for publication and distribution? | | | | | Yes ☐ | No☐ |
| Feedback and General Comments | | | | | | |
|  | | | | | | |
| Approval (Lecturer/Assessor use only) | | | | | | |
| Approver Name | Title | Signature | Date | Approved? | | |
|  |  |  |  |  | | |
|  |  |  |  |  | | |
| Lecturer Feedback | | | | | | |
|  | | | | | | |
| Recommendations | | | | | | |
|  | | | | | | |

# Debugging and Testing

The client has requested a series of tests of the final prototype that will satisfy the following client requirements.

# Question Nineteen

**USER INTERFACE TEST:** Create a User Interface Test Table using the Template from question 15 with the following headings as shown below. These tests will ensure the full User Interface functionality of your prototype and the various error trapping/user messages prior to your demonstration. Use the following list of client requirements to create, perform and record the User Interface tests: your tests must include the following.

* add data,
* delete data,
* edit data,
* binary search,
* linear search,
* tag data item,
* load,
* save.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| User Interface Testing | | | | | | | | | | | | | | | |
| **Project Name:** | | | | Vehicle Registration | | | | | | | | | | | |
| **Type of Test:** | | | | User Interface Test | | | | | | | | | | | |
| **Version No:** | | | | 1.1 | **Client Name:** | | | Acme Systems Pty | | | | | | | |
| **Description:** | | | | Check full user interface functionality of the vehicle registration prototype and the various error trapping or user messages. | | | | | | | | | | | |
| **Date:** | | | | 17/11/2021 | | | | | | | | | | | |
| **Developer:** | | Jervin Alejandro | | | | | **Tester:** | | Jervin Alejandro | | | | | | |
| Test Case No | Test Case Name | | | Test steps | | | | Test Data | | | | Expected result | | | Pass / Fail |
| 1 | Check status strip when program is loaded | | | 1. Launch Visual Studio and open the Lists program. 2. Click the start button. | | | |  | | | | Display “Default text file successfully loaded” | | | PASS |
| 2 | Check enter button status strip when adding a valid registration plate | | | 1. Launch Visual Studio and open the Lists program. 2. Click the start button. 3. Enter a valid registration plate number in the text field. 4. Click Enter button. | | | | 1GAL003 | | | | Display “Add Success” | | | PASS |
| 3 | Check enter button status strip when adding an empty text | | | 1. Launch Visual Studio and open the Lists program. 2. Click the start button. 3. Enter 6 whitespaces in the text field. 4. Click Enter button. | | | | “ ” | | | | Display “Text box is empty” | | | PASS |
| 4 | Check enter button status strip when adding an invalid registration plate | | | 1. Launch Visual Studio and open the Lists program. 2. Click the start button. 3. Enter an invalid registration plate number in the text field. 4. Click Enter button. | | | | aaaaa | | | | Display “Registration plate is invalid” | | | PASS |
| 5 | Check enter button status strip when adding a duplicate registration plate | | | 1. Launch Visual Studio and open the Lists program. 2. Click the start button. 3. Enter an existing registration plate number in the text field. 4. Click Enter button. | | | | 1GAL003 | | | | Display “Registration Plate already exist” | | | PASS |
| 6 | Check delete button status strip when deleting a registration plate using a the double click method | | | 1. Launch Visual Studio and open the Lists program. 2. Click the start button. 3. Double-click a plate number in the list. | | | |  | | | | Display “Remove Success” | | | PASS |
| 7 | Check delete button status strip when deleting a registration plate using text box | | | 1. Launch Visual Studio and open the Lists program. 2. Click the start button. 3. Enter an existing registration plate in the text field 4. Click Delete button. | | | | 1GAL003 | | | | Display “Remove Success” | | | PASS |
| 8 | Check delete button status strip when deleting a non-existing registration plate using text box | | | 1. Launch Visual Studio and open the Lists program. 2. Click the start button. 3. Enter a non-existing registration plate in the text field 4. Click Delete button. | | | | 1GAL003A12 | | | | Display “Registration plate does not exist” | | | PASS |
| 9 | Check delete button status strip when textbox is empty | | | 1. Launch Visual Studio and open the Lists program. 2. Click the start button. 3. Enter 6 whitespaces in the text field. 4. Click Delete button. | | | | “ ” | | | | Display “Text box is empty” | | | PASS |
| 10 | Check edit button status strip when editing a registration plate with an existing registration plate | | | 1. Launch Visual Studio and open the Lists program. 2. Click the start button. 3. Click on the first registration plate on the list. 4. Enter an existing registration plate in the text field. 5. Click Edit button. | | | | 9XJZ768 | | | | Display “Registration plate already exist” | | | PASS |
| 11 | Check edit button status strip when editing a registration plate with non existing registration plate | | | 1. Launch Visual Studio and open the Lists program. 2. Click the start button. 3. Click on the first registration plate on the list. 4. Enter a non-existing registration plate in the text field. 5. Click Edit button. | | | | 1GAL003 | | | | Display “Edit Success” | | | PASS |
| 12 | Check edit button status strip when editing a registration plate with an empty text field | | | 1. Launch Visual Studio and open the Lists program. 2. Click the start button. 3. Click on the first registration plate on the list. 4. Enter 6 whitespaces in the text field 5. Click Edit button. | | | | “ “ | | | | Display “Text Box is empty” | | | PASS |
| 13 | Check tag button message output when tagging a registration plate | | | 1. Launch Visual Studio and open the Lists program. 2. Click the start button. 3. Click on the first registration plate on the list. 4. Click Tag button. | | | |  | | | | Display “Tag Success” | | | PASS |
| 14 | Check tag button message output when untagging a registration plate | | | 1. Launch Visual Studio and open the Lists program. 2. Click the start button. 3. Click on a tagged registration plate on the list. 4. Click Tag button. | | | |  | | | | Display “Registration plate untagged” | | | PASS |
| 15 | Check binary search button message output when searching for a non existing registration plate | | | 1. Launch Visual Studio and open the Lists program. 2. Click the start button. 3. Enter a non-existing registration plate in the text field. 4. Click Bin search button. | | | | 7KKK999 | | | | Display “Registration plate does not exist” | | | PASS |
| 16 | Check binary search button message output when searching for an empty text | | | 1. Launch Visual Studio and open the Lists program. 2. Click the start button. 3. Enter 6 whitespaces in the text field. 4. Click Bin search button. | | | | “ “ | | | | Display “Text box is empty” | | | PASS |
| 17 | Check binary search button message output when searching for an existing registration plate | | | 1. Launch Visual Studio and open the Lists program. 2. Click the start button. 3. Enter an existing registration plate in the text field. 4. Click Bin search button. | | | | 9XJZ768 | | | | Display “Search Success” | | | PASS |
| 18 | Check linear search button message output when searching for a non-existing registration plate | | | 1. Launch Visual Studio and open the Lists program. 2. Click the start button. 3. Enter a non-existing registration plate in the text field. 4. Click Lin search button. | | | | 4TYL000 | | | | Display “Registration plate does not exist” | | | PASS |
| 19 | Check linear search button message output when searching for an empty text | | | 1. Launch Visual Studio and open the Lists program. 2. Click the start button. 3. Enter 6 whitespaces in the text field. 4. Click Lin search button. | | | | “ “ | | | | Display “Text box is empty” | | | PASS |
| 20 | Check linear search button message output when searching for an existing registration plate | | | 1. Launch Visual Studio and open the Lists program. 2. Click the start button. 3. Enter an existing registration plate in the text field. 4. Click Lin search button. | | | | 0AHM954 | | | | Display “Search Success” and searched registration plate should be selected | | | PASS |
| 21 | Check save button output when clicked | | | 1. Launch Visual Studio and open the Lists program. 2. Click the start button. 3. Click save button. | | | |  | | | | Display Save file window | | | PASS |
| 22 | Check save button output when saving a text file | | | 1. Launch Visual Studio and open the Lists program. 2. Click the start button. 3. Click save button. 4. Enter demo\_1.txt 5. Click save. | | | | demo\_1.txt | | | | Display “Save success” and Create a text file called demo\_1.txt | | | PASS |
| 23 | Check save button output when cancelling the save | | | 1. Launch Visual Studio and open the Lists program. 2. Click the start button. 3. Click save button. 4. Click cancel | | | |  | | | | Display “Save cancelled” | | | PASS |
| 24 | Check open button output when clicked | | | 1. Launch Visual Studio and open the Lists program. 2. Click the start button. 3. Click open button. | | | |  | | | | Display Open file window | | | PASS |
| 25 | Check open button output when opening a text file | | | 1. Launch Visual Studio and open the Lists program. 2. Click the start button. 3. Click open button. 4. Enter demo\_1.txt 5. Click open. | | | | demo\_1.txt | | | | Display “Load success” and display the registration plates from demo\_1.txt in the list box | | | PASS |
| 26 | Check open button output when cancelling the open | | | 1. Launch Visual Studio and open the Lists program. 2. Click the start button. 3. Click open button. 4. Click cancel | | | |  | | | | Display “Load cancelled” | | | PASS |
| 27 | Check reset button output when confirming reset | | | 1. Launch Visual Studio and open the Lists program. 2. Click the start button. 3. Click Reset button. 4. Click ok. | | | |  | | | | Display “Reset Success” and clears the entire registration list. | | | PASS |
| 28 | Check reset button output when cancelling reset | | | 1. Launch Visual Studio and open the Lists program. 2. Click the start button. 3. Click Reset button. 4. Click cancel. | | | |  | | | | Display “Reset Cancelled” and registration list should stay the same | | | PASS |
| 29 | Check list box tool tip | | | 1. Launch Visual Studio and open the Lists program. 2. Click the start button. 3. Hover the cursor over list box | | | |  | | | | Display “List of registration plates” | | | PASS |
| 30 | Check text box tool tip | | | 1. Launch Visual Studio and open the Lists program. 2. Click the start button. 3. Hover the cursor over the text box | | | |  | | | | Display “Enter Registration Plate(Standard WA rego plate format is NLLLNNN)” | | | PASS |
| 31 | Check Enter button tool tip | | | 1. Launch Visual Studio and open the Lists program. 2. Click the start button. 3. Hover the cursor over the Enter button | | | |  | | | | Display “Add Registration Plate to list” | | | PASS |
| 32 | Check Delete button tool tip | | | 1. Launch Visual Studio and open the Lists program. 2. Click the start button. 3. Hover the cursor over the Delete button | | | |  | | | | Display “Remove Selected Registration Plate from the List” | | | PASS |
| 33 | Check Single display Text box tool tip | | | 1. Launch Visual Studio and open the Lists program. 2. Click the start button. 3. Hover the cursor over the Text box on the right | | | |  | | | | Display “Change the selected Registration Plate on the list.” | | | PASS |
| 34 | Check Edit button tool tip | | | 1. Launch Visual Studio and open the Lists program. 2. Click the start button. 3. Hover the cursor over the Edit button | | | |  | | | | Display ”Display Registration Plate” | | | PASS |
| 35 | Check Tag button tool tip | | | 1. Launch Visual Studio and open the Lists program. 2. Click the start button. 3. Hover the cursor over the Tag button | | | |  | | | | Display “Flag a Registration Plate for future inspection” | | | PASS |
| 36 | Check Bin search button tool tip | | | 1. Launch Visual Studio and open the Lists program. 2. Click the start button. 3. Hover the cursor over the Bin search button | | | |  | | | | Display “Search a particular Registration Plate in binary search method” | | | PASS |
| 37 | Check Lin search button status strip | | | 1. Launch Visual Studio and open the Lists program. 2. Click the start button. 3. Hover the cursor over the Lin search button | | | |  | | | | Display “Search a particular Registration Plate in linear search method” | | | PASS |
| 38 | Check Save button status strip | | | 1. Launch Visual Studio and open the Lists program. 2. Click the start button. 3. Hover the cursor over the Save button | | | |  | | | | Display “Save file to text format” | | | PASS |
| 39 | Check Open button status strip | | | 1. Launch Visual Studio and open the Lists program. 2. Click the start button. 3. Hover the cursor over the Open button | | | |  | | | | Display “Open a text file” | | | PASS |
| 40 | Check Reset button status strip | | | 1. Launch Visual Studio and open the Lists program. 2. Click the start button. 3. Hover the cursor over the Reset button | | | |  | | | | Display “Remove all list” | | | PASS |
| User Interface Testing Criteria | | | | | | | | | | | | | | | |
| All client requirements have been tested | | | | | | | | | | | | | Yes ☐ | | No☐ |
| All error messages have been tested | | | | | | | | | | | | | Yes ☐ | | No☐ |
| All user feedback messages have been tested | | | | | | | | | | | | | Yes ☐ | | No☐ |
| Test for valid and non-valid data | | | | | | | | | | | | | Yes ☐ | | No☐ |
| Dose this UI satisfy the all the clients UI requirements | | | | | | | | | | | | | Yes ☐ | | No☐ |
| Feedback and General Comments | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | |
| Approval (Lecturer/Assessor use only) | | | | | | | | | | | | | | | |
| Approver Name | | | Title | | | Signature | | | | | Date | | | Approved | |
|  | | |  | | |  | | | |  | | | |  | |
|  | | |  | | |  | | | |  | | | |  | |
| Lecturer Feedback | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | |

# Question Twenty

**SEARCH TEST:** Create a Code Test Table using the Template from question 15 with the following headings as shown below. The tests will determine the correct functionality of your Linear Search method. The client requires evidence that all the variables in the algorithm are traced using the built-in debug features of the IDE. Create, perform and record the tests; ensure you capture screen images of all the various variables and conditions (IF, While and For).

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Code Testing Criteria | | | | | | | | | | |
| **Project Name:** | | | Vehicle Registration Plate | | | | | | | |
| **Type of Test:** | | | Functional Testing | | | | | | | |
| **Version No:** | | | 1.1 | **Client Name:** | | Acme Systems Pty | | | | |
| **Description:** | | | The test will determine whether or not the Linear Search method is functioning correctly. | | | | | | | |
| **Date:** | | | 16/11/2021 | | | | | | | |
| **Developer:** | | Jervin Alejandro | | | **Tester:** | | Jervin Alejandro | | | |
| Test # | Test Case Name | | Test Data | | | Expected Result | | Evidence  (Screen Shots Below) | | PASS/FAIL |
| 1 | Check linear search status strip result when input is empty | | “ “ | | | Status strip should show “Text box is empty” | |  | | PASS |
| 2 | Check which if statement will be executed when input is empty | | “ “ | | | If text box is empty should be executed. | |  | | PASS |
| 3 | Check linear search status strip result when searching for a registration plate that is not in the list | | “1TST123“ | | | Status strip should show “Registration Plate does not exist” | |  | | PASS |
| 4 | Check which if statement will be executed when searching for a registration plate that is not in the list | | “1TST123” | | | If text box is not null or whitespace should be executed. | |  | | PASS |
| 5 | Check linear search status strip result when searching for a registration plate that is on the list | | “9XJZ768” | | | Status strip should show “Search Success” | |  | | PASS |
| 6 | Check which if statement will be executed when searching for a registration plate that is on the list | | “9XJZ768” | | | If found is true should be executed | |  | | PASS |
| 7 | Check linear search Boolean result when search is not found | | “5RTA634” | | | Boolean found should be false | |  | | PASS |
| 8 | Check linear search Boolean result when search is found | | “7AKE080” | | | Boolean found should be true | |  | | PASS |
| 9 | Check linear search for loop variable i when searching the fourth item on the list | | “2VYR093” | | | Variable i should be 3 since counting starts from 0 | |  | | PASS |
| 10 | Check linear search for loop variable i when searching an item not on the list | | “6AKH385” | | | Variable i should be 10 since there is 10 items in the list | |  | | PASS |
| Test Evidence #1 and #2  Test Evidence #3 and #4  Test Evidence #5 and #6  Test Evidence #7  Test Evidence #8  Test Evidence #9  Test Evidence #10 | | | | | | | | | | |
| Linear Search Testing Criteria | | | | | | | | | | |
| Test for null entry into TextBox | | | | | | | | | Yes ☐ | No☐ |
| Test IF compare statement | | | | | | | | | Yes ☐ | No☐ |
| Test loop variable | | | | | | | | | Yes ☐ | No☐ |
| Test error message | | | | | | | | | Yes ☐ | No☐ |
| Test Boolean found | | | | | | | | | Yes ☐ | No☐ |

## Demonstration and Submission

Demonstrate your working program to the lecturer before final submission to ensure your code and documentation is compliant. You must submit all program code and the written documentation to the appropriate Blackboard section. Consult your lecturer for further information.

End of Project