A black background with purple text

Description automatically generated

**WENTWORTH INSTITUTE OF HIGHER EDUCATION**

**FINAL EXAMINATION**

MIT211 Semester 1, 2024

Examination Duration: 170 minutes. Plus 10 minutes reading time

Total Marks:40 Marks

**Exam Conditions:**

This is a PRACTICAL examination

This is a open BOOK examination

**Materials Permitted In The Exam Venue:**

Computer system

Internet

**Submission Instructions:**

Create a Zip folder of your project and submit it on CANVAS

**Simple Contact Management Windows Form App in C#**

**Objective:** Develop a Windows Forms app to manage a basic contact list where users can add, edit, and delete contacts, and save/load the contact list from/to a file.

**Requirements:**

1. **Contact Data:** Each contact will have a name, phone number, email, and address.
2. **User Interface:** The main form should allow users to add, edit, delete, save, and load contacts.
3. **Data Storage:** The app should store the contact data in an ArrayList and have the ability to save this data to a file and load it back.
4. **Exception Handling:** Properly handle potential errors especially during file operations.

**Steps:**

1. **Project Setup:**
   * Start Visual Studio and create a new Windows Forms App project.
   * Add necessary controls: labels, text boxes (for name, phone number, email, address), buttons (add, edit, delete, save, load), and a ListView to display contacts.
2. **Create the Contact Class:**
   * Create a new class named Contact with properties for Name, PhoneNumber, Email, and Address.
3. **Initialize Data Store:**
   * Declare an ArrayList named contactList to store contact objects.
4. **Implement UI Interactions:**
   * **Add Contact:**
     1. Capture the details from text boxes.
     2. Create a new Contact object and add it to the contactList.
     3. Refresh the ListView to display the new contact.
   * **Edit Contact:**
     1. When a contact is selected in the ListView, populate the text boxes with the contact's details.
     2. Allow the user to edit the details and update the Contact object in the contactList.
   * **Delete Contact:**
     1. Allow users to select a contact from the ListView and remove it from the contactList.
5. **Implement File Operations:**
   * **Save Contacts:**
     1. Serialize the contactList to a file using StreamWriter.
     2. Use exception handling to manage potential write errors.
   * **Load Contacts:**
     1. Deserialize contact data from a file into the contactList using StreamReader.
     2. Use exception handling for file not found or read errors.
6. **Handle Events and Exceptions:**
   * Ensure every button click and user interaction is event-driven.
   * Handle potential exceptions especially during file operations (e.g., file not found, permission denied).
7. **Test the App:**
   * Add, edit, and delete contact entries.
   * Save the contact list to a file and load it back to ensure data persistence.

This exam task covers various aspects of Windows Forms app development in C#, from UI design to file operations, emphasizing best practices in object-oriented programming, event handling, and exception management.

**Marking Criteria: Windows Form App in C#**

Total Marks: 40

1. **UI Design & Usability** - [10 Marks]

* Clear layout and organization of controls: **3 marks**
* Consistent and intuitive UI/UX: **3 marks**
* Proper labeling and placeholders (if used): **2 marks**
* Responsiveness (UI should not freeze during operations): **2 marks**

2. **Class & ArrayList Implementation** - [5 Marks]

* Properly defined class with all required properties: **3 marks**
* Successful implementation and management of ArrayList: **2 marks**

3. **UI Interactions** - [8 Marks]

* **Add**: Captures input and adds to the ArrayList & ListView: **3 marks**
* **Edit**: Edits the selected details in both ArrayList & ListView: **2.5 marks**
* **Delete**: Removes the selected from both ArrayList & ListView: **2.5 marks**

4. **File Operations** - [8 Marks]

* **Save**:
  + Successfully writes to a file: **4 marks.**
  + Uses exception handling during write operations: **1 mark.**
* **Load**:
  + Reads the data from a file and populates: **2.5 marks.**
  + Uses exception handling during read operations (e.g., file not found): **0.5 marks.**

5. **Event Handling & Exception Management** - [7 Marks]

* Proper event-driven responses to button clicks and user interactions: **4 marks.**
* Comprehensive exception handling beyond just file operations (e.g., input validation): **3 marks**

6. **Overall Functionality and Testing** - [2 Marks]

* App runs without crashes and behaves as expected: **2 marks.**

**Note**: Marks will be deducted for errors, bugs, or incomplete implementations relevant to each section. If a student has used an alternative yet effective method to implement a feature, they should still be awarded appropriate marks for that section.