Level 1, 302-306 Elizabeth St:

Surry Hills, NSW, 2010

T: +61 (0)2 8252 9999

Level1, 15 Moore St:

Canberra, ACT 2601

T: +61 (0)2 6112 8839

info@win.edu.au

www.win.edu.au

**MIT 211 ADVANCED PROGRAMMING TECHNIQUES**

**Assessment 2: GUI Application Development**

**Group # 1**

|  |  |
| --- | --- |
| **Due date:** | Week 12 |
| **Weighting:** | 40% |
| **Length and Format**: | The Word file with 1500-2500 words, the working project files, test plan and demonstration. |
| **Assessment Details:** | Students working in small groups research, evaluate and provide an application such as a game, a Windows 10 application, or a cryptography application, and develop a graphical user interface (GUI) for it.  Deliverables:  1. source code  2. test plan  3. demonstration |
| **Assessment addresses SLOs** | C, D, E |
| **Marking Criteria** | Please refer to Marking Rubric on Canvas |

Submission Requirements

Students must submit their assignments through CANVAS shell. Email submissions will not be accepted.

As this is a group assignment, one student can submit the project on behalf of the group.

A Turnitin report must be generated.

**Deliverables**:

All source code of your system as a Zip file and the documentation in word format.

* One student (group leader) should submit the report in docx, or pdf format.
* One student (group member) should submit the zip file of the code.

# **Personal Finance Tracker**

**Overview**

Your task is to develop a Personal Finance Tracker in C#. This application will assist users in tracking their personal finances, enabling them to manage their expenses, income, and savings. The application should provide functionalities to add, delete, modify, and view financial transactions, categorize expenses, and generate reports.

**Objectives**

1. Create a C# Windows Forms Application.
2. Develop an intuitive Graphical User Interface (GUI) using the specified control objects.
3. Implement event handling to respond to user actions.
4. Manage data storage using file operations.
5. Execute functionalities: add, delete, modify, view, and search for financial transactions.

**Requirements**

**Control Objects:**

You are required to use the following control objects in your application:

1. **Labels**: To display field names such as transaction ID, date, description, category, amount, and search.
2. **TextBoxes**: For users to input transaction information and search criteria.
3. **Buttons**: To execute various actions like adding, deleting, modifying, and searching.
4. **DataGridView**: To display the list of financial transactions.
5. **MenuStrip**: To provide a menu for navigation.
6. **OpenFileDialog** and **SaveFileDialog**: To allow users to open and save financial data files.

**Features:**

1. **Add Transaction**: Users input transaction details, and upon clicking the 'Add' button, the transaction is added to the DataGridView and the data file.
2. **Delete Transaction**: Users select a transaction from the DataGridView and delete it via the 'Delete' button.
3. **Modify Transaction**: Users select a transaction from the DataGridView, adjust the details in the TextBoxes, and click the 'Modify' button.
4. **View Transactions**: Refresh the DataGridView each time a transaction is added, deleted, or modified.
5. **Search Functionality**: Use a TextBox and 'Search' button to find transactions based on transaction ID, date, description, category, or amount.
6. **File Handling**:
   * **Load Transactions**: Implement a "Load" option in the MenuStrip that uses OpenFileDialog to load financial data from a file.
   * **Save Transactions**: Implement a "Save" option in the MenuStrip that uses SaveFileDialog to save the current financial data to a file.
   * **Automatic Save**: Before the application closes, ensure that the financial data is saved or prompt the user to save changes.
7. **Event Handling**: Implement appropriate event handlers to manage button clicks, data modifications, and other user interactions.
8. **Data Validation**: Ensure only valid data is accepted, particularly for amount fields and while searching.
9. **Exception Handling**: Account for potential issues, such as file access errors or invalid data input.

**Additional Considerations:**

* Store financial data in a CSV or JSON format for simplicity.
* Implement efficient searching algorithms for quick retrieval.
* Ensure the design is user-friendly, with clear instructions and feedback.

**Submission:**

Provide the following:

1. Source code with appropriate comments.
2. Brief documentation detailing how to use the application and any special features you've added.
3. A sample data file to demonstrate the load functionality.

Your grade will be based on the application's completeness, functionality, code structure, and user experience.

Good luck and enjoy the coding journey!

# Marking Rubric

**Total Marks: 40**

**1. User Interface Design (Total: 8 Marks)**

* Layout and organization of control objects: 2 marks
  + Clear and organized layout: 2 marks
  + Disorganized or confusing layout: 0-1 marks
* Responsiveness and user feedback (e.g., validation feedback, action confirmations): 3 marks
  + Immediate and appropriate feedback: 3 marks
  + Delayed or unclear feedback: 0-2 marks
* Use of specified control objects: 3 marks
  + All specified controls implemented and functional: 3 marks
  + Missing or improperly implemented controls: 0-2 marks

**2. Functionality Implementation (Total: 10 Marks)**

* Add, Delete, Modify, and View functionalities: 4 marks
  + All functions working perfectly: 4 marks
  + One or more functions not working as expected: 0-3 marks
* Search functionality: 3 marks
  + Efficient and accurate search: 3 marks
  + Slow or inaccurate search: 0-2 marks
* File Handling (Load, Save, Automatic Save): 3 marks
  + All file operations working perfectly: 3 marks
  + Issues with one or more file operations: 0-2 marks

**3. Event and Exception Handling (Total: 8 Marks)**

* Effective event handling for all user interactions: 4 marks
  + Events handled and give expected output: 4 marks
  + Missing or flawed event handling: 0-3 marks
* Exception handling: 4 marks
  + Robust exception handling preventing crashes: 4 marks
  + Missing or inadequate exception handling: 0-3 marks

**4. Data Validation and Integrity (Total: 5 Marks)**

* Proper validation of numeric fields (quantity, price): 2 marks
  + Fully validated: 2 marks
  + Partially validated or not validated: 0-1 marks
* Handling of missing or incomplete data: 3 marks
  + Proper checks and feedback for missing data: 3 marks
  + Inadequate checks or feedback: 0-2 marks

**5. Code Quality and Documentation (Total: 7 Marks)**

* Code structure and organization: 3 marks
  + Clean, organized, and efficient code: 3 marks
  + Disorganized or inefficient code: 0-2 marks
* Commenting and documentation: 4 marks
  + Clear comments explaining logic, well-documented submission: 4 marks
  + Sparse or unclear comments, inadequate documentation: 0-3 marks

**6. Creativity and Additional Features (Total: 2 Marks)**

* Innovative features or improvements beyond requirements: 1 mark
* Enhanced user experience (e.g., improved aesthetics, user guides): 1 mark

**NOTE**: The total marks can be adjusted based on the overall quality and functionality of the application. It's important to emphasize that while the rubric provides a structured approach, there can be some flexibility in grading based on the assessor's discretion and the specific project requirements.