Assignment 5 - File Processing and UI

Student Name: Durvank Deorukhkar

Summary

This report documents the development of a Windows Forms C# application that implements a comprehensive Student Management System. The application meets all specified requirements including file reading capabilities, data display, user authentication, and student data management through an intuitive graphical user interface.

1. Project Overview

1.1 Objective

To create a Windows Forms application that allows users to manage student information with secure authentication, file-based data loading, and interactive data entry capabilities.

1.2 Technologies Used

- **Platform:** .NET Framework
- **Language:** C#
- **UI Framework:** Windows Forms
- **Data Structure:** ArrayList (as specified)
- **IDE:** Microsoft Visual Studio

2. System Architecture

2.1 Application Structure

The application follows a multi-form architecture with the following components:

Core Components:

- **Student.cs** - Data model class
- **Program.cs** - Application entry point
- **Form1.cs/Designer.cs** - Main application form
- **LoginForm.cs/Designer.cs** - Authentication form
- **AddStudentForm.cs/Designer.cs** - Data entry form

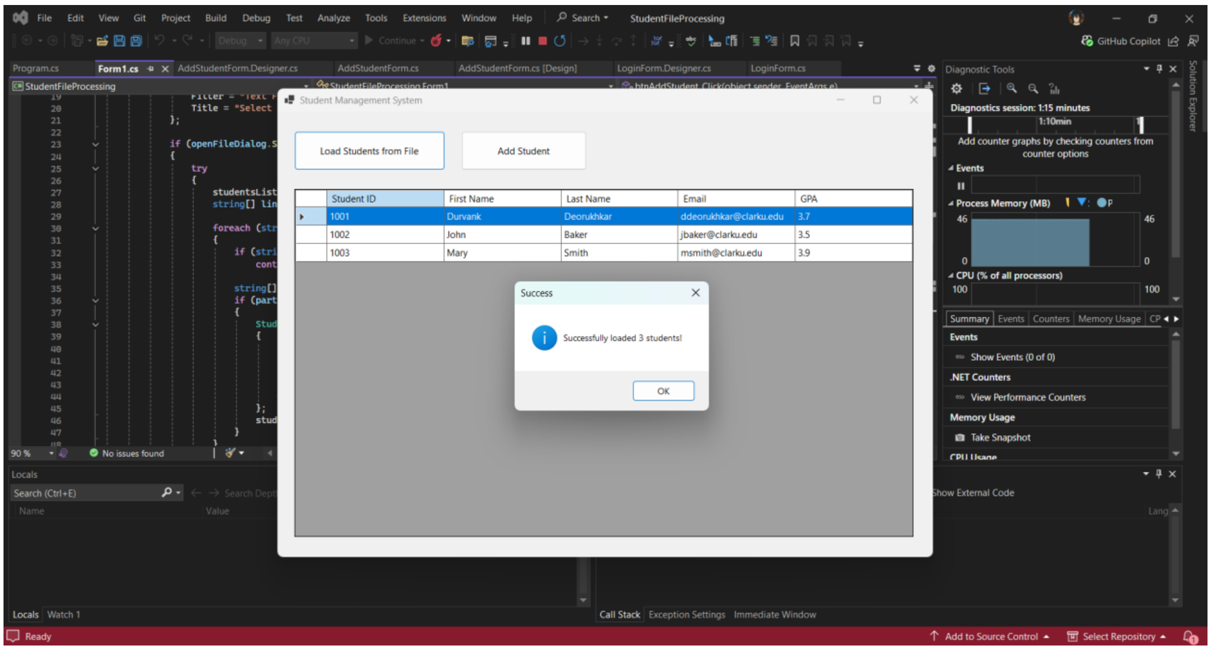
2.2 Data Model

```
public class Student

{

   public int Id { get; set; }

   public string FirstName { get; set; }

   public string LastName { get; set; }

   public string Email { get; set; }

   public double GPA { get; set; }

}
```

The Student class encapsulates all relevant student information with appropriate properties and methods for data manipulation.

3. Implementation of Requirements

3.1 Requirement 1: File Reading and ArrayList Storage



Student ID	First Name	Last Name	Email	GPA
1001	Durvank	Deorukhkar	ddeorukhkar@clarku.edu	3.7
1002	John	Baker	jbaker@clarku.edu	3.5
1003	Mary	Smith	msmith@clarku.edu	3.9

Successfully loaded 3 students!

Implementation:

- A "Load Students from File" button is implemented on the main form
- Uses OpenFileDialog to allow users to select data files
- Supports multiple file formats (.txt, .csv)
- Parses comma-separated values (CSV format)
- Stores Student objects in an ArrayList collection

- Includes error handling for file operations

Code Highlights:

```
private void btnLoadFile_Click(object sender, EventArgs e)
{
   OpenFileDialog openFileDialog = new OpenFileDialog();
   // Parse file and add to studentsList (ArrayList)
}
```

File Format Expected:

```
Id,FirstName,LastName,Email,GPA
1,John,Doe,john.doe@email.com,3.5
```

3.2 Requirement 2: Display Student Information

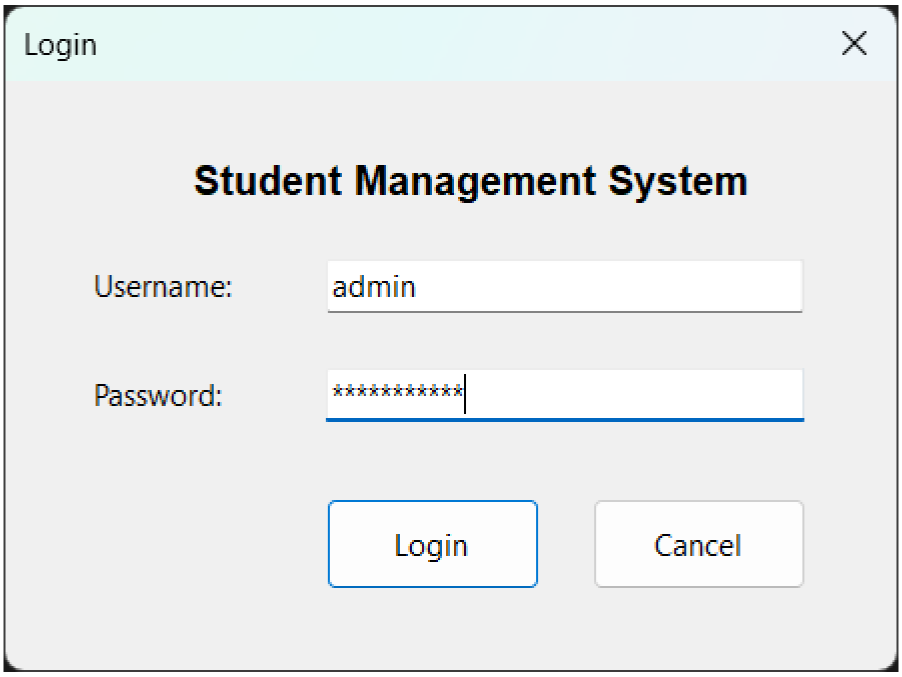
Implementation:

- DataGridView component displays all student information
- Auto-sized columns for optimal viewing
- Read-only grid to prevent accidental modifications
- Full row selection for better user experience
- Automatic refresh after data operations

Features:

- Five columns: Student ID, First Name, Last Name, Email, GPA
- Professional tabular layout
- Scrollable interface for large datasets

3.3 Requirement 3: Login Form with Authentication



Implementation:

- Separate LoginForm class with modal display
- Username and password text fields
- Password masking for security
- Credential validation before main form access
- Login and Cancel buttons

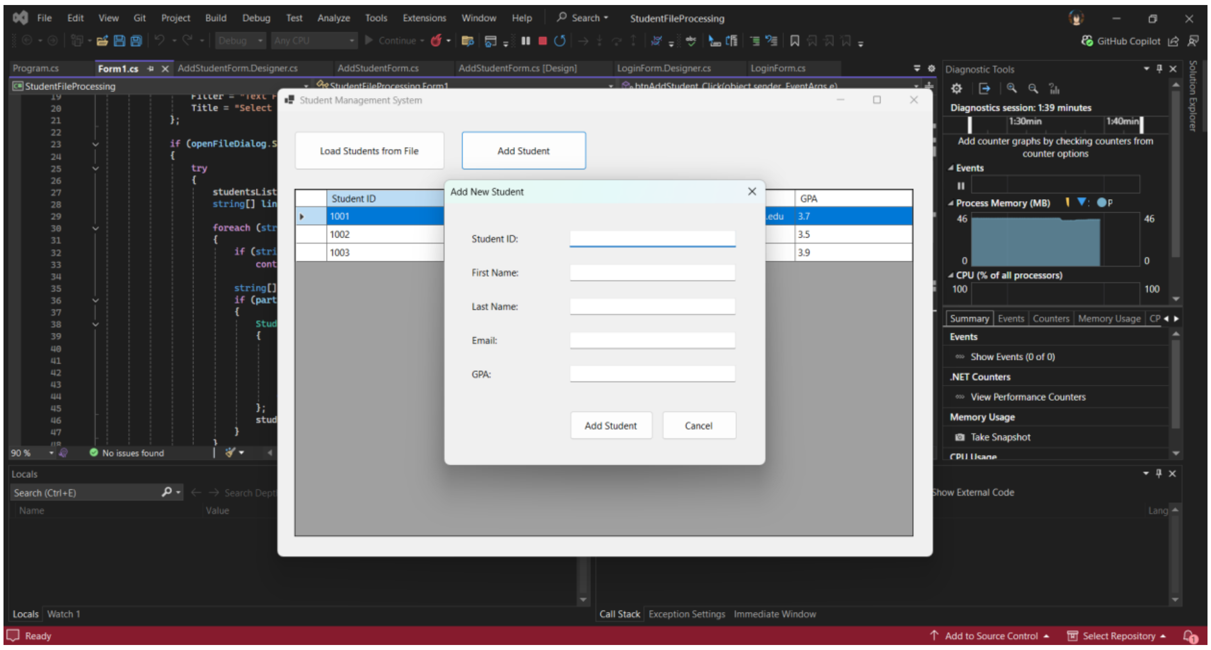
Authentication Details:

- **Default Username:** admin
- **Default Password:** password123
- Modal dialog prevents bypassing authentication
- Application terminates if login fails or is cancelled

Security Features:

- Password field uses masking (*)
- Clear error messages for failed attempts
- Password field clears after failed login

3.4 Requirement 4: Add Student Form



```
Filter = "Text F
Title = "Select
};

if (openFileDialog.S
{
    try
    {
        studentsList
        string[] lin

        foreach (str
        {
            if (stri
                cont

            string[]
            if (part
            {
                Stud
                {



                };
                stud
            }
```

	Student ID	GPA
▸	1001	3.7
	1002	3.5
	1003	3.9

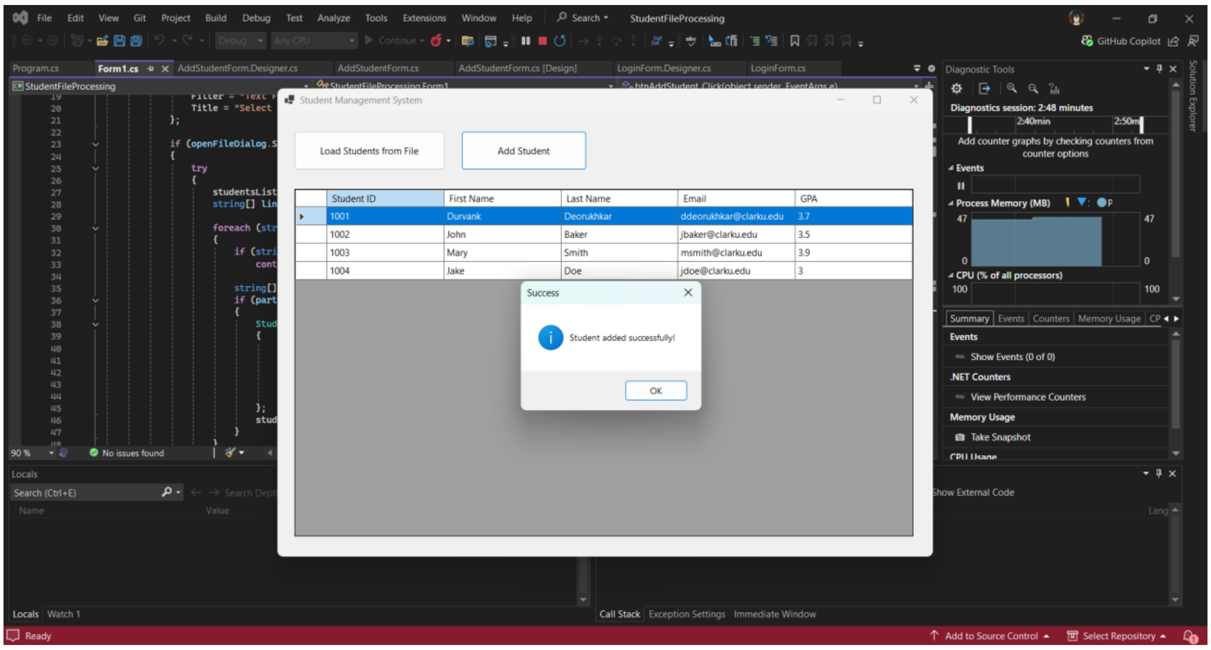
Implementation:

- Dedicated AddStudentForm for data entry
- Visual design with labels and text boxes for all fields:
    - Student ID
    - First Name
    - Last Name
    - Email
    - GPA
- Modal display for focused user interaction
- Add and Cancel buttons for user control

Validation Features:

- All fields required (no empty submissions)
- Student ID must be a valid integer
- GPA must be numeric and between 0.0 and 4.0
- Clear error messages for validation failures

3.5 Requirement 5: Add Student to Collection



	Student ID	First Name	Last Name	Email	GPA
▸	1001	Durvank	Deorukhkar	ddeorukhkar@clarku.edu	3.7
	1002	John	Baker	jbaker@clarku.edu	3.5
	1003	Mary	Smith	msmith@clarku.edu	3.9
	1004	Jake	Doe	jdoe@clarku.edu	3

Student added successfully!

Implementation:

- New students added to ArrayList collection
- Immediate DataGridView refresh after addition
- Success confirmation message
- No page reload required
- Data persists in collection during session

Workflow:

1. User clicks "Add Student" button
2. AddStudentForm opens modally
3. User enters student information
4. Validation occurs on submission
5. Valid data adds Student to ArrayList
6. DataGridView automatically updates
7. Confirmation message displays

4. Key Features

4.1 Modal Dialog Implementation

All secondary forms (LoginForm and AddStudentForm) use ShowDialog() method to ensure modal behavior, preventing users from interacting with other forms until the current dialog is closed.

4.2 Data Persistence

- Data stored in ArrayList during application session
- File loading preserves data integrity
- Manual additions integrate seamlessly with loaded data

4.3 Error Handling

- Try-catch blocks for file operations
- Input validation with user-friendly messages
- Graceful handling of invalid data formats

4.4 User Experience Enhancements

- Confirmation messages for successful operations
- Clear error messages for failures
- Intuitive button labels and form titles
- Professional visual design

5. Testing

5.1 Test Scenarios

Login Authentication:

- Valid credentials allow access
- Invalid credentials show error
- Cancel button closes application
- Enter key submits login

File Loading:

- Successfully loads properly formatted files
- Handles empty files gracefully
- Displays error for corrupted data
- Updates DataGridView correctly

Add Student:

- Valid data adds successfully

  Empty fields trigger validation
- Invalid ID format rejected
- GPA validation (0.0-4.0 range)
- Cancel button closes without adding
- DataGridView updates immediately

5.2 Sample Test Data

1,John,Doe,john.doe@email.com,3.5

2,Jane,Smith,jane.smith@email.com,3.8

3,Bob,Johnson,bob.j@email.com,3.2

4,Alice,Williams,alice.w@email.com,3.9

5,Charlie,Brown,charlie.b@email.com,3.1

6. Usage Instructions

1. **Login:** Enter username "admin" and password "password123"
2. **Load Data:** Click "Load Students from File" and select a CSV file
3. **Add Student:** Click "Add Student" and fill in the form
4. **View Data:** All students display in the DataGridView automatically

7. Appendix

7.1 Sample Data File Format

Create a text file named students.txt with the following format:

1,John,Doe,john.doe@email.com,3.5

2,Jane,Smith,jane.smith@email.com,3.8

3,Bob,Johnson,bob.j@email.com,3.2

7.2 Login Credentials

- **Username:** admin
- **Password:** password123