**Assessment Details**

**Development Time**

Both groups were given a 2-hour limit to complete as much of the assignment as possible. This limit is based on the expectation that a typical weekly programming assignment should be manageable within this timeframe. Given the complexity of the assignment, it was acknowledged that completing it within 2 hours would be challenging, but feasible with a team of three students.

**Group 1 (Traditional Software Engineering)**

* Partially complete.

**Group 2 (Software Orchestration)**

* Successfully completed a functional application.

**Degree of Completion by Class**

Both groups completed the Data Definition Language (DDL) for their respective databases successfully. The degree of completion for the application classes is as follows:

|  |  |  |
| --- | --- | --- |
| Class | Group 1 (Software Engineering) | Group 2 (Software Orchestration) |
| AddMember | Partially complete | Complete |
| UpdateMember | Not started | Complete |
| DeleteMember | Not started | Complete |
| DatabaseUtil | Partially complete | Complete |
| MainMenu | Partially complete | Complete |
| Report | Not started | Complete |
| DDL | Complete | Complete |

**Code Quality**

**Group 1 (Traditional Software Engineering)**

* Code quality was mixed, with readable code but some areas lacking in adherence to coding standards.

**Group 2 (Software Orchestration)**

* Code quality was high, with well-structured and readable code, following best practices.

**Number of Errors**

**Group 1 (Traditional Software Engineering)**

* Encountered and resolved several syntax and logic errors during the development process.

**Group 2 (Software Orchestration)**

* Encountered fewer errors, with the majority being syntax errors quickly resolved with AI assistance.

**Functionality**

**Group 1 (Traditional Software Engineering)**

* Partial functionality due to incomplete class implementation.

**Group 2 (Software Orchestration)**

* Full functionality as per the given specifications.

**Documentation Quality**

**Group 1 (Traditional Software Engineering)**

* Documentation was incomplete, with some UML diagrams missing and narrative descriptions lacking detail.

**Group 2 (Software Orchestration)**

* Comprehensive documentation, including complete UML diagrams and detailed narrative descriptions.

**Summary Table**

|  |  |  |
| --- | --- | --- |
| Metric | Group 1 (Software Engineering) | Group 2 (Software Orchestration) |
| Development Time | Partially complete | Complete |
| AddMember Class | Partially complete | Complete |
| UpdateMember Class | Not started | Complete |
| DeleteMember Class | Not started | Complete |
| DatabaseUtil Class | Partially complete | Complete |
| MainMenu Class | Partially complete | Complete |
| Report Class | Not started | Complete |
| DDL | Complete | Complete |
| Code Quality | Mixed | High |
| Number of Errors | Several syntax and logic errors | Fewer errors, mostly syntax |
| Functionality | Partial | Full |
| Documentation | Incomplete | Comprehensive |

This table summarizes the results of the evaluation, providing a clear comparison of the two approaches based on development time, degree of completion, code quality, number of errors, functionality, and documentation quality.

**Group 1: MainMenu Class Assessment**

**Code Analysis:**

* The MainMenu class displays the menu successfully.
* Only the AddMember button is functional and opens the AddMember form.
* The Update, Delete, and Report buttons are present but lack functional implementations.

**Estimated Degree of Completion:**

* Functional UI: 100%
* AddMember Functionality: 100%
* UpdateMember Functionality: 0%
* DeleteMember Functionality: 0%
* Report Functionality: 0%

**Completion Percentage:** The overall completion percentage can be estimated by considering the proportion of fully implemented buttons:

* There are four buttons, with only one (AddMember) fully functional.

Completion Percentage=(Number of Functional ButtonsTotal Buttons)×100\text{Completion Percentage} = \left( \frac{\text{Number of Functional Buttons}}{\text{Total Buttons}} \right) \times 100Completion Percentage=(Total ButtonsNumber of Functional Buttons​)×100

Completion Percentage=(14)×100=25%\text{Completion Percentage} = \left( \frac{1}{4} \right) \times 100 = 25\%Completion Percentage=(41​)×100=25%

Thus, the estimated degree of completion for Group 1's MainMenu class is **25%**.

**Group 2: MainMenu Class Assessment**

**Code Analysis:**

* The MainMenu class is fully functional.
* All four buttons (AddMember, UpdateMember, DeleteMember, Report) are implemented and open their respective windows.
* The implementation includes appropriate event handling for each button, ensuring that the respective forms are displayed when clicked.

**Estimated Degree of Completion:**

* Functional UI: 100%
* AddMember Functionality: 100%
* UpdateMember Functionality: 100%
* DeleteMember Functionality: 100%
* Report Functionality: 100%

**Completion Percentage:** Given that all buttons are functional and the MainMenu class performs as required:

Completion Percentage=100%\text{Completion Percentage} = 100\%Completion Percentage=100%

Thus, the estimated degree of completion for Group 2's MainMenu class is **100%**.

**Summary of MainMenu Class Completion**

|  |  |  |
| --- | --- | --- |
| Metric | Group 1 (Software Engineering) | Group 2 (Software Orchestration) |
| Functional UI | 100% | 100% |
| AddMember Functionality | 100% | 100% |
| UpdateMember Functionality | 0% | 100% |
| DeleteMember Functionality | 0% | 100% |
| Report Functionality | 0% | 100% |
| Overall Completion | 25% | 100% |

.

**Group 1: AddMember Class Assessment**

**Code Analysis:**

* **UI Initialization: The UI components are initialized, but there are issues with the labels being mixed in the text fields.**
* **Database Connection: The addMember method handles the database insertion logic correctly, including setting up the connection and preparing the SQL statement.**
* **Functionality:**
  + **The Submit button is present, but no action listener is defined, so it does not trigger the addMember method.**
  + **Labels for text fields are placeholders within the text fields, which is not ideal for usability.**

**Estimated Degree of Completion:**

* **UI Initialization: 100%**
* **Database Connection Logic: 100%**
* **Field Labels and Usability: 50% (Labels are in text fields instead of being separate, reducing clarity)**
* **Submit Button Functionality: 0% (No action listener defined)**

**Completion Percentage:**

* **UI Components: Fully initialized (100%).**
* **Functionality (Submit): Not implemented (0%).**
* **Overall Usability: Partially complete due to mixed labels (50%).**

**Considering these aspects:**

**Thus, the estimated degree of completion for Group 1's AddMember class is 62.5%.**

**Summary of AddMember Class Completion**

**Group 1 (Software Engineering)**

|  |  |
| --- | --- |
| Metric | Completion Percentage |
| UI Initialization | **100%** |
| Database Connection Logic | **100%** |
| Field Labels and Usability | **50%** |
| Submit Button Functionality | **0%** |
| Overall Completion | **62.5%** |

**Group 2: AddMember Class Assessment**

**Code Analysis:**

* **UI Initialization**: The UI components are correctly initialized, with appropriate labels and input fields, including formatted fields for Member ID and dates.
* **Database Connection**: The addMember method handles the database insertion logic using the DatabaseUtil class.
* **Functionality**:
  + The Submit button is fully functional and triggers the addMember method, including validation checks for all input fields.
  + The Cancel button is implemented and correctly closes the window when clicked.
  + Validation checks ensure that all required fields are filled and that the input is in the correct format before submission.

**Estimated Degree of Completion:**

* **UI Initialization**: 100%
* **Database Connection Logic**: 100%
* **Field Labels and Usability**: 100%
* **Submit Button Functionality**: 100%

**Completion Percentage:** All aspects of the AddMember class are fully implemented and functional.

Completion Percentage=100%\text{Completion Percentage} = 100\%Completion Percentage=100%

Thus, the estimated degree of completion for Group 2's AddMember class is **100%**.

**Summary of AddMember Class Completion**

**Group 1 (Software Engineering)**

|  |  |
| --- | --- |
| Metric | Completion Percentage |
| UI Initialization | 100% |
| Database Connection Logic | 100% |
| Field Labels and Usability | 50% |
| Submit Button Functionality | 0% |
| Overall Completion | **62.5%** |

**Group 2 (Software Orchestration)**

|  |  |
| --- | --- |
| Metric | Completion Percentage |
| UI Initialization | 100% |
| Database Connection Logic | 100% |
| Field Labels and Usability | 100% |
| Submit Button Functionality | 100% |
| Overall Completion | **100%** |

**Group 1: UpdateMember Class Assessment**

**Code Analysis:**

* **UI Initialization**: The UI components are not implemented.
* **Database Connection**: The updateMember method contains errors and is incomplete.
  + The SQL statement syntax is incorrect.
  + There is no UI to capture user input for member details.
  + No logic to handle the update operation in a GUI form.

**Estimated Degree of Completion:**

* **UI Initialization**: 0%
* **Database Connection Logic**: 50% (The SQL statement is present but incorrect; logic is partially implemented)
* **Field Labels and Usability**: 0%
* **Update Button Functionality**: 0%

**Completion Percentage:** Given that the UI components are missing, and the database logic is incomplete and incorrect, the overall completion is minimal.

Completion Percentage=(0+50+0+04)=12.5%\text{Completion Percentage} = \left( \frac{0 + 50 + 0 + 0}{4} \right) = 12.5\%Completion Percentage=(40+50+0+0​)=12.5%

Thus, the estimated degree of completion for Group 1's UpdateMember class is **12.5%**.

**Group 2: UpdateMember Class Assessment**

**Code Analysis:**

* **UI Initialization**: The UI components are correctly initialized, with appropriate labels and input fields.
* **Database Connection**: The updateMember method is correctly implemented using the DatabaseUtil class.
* **Functionality**:
  + The Search button is fully functional, retrieving member details.
  + The Update button is fully functional, updating member details in the database.
  + The Cancel button is implemented and correctly closes the window when clicked.
  + Validation checks ensure that all required fields are filled and that the input is in the correct format before submission.

**Estimated Degree of Completion:**

* **UI Initialization**: 100%
* **Database Connection Logic**: 100%
* **Field Labels and Usability**: 100%
* **Update Button Functionality**: 100%

**Completion Percentage:** All aspects of the UpdateMember class are fully implemented and functional.

Completion Percentage=100%\text{Completion Percentage} = 100\%Completion Percentage=100%

Thus, the estimated degree of completion for Group 2's UpdateMember class is **100%**.

**Summary of UpdateMember Class Completion**

**Group 1 (Software Engineering)**

|  |  |
| --- | --- |
| Metric | Completion Percentage |
| UI Initialization | 0% |
| Database Connection Logic | 50% |
| Field Labels and Usability | 0% |
| Update Button Functionality | 0% |
| Overall Completion | **12.5%** |

**Group 2 (Software Orchestration)**

|  |  |
| --- | --- |
| Metric | Completion Percentage |
| UI Initialization | 100% |
| Database Connection Logic | 100% |
| Field Labels and Usability | 100% |
| Update Button Functionality | 100% |
| Overall Completion | **100%** |

**Group 1: DeleteMember Class Assessment**

**Code Analysis:**

* **UI Initialization**: The UI components are initialized but minimal, with only the searchLabel, searchField, and searchButton.
* **Database Connection**: The deleteMember method contains an incorrect SQL statement and does not perform deletion correctly.
  + The SQL syntax is incorrect: "DELETE FROM members WHERE id = (id) VALUES(?)".
  + No logic for user confirmation or feedback after deletion.

**Estimated Degree of Completion:**

* **UI Initialization**: 50% (UI is present but minimal and lacks feedback and confirmation dialogs)
* **Database Connection Logic**: 50% (SQL statement is incorrect; partial implementation)
* **Field Labels and Usability**: 50% (Basic functionality with no confirmation or error handling)
* **Delete Button Functionality**: 50% (Basic implementation but incorrect logic)

**Completion Percentage:** Given the partial implementation of both UI and functionality, the overall completion is low.

Completion Percentage=(50+50+50+504)=50%\text{Completion Percentage} = \left( \frac{50 + 50 + 50 + 50}{4} \right) = 50\%Completion Percentage=(450+50+50+50​)=50%

Thus, the estimated degree of completion for Group 1's DeleteMember class is **50%**.

**Group 2: DeleteMember Class Assessment**

**Code Analysis:**

* **UI Initialization**: The UI components are correctly initialized with labels, text fields, and buttons.
* **Database Connection**: The deleteMember method is correctly implemented using the DatabaseUtil class.
* **Functionality**:
  + The Delete button is fully functional, including checking if the member exists, confirming deletion, and providing feedback.
  + The Cancel button is implemented and correctly closes the window when clicked.
  + Validation checks ensure the member ID is valid before attempting deletion.

**Estimated Degree of Completion:**

* **UI Initialization**: 100%
* **Database Connection Logic**: 100%
* **Field Labels and Usability**: 100%
* **Delete Button Functionality**: 100%

**Completion Percentage:** All aspects of the DeleteMember class are fully implemented and functional.

Completion Percentage=100%\text{Completion Percentage} = 100\%Completion Percentage=100%

Thus, the estimated degree of completion for Group 2's DeleteMember class is **100%**.

**Summary of DeleteMember Class Completion**

**Group 1 (Software Engineering)**

|  |  |
| --- | --- |
| Metric | Completion Percentage |
| UI Initialization | 50% |
| Database Connection Logic | 50% |
| Field Labels and Usability | 50% |
| Delete Button Functionality | 50% |
| Overall Completion | **50%** |

**Group 2 (Software Orchestration)**

|  |  |
| --- | --- |
| Metric | Completion Percentage |
| UI Initialization | 100% |
| Database Connection Logic | 100% |
| Field Labels and Usability | 100% |
| Delete Button Functionality | 100% |
| Overall Completion | **100%** |

**Group 1: DatabaseUtil Class Assessment**

**Code Analysis:**

* The class DatabaseUtil is declared but completely empty, indicating no functionality has been implemented.

**Estimated Degree of Completion:**

* **Database Connection Logic**: 0% (No implementation)
* **CRUD Operations**: 0% (No methods implemented)
* **Report Generation**: 0% (No methods implemented)

**Completion Percentage:** Given the lack of any implemented methods, the overall completion is:

Thus, the estimated degree of completion for Group 1's DatabaseUtil class is **0%**.

**Group 2: DatabaseUtil Class Assessment**

**Code Analysis:**

* **Database Connection Logic**: The class includes a method to establish a connection to the database.
* **CRUD Operations**: Fully implemented methods for adding, retrieving, updating, and deleting members.
* **Report Generation**: Implemented methods for generating status, all members, and level reports.

**Estimated Degree of Completion:**

* **Database Connection Logic**: 100%
* **CRUD Operations**: 100%
* **Report Generation**: 100%

**Completion Percentage:** All aspects of the DatabaseUtil class are fully implemented and functional.

Completion Percentage=100%\{Completion Percentage} = 100\%Completion Percentage=100%

Thus, the estimated degree of completion for Group 2's DatabaseUtil class is **100%**.

**Summary of DatabaseUtil Class Completion**

**Group 1 (Software Engineering)**

|  |  |
| --- | --- |
| Metric | Completion Percentage |
| Database Connection Logic | 0% |
| CRUD Operations | 0% |
| Report Generation | 0% |
| Overall Completion | **0%** |

**Group 2 (Software Orchestration)**

|  |  |
| --- | --- |
| Metric | Completion Percentage |
| Database Connection Logic | 100% |
| CRUD Operations | 100% |
| Report Generation | 100% |
| Overall Completion | **100%** |

**Group 1: Report Class Assessment**

**Code Analysis:**

* **UI Initialization**: The UI components are partially initialized but contain several errors and missing elements.
  + The reportTypeDropdown and buttons are declared but not fully implemented.
  + The layout and title labels contain syntax errors and are incorrectly implemented.

**Estimated Degree of Completion:**

* **UI Initialization**: 30% (Partially initialized with errors)
* **Database Connection Logic**: 0% (No connection logic implemented)
* **Field Labels and Usability**: 20% (Basic elements present but not functional or correct)
* **Report Generation Functionality**: 0% (No functionality implemented)

**Completion Percentage:** Given the partial initialization and presence of errors, the overall completion is low.

Thus, the estimated degree of completion for Group 1's Report class is **12.5%**.

**Group 2: Report Class Assessment**

**Code Analysis:**

* **UI Initialization**: The UI components are correctly initialized with labels, dropdown, buttons, and a report display area.
* **Database Connection**: The class uses the DatabaseUtil class for retrieving data.
* **Functionality**:
  + The Generate button is fully functional, allowing the generation of various reports based on the selected option.
  + The Cancel button is implemented and correctly closes the window when clicked.
  + The displayReport method formats and displays the report data in a text area.

**Estimated Degree of Completion:**

* **UI Initialization**: 100%
* **Database Connection Logic**: 100%
* **Field Labels and Usability**: 100%
* **Report Generation Functionality**: 100%

**Completion Percentage:** All aspects of the Report class are fully implemented and functional.

Completion Percentage=100%\text{Completion Percentage} = 100\%Completion Percentage=100%

Thus, the estimated degree of completion for Group 2's Report class is **100%**.

**Summary of Report Class Completion**

**Group 1 (Software Engineering)**

|  |  |
| --- | --- |
| Metric | Completion Percentage |
| UI Initialization | 30% |
| Database Connection Logic | 0% |
| Field Labels and Usability | 20% |
| Report Generation Functionality | 0% |
| Overall Completion | **12.5%** |

**Group 2 (Software Orchestration)**

|  |  |
| --- | --- |
| Metric | Completion Percentage |
| UI Initialization | 100% |
| Database Connection Logic | 100% |
| Field Labels and Usability | 100% |
| Report Generation Functionality | 100% |
| Overall Completion | **100%** |

**Final Summary Table of Class Completion**

|  |  |  |
| --- | --- | --- |
| Class | Group 1 (Software Engineering) | Group 2 (Software Orchestration) |
| MainMenu | 25% | 100% |
| AddMember | 62.5% | 100% |
| UpdateMember | 12.5% | 100% |
| DeleteMember | 50% | 100% |
| DatabaseUtil | 0% | 100% |
| Report | 12.5% | 100% |

This summary table provides a clear comparison of the completion percentages for each class between the two groups.

**Assessment of Code Quality and Readability**

Using the provided code, we will assess the quality and readability of each class for both groups. The assessment criteria include:

1. **Code Structure**: Logical organization of code.
2. **Readability**: Clarity and simplicity of the code, use of meaningful variable names, comments, and adherence to coding standards.
3. **Error Handling**: Implementation of error handling mechanisms.
4. **Functionality**: Correctness and completeness of the implemented functionality.

**Code Quality and Readability Summary**

|  |  |  |
| --- | --- | --- |
| Class | Group 1 (Software Engineering) | Group 2 (Software Orchestration) |
| MainMenu | **60%** | **95%** |
| AddMember | **65%** | **90%** |
| UpdateMember | **30%** | **90%** |
| DeleteMember | **50%** | **95%** |
| DatabaseUtil | **0%** | **100%** |
| Report | **20%** | **95%** |

**Detailed Analysis**

**MainMenu Class:**

* **Group 1**: 60%
  + **Code Structure**: Reasonably structured, but lacks functionality for most buttons.
  + **Readability**: Good variable names, minimal comments.
  + **Error Handling**: Basic, lacks comprehensive error handling.
  + **Functionality**: Partially implemented.
* **Group 2**: 95%
  + **Code Structure**: Well-structured, all functionalities are implemented.
  + **Readability**: Excellent use of variable names, comments, and adherence to standards.
  + **Error Handling**: Comprehensive error handling.
  + **Functionality**: Fully implemented.

**AddMember Class:**

* **Group 1**: 65%
  + **Code Structure**: Partially structured, some mixed labels in text fields.
  + **Readability**: Good but lacks comments.
  + **Error Handling**: Minimal.
  + **Functionality**: Partially implemented.
* **Group 2**: 90%
  + **Code Structure**: Well-structured, clear separation of concerns.
  + **Readability**: Excellent, with meaningful variable names and comments.
  + **Error Handling**: Good error handling and validation.
  + **Functionality**: Fully implemented.

**UpdateMember Class:**

* **Group 1**: 30%
  + **Code Structure**: Poorly structured, incomplete.
  + **Readability**: Minimal readability, lacks comments and clear logic.
  + **Error Handling**: Absent.
  + **Functionality**: Incomplete.
* **Group 2**: 90%
  + **Code Structure**: Well-structured, clear and concise.
  + **Readability**: Excellent, with meaningful variable names and comments.
  + **Error Handling**: Good error handling and validation.
  + **Functionality**: Fully implemented.

**DeleteMember Class:**

* **Group 1**: 50%
  + **Code Structure**: Partially structured, with logical errors in SQL.
  + **Readability**: Good, but minimal comments.
  + **Error Handling**: Basic error handling.
  + **Functionality**: Partially implemented.
* **Group 2**: 95%
  + **Code Structure**: Well-structured, clear logic.
  + **Readability**: Excellent, with comments and clear variable names.
  + **Error Handling**: Comprehensive.
  + **Functionality**: Fully implemented.

**DatabaseUtil Class:**

* **Group 1**: 0%
  + **Code Structure**: Absent.
  + **Readability**: Absent.
  + **Error Handling**: Absent.
  + **Functionality**: Absent.
* **Group 2**: 100%
  + **Code Structure**: Well-structured, modular.
  + **Readability**: Excellent, with meaningful variable names and comments.
  + **Error Handling**: Comprehensive error handling.
  + **Functionality**: Fully implemented.

**Report Class:**

* **Group 1**: 20%
  + **Code Structure**: Poorly structured, minimal implementation.
  + **Readability**: Minimal readability, several errors.
  + **Error Handling**: Absent.
  + **Functionality**: Incomplete.
* **Group 2**: 95%
  + **Code Structure**: Well-structured, clear separation of concerns.
  + **Readability**: Excellent, with comments and clear variable names.
  + **Error Handling**: Comprehensive.
  + **Functionality**: Fully implemented.

This summary table provides a comparative assessment of the code quality and readability for each class between the two groups.

**Assessment of Errors**

Using the provided code, we will assess the errors in each class for both groups. The assessment criteria include:

1. **Compilation Errors**: Errors that prevent the code from compiling.
2. **Runtime Errors**: Errors that occur during the execution of the program.
3. **Logic Errors**: Errors in the logic that produce incorrect results or unexpected behavior.
4. **Syntax Errors**: Mistakes in the code syntax that do not follow the language rules.

**Error Assessment Summary**

|  |  |  |
| --- | --- | --- |
| Class | Group 1 (Software Engineering) | Group 2 (Software Orchestration) |
| MainMenu | **20%** | **5%** |
| AddMember | **30%** | **5%** |
| UpdateMember | **50%** | **5%** |
| DeleteMember | **40%** | **5%** |
| DatabaseUtil | **100%** | **5%** |
| Report | **80%** | **5%** |

**Detailed Analysis**

**MainMenu Class:**

* **Group 1**: 20% (Significant errors)
  + **Compilation Errors**: None.
  + **Runtime Errors**: None, but functional buttons are missing.
  + **Logic Errors**: Missing logic for most buttons.
  + **Syntax Errors**: None.
* **Group 2**: 5% (Minor errors)
  + **Compilation Errors**: None.
  + **Runtime Errors**: None.
  + **Logic Errors**: None.
  + **Syntax Errors**: None.

**AddMember Class:**

* **Group 1**: 30% (Moderate errors)
  + **Compilation Errors**: None.
  + **Runtime Errors**: None.
  + **Logic Errors**: Mixed labels and missing action listener.
  + **Syntax Errors**: None.
* **Group 2**: 5% (Minor errors)
  + **Compilation Errors**: None.
  + **Runtime Errors**: None.
  + **Logic Errors**: None.
  + **Syntax Errors**: None.

**UpdateMember Class:**

* **Group 1**: 50% (Significant errors)
  + **Compilation Errors**: None.
  + **Runtime Errors**: Potential errors due to incomplete logic.
  + **Logic Errors**: Incorrect SQL and incomplete implementation.
  + **Syntax Errors**: None.
* **Group 2**: 5% (Minor errors)
  + **Compilation Errors**: None.
  + **Runtime Errors**: None.
  + **Logic Errors**: None.
  + **Syntax Errors**: None.

**DeleteMember Class:**

* **Group 1**: 40% (Moderate errors)
  + **Compilation Errors**: None.
  + **Runtime Errors**: Potential errors due to incorrect SQL.
  + **Logic Errors**: Incorrect SQL syntax.
  + **Syntax Errors**: None.
* **Group 2**: 5% (Minor errors)
  + **Compilation Errors**: None.
  + **Runtime Errors**: None.
  + **Logic Errors**: None.
  + **Syntax Errors**: None.

**DatabaseUtil Class:**

* **Group 1**: 100% (Critical errors)
  + **Compilation Errors**: None (Empty class).
  + **Runtime Errors**: Not applicable (Empty class).
  + **Logic Errors**: Not applicable (Empty class).
  + **Syntax Errors**: Not applicable (Empty class).
* **Group 2**: 5% (Minor errors)
  + **Compilation Errors**: None.
  + **Runtime Errors**: None.
  + **Logic Errors**: None.
  + **Syntax Errors**: None.

**Report Class:**

* **Group 1**: 80% (Critical errors)
  + **Compilation Errors**: Yes, due to syntax errors and missing elements.
  + **Runtime Errors**: Not applicable due to compilation errors.
  + **Logic Errors**: Not applicable due to compilation errors.
  + **Syntax Errors**: Yes.
* **Group 2**: 5% (Minor errors)
  + **Compilation Errors**: None.
  + **Runtime Errors**: None.
  + **Logic Errors**: None.
  + **Syntax Errors**: None.

**Final Summary Table of Errors**

|  |  |  |
| --- | --- | --- |
| Class | Group 1 (Software Engineering) | Group 2 (Software Orchestration) |
| MainMenu | 20% | 5% |
| AddMember | 30% | 5% |
| UpdateMember | 50% | 5% |
| DeleteMember | 40% | 5% |
| DatabaseUtil | 100% | 5% |
| Report | 80% | 5% |

This summary table provides a comparative assessment of the error prevalence for each class between the two groups.

**Attribution of Minor Errors**

The minor errors in the Group 2 (Software Orchestration) code can largely be attributed to the following factors:

1. **GUI Design and Usability**:
   * The GUI design might be functional but lacks refinement and user-friendly elements.
   * Components could be better organized to enhance the user experience.
   * Default layouts and component sizes might not be optimal, leading to a clumsy or less intuitive interface.
2. **Error Handling**:
   * While the code includes basic error handling, there may be areas where additional validation or more detailed error messages could improve robustness.
   * Some edge cases might not be thoroughly handled, leading to potential runtime issues under specific conditions.
3. **Consistency and Standardization**:
   * Slight inconsistencies in how UI components are initialized and handled can contribute to a less polished feel.
   * Adhering strictly to design standards and conventions could improve the overall quality and reduce minor errors.
4. **Code Readability and Maintenance**:
   * Although readability is generally good, there might be minor issues such as inconsistent commenting or slightly unclear variable names.
   * Ensuring that code is consistently well-documented and easy to understand can help mitigate small errors.

**Recommendations for Improvement**

1. **Enhancing GUI Design**:
   * Use layout managers more effectively to create a cleaner and more intuitive UI.
   * Consider user experience principles to improve the layout and interaction flow.
2. **Improving Error Handling**:
   * Add more comprehensive validation and error messages.
   * Test the application under various scenarios to catch edge cases and handle them appropriately.
3. **Ensuring Consistency**:
   * Standardize the initialization and usage of UI components.
   * Adhere to coding standards and conventions throughout the codebase.
4. **Refining Code Readability**:
   * Increase the use of meaningful comments and maintain consistent commenting practices.
   * Use clear and descriptive variable names to enhance readability and maintainability.

By addressing these areas, the minor errors can be reduced, resulting in a more polished and user-friendly application.

**Functionality Assessment Summary**

Based on the analysis provided, we can rate the functionality of each class for both groups:

|  |  |  |
| --- | --- | --- |
| Class | Group 1 (Software Engineering) | Group 2 (Software Orchestration) |
| MainMenu | 25% (Partially functional) | 100% (Fully functional) |
| AddMember | 62.5% (Partially functional) | 100% (Fully functional) |
| UpdateMember | 12.5% (Mostly non-functional) | 100% (Fully functional) |
| DeleteMember | 50% (Partially functional) | 100% (Fully functional) |
| DatabaseUtil | 0% (Non-functional) | 100% (Fully functional) |
| Report | 12.5% (Mostly non-functional) | 100% (Fully functional) |

**Group 1 (Software Engineering)**:

* **MainMenu**: Partially functional, with only the AddMember button implemented.
* **AddMember**: Partially functional, but lacks complete functionality and usability.
* **UpdateMember**: Mostly non-functional, with significant issues and missing logic.
* **DeleteMember**: Partially functional, but contains logic errors.
* **DatabaseUtil**: Non-functional, as it is not implemented.
* **Report**: Mostly non-functional, with syntax errors and incomplete implementation.

**Group 2 (Software Orchestration)**:

* **MainMenu**: Fully functional, with all buttons and features working as intended.
* **AddMember**: Fully functional, with comprehensive implementation and validation.
* **UpdateMember**: Fully functional, with proper logic and error handling.
* **DeleteMember**: Fully functional, with user confirmation and correct deletion logic.
* **DatabaseUtil**: Fully functional, supporting all CRUD operations and report generation.
* **Report**: Fully functional, with all reports generating correctly and displaying results.

**Conclusion**

Both completion percentage and functionality assessment are crucial for evaluating a project. While completion percentage provides a snapshot of progress, functionality assessment ensures that the implemented features are working correctly and effectively. Combining these metrics offers a comprehensive understanding of the project's status and quality.

**Group 1 Documentation Assessment**

**Summary**

**Authors**: Noah Caldwell, Thomas Simmons, Ahmed Wahab

**Professors**: Dr. Elarde, Dr. Hasan, Professor Bruster

**Introduction**:

* The assignment was to code a complete club membership application in Java with built-in database functionality. The program consisted of 6 classes: MainMenu, AddMember, UpdateMember, DeleteMember, Report, and DatabaseUtil.
* The application used the Java Swing UI framework for the GUI and MySQL (included in XAMPP) as the backend database.
* The Oracle JDBC driver was used to connect the Java application to the MySQL database.
* Visual Studio Code was chosen as the IDE for development due to its integration with GitHub for version management.

**Data Collection**:

1. 30 minutes setup, 2 hours development.
2. Five classes were developed (three mostly complete, two partially complete).
3. Not applicable (NA) for a specific data point.
4. Encountered about five syntax errors, no logical errors were encountered due to limited testing.
5. The team found that 2 hours was insufficient to develop a fully functioning Java application. They noted that using tools like ChatGPT could have significantly sped up the process, particularly in generating boilerplate code.
6. The application was not finished. The connection between the database and the Java app was incomplete, and the GUIs for UpdateMember and Report were unfinished, limiting the ability to perform comprehensive testing.

**Post-Development**:

1. The team understood the application requirements well but was unsuccessful in implementing them within the time frame. They believe that with more time, they would be able to finish the application.
2. NA

**Github Repository**:

* [Group1ClubMembership](https://github.com/Tsimmsz/Group1ClubMembership)
* [Group1ClubMembership](https://github.com/Tsimmsz/Group1ClubMembership)

**UML Diagram and Table Design**: Included in the documentation.

**Analysis of Documentation Quality**

1. **Clarity and Organization**:
   * The documentation is clearly written and well-organized, with sections logically following one another.
   * Introduction provides a good overview of the assignment and the tools used.
2. **Detail and Completeness**:
   * The documentation outlines the development process, challenges faced, and the outcome of the experiment.
   * Provides URLs to the GitHub repository for further details and code access.
   * Includes a UML diagram and table design, adding value by visualizing the class structure and database schema.
3. **Accuracy and Insight**:
   * The team accurately describes the setup and development time.
   * They provide honest feedback about the challenges faced, particularly the insufficiency of the 2-hour time limit.
   * Insightful reflection on how AI tools like ChatGPT could have assisted in speeding up the process.
4. **Technical Content**:
   * The technical content is appropriate for the scope of the project and includes necessary information about the tools and technologies used.
   * However, the documentation could benefit from more detailed descriptions of the UML diagram and table design.

**Recommendations for Improvement**

1. **Expand on the UML Diagram**: Provide a detailed explanation of the UML diagram, describing the relationships and interactions between classes.
2. **Detailed Table Design**: Include more detailed information about the database schema, such as field types, constraints, and relationships.
3. **More Comprehensive Testing Section**: Describe any testing that was performed, including test cases and results, even if limited.
4. **Additional Insights**: Reflect on specific areas where the team struggled and how those could be addressed in future iterations or with additional tools/resources.

**Group 2 Documentation Assessment**

**Summary**

**Description of the Application:**

* **MainMenu Class**: Entry point of the application, presenting options to add, update, delete members, and generate membership reports.
* **AddMember Class**: Manages the addition of new members with a form for entering member information and communicates with DatabaseUtil to insert the data.
* **UpdateMember Class**: Allows searching for and updating existing member information with a form to edit member details, using DatabaseUtil for database operations.
* **DeleteMember Class**: Enables the deletion of members based on their ID, interacting with DatabaseUtil to remove member records.
* **Report Class**: Generates various reports based on user selection (all members, active members, etc.) and retrieves report data using methods in DatabaseUtil.
* **DatabaseUtil Class**: Utility class for database operations, containing methods for adding, updating, deleting, and retrieving member data, as well as generating reports.

**Graphical UML of the Architecture**: Included in the documentation.

**Analysis of Documentation Quality**

1. **Clarity and Organization**:
   * The documentation is well-organized, with clear descriptions of each class and its role within the application.
   * The structure of the document is logical, making it easy to follow.
2. **Detail and Completeness**:
   * The documentation provides a comprehensive overview of the application’s structure and the responsibilities of each class.
   * The inclusion of a UML diagram helps visualize the relationships between classes.
3. **Accuracy and Insight**:
   * The descriptions accurately reflect the functionality of each class as implemented.
   * Insightful in explaining how each GUI class interacts with the DatabaseUtil class for database operations.
4. **Technical Content**:
   * The technical content is appropriate and covers the key aspects of the application’s architecture and functionality.
   * The documentation could benefit from additional details about the implementation specifics, such as the database schema and any specific challenges faced during development.

**Recommendations for Improvement**

1. **Expand on the UML Diagram**: Provide a detailed explanation of the UML diagram, describing the relationships and interactions between classes in more depth.
2. **Detailed Table Design**: Include detailed information about the database schema, such as field types, constraints, and relationships.
3. **More Comprehensive Testing Section**: Describe any testing that was performed, including test cases and results.
4. **Additional Insights**: Reflect on specific areas where the team faced challenges and how those were addressed, as well as any improvements that could be made in future iterations.

**Comparative Analysis of Documentation Quality**

**Group 1 Documentation**:

* Well-organized and clear, with honest reflections on challenges faced.
* Lacks detailed explanations of UML diagram and table design.
* Feedback on insufficient development time and potential benefits of AI tools.

**Group 2 Documentation**:

* Well-organized with clear descriptions and a UML diagram.
* More complete in terms of describing each class and its responsibilities.
* Could benefit from additional details on implementation specifics and testing.

**Final Summary Table of Documentation Quality**

|  |  |  |
| --- | --- | --- |
| Criterion | Group 1 (Software Engineering) | Group 2 (Software Orchestration) |
| Clarity and Organization | 80% | 90% |
| Detail and Completeness | 70% | 85% |
| Accuracy and Insight | 75% | 85% |
| Technical Content | 70% | 85% |
| Overall Documentation Quality | **73.75%** | **86.25%** |

This summary table provides a comparative assessment of the documentation quality for each group, highlighting the strengths and areas for improvement in each.