Better Feedback Forms - Customizable and Graphic Interface

Engineering Design Review

Author: Yu Li 李渔 **Date**: October 2024

Introduction

Providing meaningful and constructive feedback is essential for promoting student learning and improvement. However, in many systems , the current feedback options are limited to just two inputs per student submission: a numeric overall grade and a single free-form text field for comments. This lack of flexibility hinders markers from delivering detailed, structured feedback tailored to specific elements of the assignment. As a result, students may not receive adequate guidance on how to improve specific aspects of their work. Furthermore, the absence of a standardized approach to feedback leads to inconsistencies across different assessments and markers, potentially causing confusion for students who rely on clear and comprehensive feedback to progress in their studies.

To tackle this issue, we propose the introduction of a custom feedback form feature that allows module organisers to create structured and assignment-specific feedback templates. These forms would enable markers to provide more detailed, multi-faceted feedback by including section-specific instructions, component marks, and multiple text fields for different aspects of the assignment. This solution aims to make feedback more comprehensive, organized, and consistent, thereby enhancing the feedback experience for both educators and students.

Goals and Non-goals

- **Goal**: Customizable introduction , free-form text fields and mark for each component of the assignment.
- **Goal**: Graphic interface which is easy to add mark fields.
- **Non-Goal**: Tag system allows markers to filter submissions.
- **Non-goal**: Students will see the feedback in the form , while marks will only be saved in database.

Design Overview

The whole system will be divided into two parts , the editing part and the marking part.

The new marks editing system called EDIT.exe will primarily consist of a group of buttons which can create text fields or input parts and a table showing current result of the editing.

After that , there will be a FINISH button in the panel . Once clicked , the result will be saved as a .toMark file which could not be edited anymore.

As for markers , a program called MARK.exe should be used for marking. This program read the data in .toMark file and allow markers to write.

When marks are done , markers can submit the result by SUBMIT button in the program. And the program will generate a read-only .Mark file.

The final file should be handed to students as their grades .

Editing System

Each button has a unique type and name , and optional text or input content according to its type. When one button is clicked , an item will be created in the data base and be shown on the table. Besides , each item will be given a specific id automatically. Main types are shown below:

- TITLE , a text only component without input , in big and bold font. Designed to be placed at the beginning of each section.
- INTRODUCTION , a text only component without input , used as its name.
- GRADE , a text about focus of this mark and a number input following , allowing the markers to fill the grades.
- COMMENT , a text about focus of this mark and a text input following , allowing the markers to write down their words.
- DELETE , a button only works in the editing page , removing the last item .

The format of an item above except DELETE is displayed as below:

ID | TYPE | NOTE | TEXT | INPUT

- ID , will be assigned by a counter function and invisible to the editors , markers or students . Only used in development and debugging.

- TYPE , will be determined by the button clicked and only visible to editors and markers . This element controls whether input part will be shown.
- NOTE , should be filled by the editors and only visible to editors and markers . Each item needs to have unique NOTE.
- TEXT , should be filled by the editors and visible to everyone. This element cannot be blank.
- INPUT , the only part should be filled by markers and visible to everyone . This element normally follows the TEXT . It can have two kinds of input , text or numbers , depending on it TYPE .

The rights to access and view elements are displayed below:

Element	Access	View
ID	Developer	Developer
TYPE	Developer	Developer,Editor,Marker
NOTE	Editor	Editor,Marker
TEXT	Editor	Editor,Marker,Student
INPUT	Marker	Marker,Student

Marking System

The MARK.exe receive file in .toMark suffix and create a graphic interface according to the file.

The interface will consiste of two parts , introduction text and following input fields and a SUBMIT button.

The program reads all input and creates three types of item.

- NAME , a simple text comes from the NOTE and indicates the target of item.
- INTRO , a paragraph comes from the TEXT and gives information about the task and result.
- MARK , an input field where markers should put the result in.Type from TITLE and INTRODUCTION won't generate MARK part.

After finishing the marking , markers need to click the SUBMIT button and the program will generate a .Mark file which contains the diverse grades of every aspect.

Student can just open .Mark file with notepad to check ,and .Mark file is easy to archive and could be used to assess the final grades.

Format of Result File

The format of .toMark and .Mark file above is below

.toMark File Format

The .toMark file contains the structure and template for grading, including the sections, instructions, and input fields that markers need to complete. It is generated by the editing program (EDIT.exe) and should no longer be edited once saved.

Fields/Attributes in .toMark file:

1. TYPE: Describes the type of component (e.g., TITLE, INTRODUCTION, GRADE, COMMENT).
2. NOTE: Editor-defined field for specific instructions or guidance for markers.
3. TEXT: Visible to all, providing the text content related to the item.

Each line represents an item and contains all elements above.

Example of .toMark content:

```
TYPE: TITLE, NOTE: "Section 1", TEXT: "Introduction"
TYPE: GRADE, NOTE: "Evaluate clarity", TEXT: "Clarity of Argument"
TYPE: COMMENT, NOTE: "Provide feedback", TEXT: "Additional feedback"
```

.Mark File Format

The .Mark file contains the finalized feedback after the markers have completed the grading process. It is a read-only file that students can open to view their feedback.

Fields/Attributes in .Mark file:

1. Field: The corresponding TEXT content from the .toMark file, giving context for the component.
2. Mark: The input values provided by the markers, either numeric grades or textual comments, depending on the TYPE.

Each line represents an item and contains all elements above.

Example of .Mark content:

```
Field: "Clarity of Argument"
Mark: 85

Field: "Additional feedback"
Mark: "Good work overall, but needs more focus on structure."
```

Alternatives

There are some alternatives methods to complete the project , below is the comparison between them and current plan.

Alternative UI as Using Commands in EDIT Part

Instead of clicking on the screen and typing simple textx, we could design a set of commands to generate the .toMark file.

Pros

1. High Flexibility: Code input allows users to create highly customized feedback structures that may not be available in the graphical interface. This is ideal for unique or non-standard assignments.
2. Lightweight and Efficient: Code input typically consumes fewer resources, as it doesn't require rendering a graphical interface. Could be more friendly to those old office computers.
3. Easier to Develop: Command-based systems are far more easier to develop because they don't require complex graphical elements or interfaces. We can focus on the core functionality, reducing the time and effort needed to build and maintain the system.

Cons

1. Increased Risk of Errors: Manual coding increases the likelihood of syntax errors or incorrect formatting, which could lead to malfunctioning feedback forms or grading structures.
2. Lack of Immediate Visualization: Users cannot see a real-time preview of what they are creating. They may need to run the code or wait until submission to check for mistakes, potentially leading to time delays.
3. Hard to Promote: Users accustomed to graphical interfaces may resist switching to a command-based system, even if it offers technical advantages. This can make it challenging to promote and integrate the system into existing workflows or organizations.

Alternative Method for Storing and Processing the Data

Using an SQL (Structured Query Language) database, rather than relying on predefined file formats, can significantly improve performance and scalability. However, introducing a new component may increase the complexity of the system. The decision to switch to SQL should depend on the overall development progress.

Alternative Data Organization For Potential Tag System

Different organization of data may allow setting a tag system into the project, which could enhance searchability and organization. But it also requires more resource in the development stage and may increase database load.

Milestones

Milestone 1: Make the regulation of data format after sufficient discussion, then draw UML graphs to make the connections and constrictions clear. This action would prevent defected design and eliminate potential ambiguity in future development.

Milestone 2: Develop the EDIT.exe editing system to create custom feedback templates. This system should include function buttons. Then generate random data to test the functionality to ensure that each button works correctly.

Milestone 2a: Draw an easy-to-use UI by Qt for cpp or other language. Fix all buttons in suitable places.

Milestone 2b: Complete the function of each buttons. Make sure that all components are in place.

Milestone 2c: Check the correctness and format of exported files. Compare the result with designed template.

Milestone 3: Develop the MARK.exe marking system, which will read the .toMark files created by EDIT.exe. And do the similar tests as above.

Milestone 3a: Design a perfect UI of the marking system and implement the function of reading .toMark file. Test with different data.

Milestone 3b: Check the output of MARK.exe and secure its compatibility of common text reader software.

Milestone 4: Invite students and teachers for alpha tests. Collect their opinions and feedbacks and adjust the project accordingly.

Milestone 5 (Optional): Explore the potential for a tagging system that allows markers to filter submissions based on specific criteria.

Milestone 6 (Optional): Develop a mechanism for localization of text and instructions, pursuing that the system can support diverse environment.

Dependencies

- *Data template team*: Will need to work out a perfect data format and negotiate with all members about the methods of conveying data.
- *UI team*: Will need to design the outlook of user inferface of EDIT.exe and MARK.exe. Need to reserve space for future possible changes.
- *Programming team*: Will need to implement all functions of the project, including the buttons and input fields in EDIT.exe and MARK.exe.
- *Legal team*: Need to review our legal agreements with customers to ensure that this feature does not expose us to any trouble.
- *Localisation team*: Will try to translate the texts of the project to target language.

Cost

All possible cost below are calculated without manual salary.

- Software Development Tools and Licenses : Need to pay for the copyright of cpp qt and other structure.
- Server Rents and Maintenance : Need to pay for the rents of running the server and potential maintenance
- Incentives and Rewards for Users in Test Stage : Need to offer rewards to testers to encourage participation, such as gift cards, vouchers, or product discounts.

Privacy and security concerns

All sensitive data collected in this project, such as student grades and other personal information, must remain protected under existing permissions and roles. And it is critical to follow established security practices when adding new features, such as tags, SQL database, and other functions, to avoid creating potential vulnerabilities. Additionally, any access to these routes should be logged using the current audit event system to ensure compliance and traceability.

Risks

Risk	Mitigation(s)
Changes to the feedback system may lead to confusion among users	Provide comprehensive training and documentation for all users to ensure they understand how to navigate the new system
Technical issues or server failures could result in system downtime	Establish a robust backup and recovery plan to minimize downtime in case of failures
Failure to follow data protection regulations	Implement regular audits to monitor compliance and make necessary adjustments

Supporting material

The following materials give enlightenment of security in the project.

- Ulven, J.B. and Wangen, G. (2021) A systematic review of cybersecurity risks in Higher Education, MDPI. Available at: https://www.mdpi.com/1999-5903/13/2/39#
- Mukherjee, M. et al. (2024) Strategic approaches to cybersecurity learning: A study of educational models and outcomes, MDPI. Available at: https://www.mdpi.com/2078-2489/15/2/117