CSCI370: Field Session

APPLIED MATHEMATICS & STATISTICS PROBLEM BANK & TEX EXAM BUILDER

submitted to

Dr Terry Bridgman

Daniel Greenberg

Colorado School of Mines

June 13, 2024

By

Brandon Ching

Landon Gehr

Brandon Lechten

Erik Luehrmann

Abstract

The Problem Bank tool is a software application developed to aid the Applied Mathematics & Statistics department in archiving exam problems along with their associated statistical performance data. This tool allows users to efficiently store, manage, and retrieve exam problems, facilitating the creation of new exams based on specific criteria such as problem type and difficulty level. The core functionalities include the entry of raw LaTeX data, handling additional files like graphics, and maintaining detailed performance statistics. The development of the Problem Bank tool followed a rigorous Quality Assurance plan, incorporating multiple levels of testing—unit, integration, system, and user acceptance—to ensure the tool's robustness, reliability, and user-friendliness. Ethical considerations, such as data confidentiality and adherence to ACM/IEEE guidelines, were integral to the development process. Initial user feedback has been positive, highlighting the tool's intuitive interface and effectiveness in meeting the department's needs. Future enhancements may include a cloud-based version and improved user interface features, further enhancing the tool's usability and accessibility.

Table of Contents

Abstract **ii**

Table of Contents **iii**

List of Figures **iv**

List of Tables **iv**

1 Introduction **1**
1.1 Client 1

2 Requirements **1**
2.1 Functional Requirements 2
2.2 Non-functional Requirements 2

3 Risks **3**
3.1 Technical Risks 3
3.2 Project Management Risks 3

4 Definition of Done **3**

5 System Architecture **3**
5.1 Technical Design Issues 3
5.2 System Design 4
5.3 Database Design 6
5.4 User Interface Design 7

6 Software Test and Quality **8**
6.1 Overview 8
6.2 Test Environment 8
6.3 Testing Phases 8
6.3.1 Unit Testing 8
6.3.2 Integration Testing 8
6.3.3 System Testing 8
6.3.4 User Acceptance Testing (UAT) 9
6.4 Test Cases 9

7 Ethical Considerations **10**
7.1 ACM/IEEE Code of Ethics 10
7.2 Michael Davis' Framework Test 10
7.2.1 Harm Test 10
7.2.2 Legality Test 11

8 Results **11**

9 Future Work **12**

10 Lessons Learned **12**

11 Acknowledgements **13**

12 Team Profile **15**

A Glossary of Terms, Abbreviations, and Acronyms **16**

B Project Prompt **17**

C References **18**

List of Figures

1 User Flow Diagram 5
2 Database UML Diagram 6
3 Sketches of GUI 7

List of Tables

1 Test Cases for Problem Bank Tool 9

1 Introduction

This project aims to develop a comprehensive tool for the Applied Mathematics & Statistics department to manage and utilize a problem bank effectively. This tool facilitates the storage, retrieval, and analysis of exam problems, enabling instructors to construct exams tailored to specific criteria such as problem type and difficulty level. By centralizing problem storage and performance data, the tool enhances the efficiency and quality of exam creation and assessment processes.

1.1 Client

The client for this project is Dr Terry Bridgman, a professor in the Applied Mathematics & Statistics department at the Colorado School of Mines. Dr Bridgman has a wealth of experience in teaching and curriculum development and is passionate about enhancing the learning experience for students - Dr Bridgman has been a professor for 25+ MATH courses at the Colorado School of Mines since 2003[1]. Dr Bridgman is looking for a tool to help streamline the process of creating exams and analyzing student performance data. The Tex Exam Builder provides a centralized platform for storing exam problems, statistics, and additional files, enabling Dr Bridgman to manage and utilize his problem bank efficiently.

2 Requirements

Per the original project prompt from the client, the Problem Bank tool had two very simple requirements:

1. Allow user to add test problems and associated (if any) graphics/files along with topic categories and performance statistics per problem.
2. Allow user to 'build' an exam by specifying problem category and difficulty level.

Dr Bridgman also provided a list of additional 'nice-to-have' features that could be implemented if time allowed. These features included:

1. A majority of the instructors use LaTeX for test creation. Thus, the problems would need to be stored in their raw form, preferably LaTex.
2. Some problems have associated graphics which would also need to be stored and accessed along with the problem.

3. Statistics (e.g., mean/median, recommended points, etc.) needs to be stored and referenced /reported upon request.
4. The category of problems requires flexibility as the list may grow over time.
5. More recent problems also have an associated rubric which should be reported upon problem request.

Based on the client's requirements and additional features, the team developed a comprehensive set of functional and non-functional requirements for the Problem Bank tool. These requirements were used to guide the development process and ensure the tool met the client's needs effectively.

2.1 Functional Requirements

- Must be able to enter raw LaTeX data (code block is per question and not necessarily compilable)
- Must be able to store multiple sets of statistics (min, max, mean, median, standard deviation) per problem
- Must be able to store additional LaTeX including headers
- Must be able to store additional files (images, etc) per problem
- Must be able to filter by year, course, exam, question number, question type, difficulty
- Must produce an output file of selected questions into a compiled LaTeX document
- Must provide a compiled preview of each question

2.2 Non-functional Requirements

- Must run locally on MacOS
- Must have a GUI interface
- Comprehensive documentation must be provided for users and developers.
- The program must be easy to install and use.

3 Risks

3.1 Technical Risks

- Compatibility issues with different LaTeX versions or packages: Due to the large archive of old exams, when compiling, there may be compatibility issues with different versions of LaTeX and/or packages.
- Handling large quantities of data: The system must be able to handle a large number of problems and statistics. This could lead to performance issues if not handled properly. Data needs to be efficiently stored and retrieved both in terms of speed and memory.
- System Scalability: The system must be able to handle a large number of problems and statistics. By the nature of the locally run software, user scalability is not a concern.

3.2 Project Management Risks

- Time: The project must be completed by the end of the 5-week term. If the project is not completed by the deadline, the client may not accept the software.
- Scope Creep: The core functionality of the project is well defined, but additional features may be requested by the client and/or proposed by the team which could lead to delays in project completion. Additional features may also be incomplete and/or poorly implemented due to time constraints and too many features being added.

4 Definition of Done

The product must be able to store and add problem data to some sort of database/file structure, and be able to retrieve that data filtered by problem type and difficulty. The client shall accept the software if they can successfully add new problem data to the product, and can retrieve the raw LaTeX for problems filtered in the ways that they want. A final product must be reviewed and approved by the client to be done.

5 System Architecture

5.1 Technical Design Issues

Per a client requirement, the program must run on MacOS. This requirement limits the technology stack to those that are compatible with MacOS. The majority of the development team is developing

on Windows machines, further limiting the stack to those that are cross-platform. The team has decided to use Python for its cross-platform compatibility with MacOS and Windows. This allows for the team to develop the software on Windows machines and then test and deploy it on MacOS.

Some of the libraries used in the project were outdated and required updating to work with the latest version of Python. Instead of installing through pip, the team added the libraries source code to the project and imported them directly. This allowed the team to modify the libraries to work with the latest version of Python.

Additional, some libraries had compatiblity and performance issues, resulting in older and more stable versions being used; this was the case with the Chlorophyll library, which was used to syntax highlight files. The team found that the latest version (0.4.2) of the library had significant performance issues when displaying large files, and reverted to version 0.4.1, which was more stable and performed better.

5.2 System Design

The system is designed to be a simple GUI that allows the user to interact with the database by adding, filtering, and previewing problems. The system is built using Python and the Tkinter library for the GUI and SQLite for the database. Python was chosen for its ease of use and compatibility with the client's existing systems. SQLite was chosen for its simplicity and ease of use for a small-scale locally stored database. Both of these technologies are cross-platform and run on MacOS as well as other operating systems should the client choose to use them or share this tool with other professors.

A basic user flow diagram is shown below in Figure 1. The user can add problems, statistics, tags, and additional files. The user is also able to filter problems by tags, difficulty, and other criteria. A preview of the compiled LaTeX is available for each problem, and the user can export a compiled LaTeX file of selected problems.

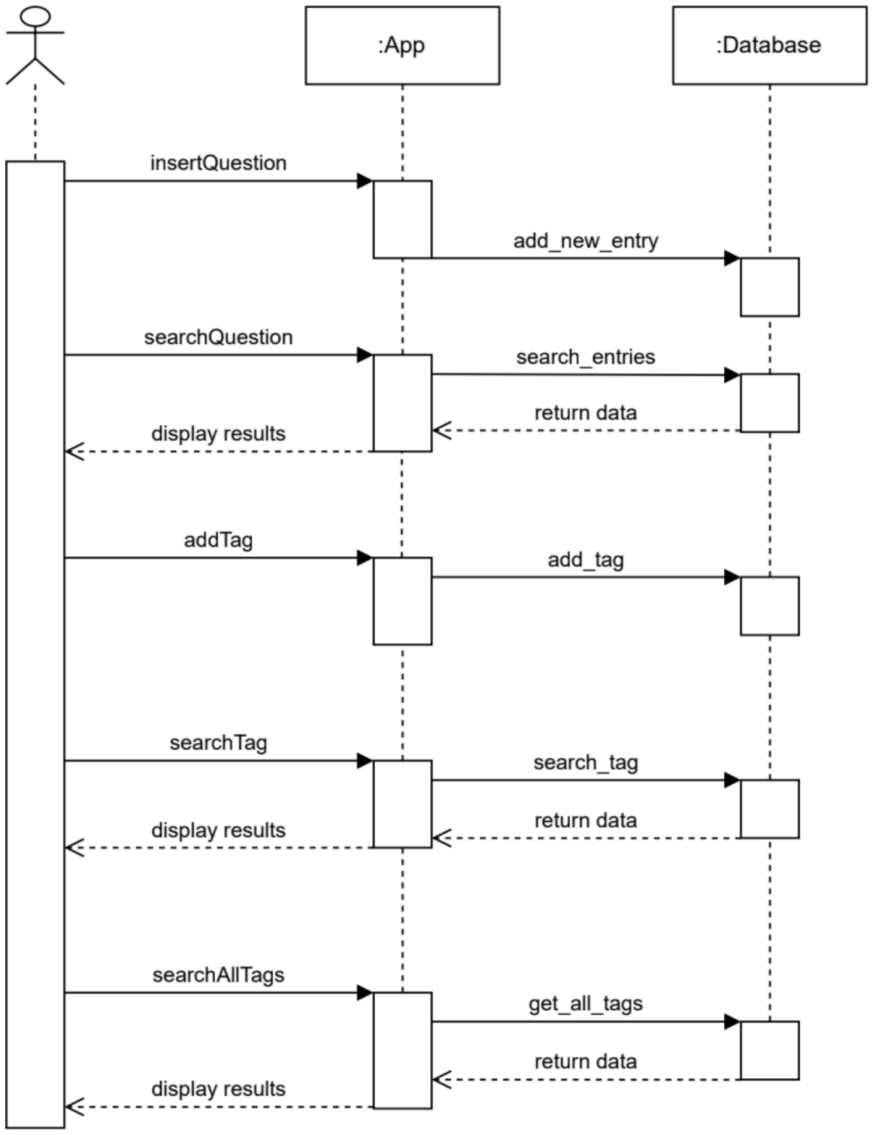


Figure 1: User Flow Diagram

5.3 Database Design

The database is designed to store problems and their associated statistics, tags, and additional files. A SQLite database is used to store the data, with tables for questions, statistics, tags, and exams. A full UML diagram for the database schema is provided in Figure 2. Files (including associated files and any temp files generated during the compiling of LaTeX) are stored as objects in a hidden data directory locally on the user's machine. The database is accessed through a Python interface, which handles the retrieval and storage of data.

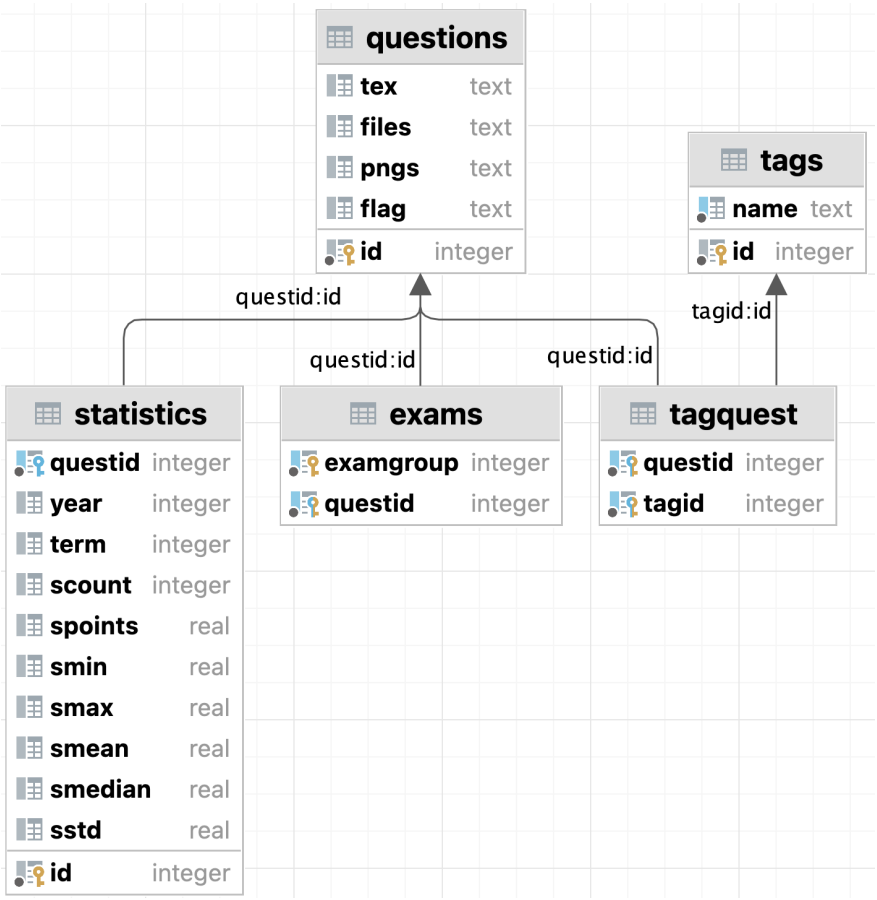
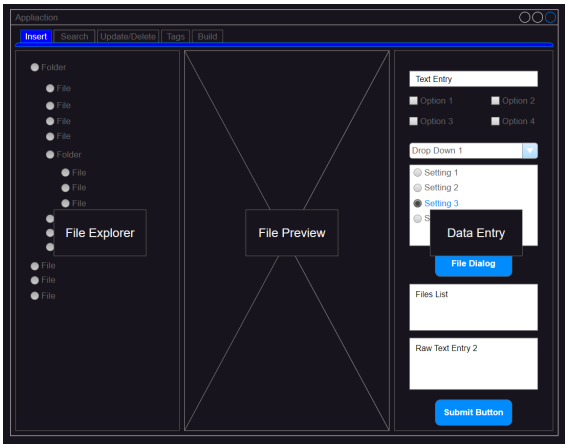


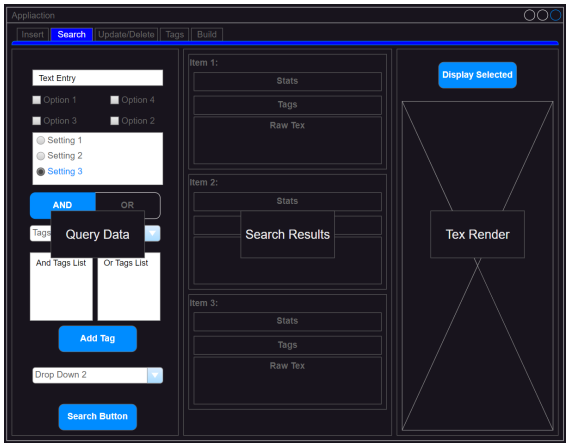
Figure 2: Database UML Diagram

5.4 User Interface Design

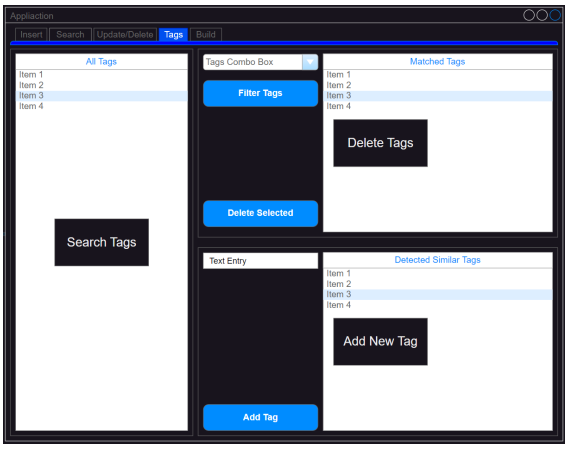
The user interface is designed to be user-friendly and intuitive. The UI is built using the Python standard library tkinter. The UI allows the user to input raw LaTeX data, add statistics, add tags, and add additional files. The user can filter problems by tags, difficulty, and other criteria. The UI generates and shows the user a preview of the compiled LaTeX of a problem, and exports a compiled LaTeX file of selected problems. Shown below are sketches of each of the GUI tabs.



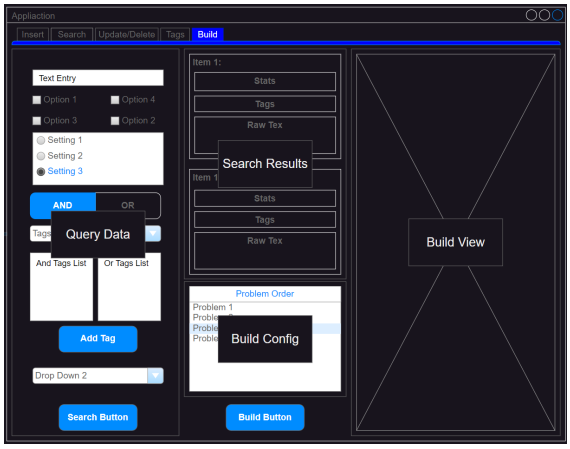
(a) Insert Tab



(b) Search Tab



(c) Tags Tab



(d) Build Tab

Figure 3: Sketches of GUI

6 Software Test and Quality

6.1 Overview

The Quality Assurance (QA) plan for this project is designed to ensure that the tool meets all functional and non-functional requirements specified. The QA strategy involves multiple levels of testing to identify and resolve defects early, ensuring the final product is robust, reliable, and user-friendly.

6.2 Test Environment

All testing are conducted on MacOS to match the deployment environment.

6.3 Testing Phases

6.3.1 Unit Testing

Where possible, each component of the tool is tested independently to verify it performs as expected. This primarily consists of testing backend functions such as LaTeX parsing, database storage, and retrieval. A suite of unit tests, built using the Python pytest framework, automatically tests all backend routes to ensure they are functioning correctly. These tests are run regularly on MacOS to ensure compatibility. The tests can be seen in Table 1.

6.3.2 Integration Testing

After unit testing, integration tests ensure that different modules and components of the tool interact correctly. This phase involves checking the combined functionality of LaTeX parsing, storage, and retrieval systems.

6.3.3 System Testing

System testing is performed on the complete, integrated tool to evaluate its compliance with the specified requirements. This phase involves testing the tool's performance on MacOS, its GUI interface, and its ability to produce compilable LaTeX documents and previews.

6.3.4 User Acceptance Testing (UAT)

Finally, UAT is conducted regularly with actual users from the Applied Mathematics & Statistics department (Dr Bridgman). Feedback is collected to ensure the tool meets their needs and expectations, and any identified issues are addressed before final deployment. UAT is to be conducted frequently in parallel with other testing phases to ensure that the tool is developed according to user requirements.

6.4 Test Cases

Test Case ID	**Description**	**Expected Result**
TC001	Enter raw LaTeX data for a problem	The problem is stored correctly with all LaTeX data intact
TC002	Store statistics for a problem	Statistics (min, max, mean, median, standard deviation) are correctly stored in the database
TC003	Include additional LaTeX headers	Headers are applied correctly compiled documents
TC004	Upload additional files (e.g., images) for a problem	Files are stored and linked correctly with the corresponding problem
TC005	Filter problems by criteria (difficulty, course, etc.)	Filtering returns accurate results based on specified criteria
TC006	Runs on MacOS	Tool operates fully when deployed on MacOS
TC007	Use GUI interface for interaction	GUI is responsive and user-friendly, all functionalities accessible
TC008	Parse raw LaTeX files for individual questions	Tool accurately parse LaTeX files into individual questions
TC009	User selects to preview a question	Generate and display the preview of the questions
TC010	Generate a complete exam	A compilable LaTeX is provided for the complete exam

Table 1: Test Cases for Problem Bank Tool

7 Ethical Considerations

As a tool that stores and manages exam problems and historical statistics, it is important to ensure that the data stored is secure and confidential. No personally identifiable information is stored in the database (only aggregated data is stored), and all data is stored locally on the user's machine. The tool does not use or interact with any external services, ensuring that data is not shared or accessible to any third parties.

7.1 ACM/IEEE Code of Ethics

The development of the Problem Bank tool adheres to the ACM/IEEE Code of Ethics. Key principles that are followed include:

- 2.05. Keep private any confidential information gained in their professional work, where such confidentiality is consistent with the public interest and consistent with the law.
- 3.10. Ensure adequate testing, debugging, and review of software and related.
- 3.11. Ensure adequate documentation, including significant problems discovered and solutions adopted, for any project on which they work.

By following these ethical guidelines, the team ensures that the Problem Bank tool is developed and deployed responsibly and ethically.

7.2 Michael Davis' Framework Test

The Problem Bank tool has been evaluated using Michael Davis' Framework Test to ensure ethical considerations are met. The tool is assessed based on the following criteria:

7.2.1 Harm Test

Does this option do less harm than any alternative? Do the benefits outweigh the harms?

There is no harm associated with the development and deployment of the Problem Bank tool. The tool is designed to enhance the efficiency and quality of exam creation and assessment processes for the Applied Mathematics & Statistics department. The benefits of the tool, including improved data management and analysis, outweigh any potential harm.

7.2.2 Legality Test

Would this choice violate a law or a policy of my employer?

As the tool stores exam problems and statistics, it is important to ensure that no student data or personally identifiable information is exposed. It is also important to ensure that the integrity of the exam problems is maintained per Colorado School of Mines policy. There is little to no risk of any data exposure or integrity issues as the data is stored locally and the tool does not interact with any external services. This ensures only users with credentials to the machine can access the data(ie only Dr Bridgman).

8 Results

The Problem Bank tool successfully meets the core functional and non-functional requirements outlined at the beginning of the project. The key results include:

- **Functional Performance:**
  - The tool allows for the entry of raw LaTeX data, including additional files.
  - Users can store multiple sets of statistics (min, max, mean, median, standard deviation) for each problem.
  - Filtering functionality is operational, enabling filtering by year, course, exam, question number, question type, and difficulty.
  - The tool allows for the user to filter by tags, difficulty, and other criteria.
  - The tool allows for the user to build and compile a LaTeX document of selected questions, including the creation of a full exam.
- **Non-Functional Performance:**
  - The tool runs locally on MacOS and provides a comprehensive GUI.
  - A full set of documentation is provided for users and developers. Additionally, the code base is well-documented, including quality comments, for future maintenance.
- **User Feedback:**
  - Initial feedback from the Applied Mathematics & Statistics department indicates that the tool is intuitive and meets the intended use cases effectively.
  - Minor adjustments were made to enhance usability and functionality based on user feedback.

Overall, the project was completed on time, and the client has accepted the software, with plans for continued use and potential future enhancements.

9 Future Work

Future work on the Problem Bank tool could include the following enhancements:

- **Cloud-based version** While not the initial scope of the project, a cloud-based version of the tool could be developed to allow for easier sharing and collaboration among instructors. This also allows the tool to be utilized by a wider audience.
- **Automated Parsing of LaTeX Files** Implementing an automated parser for LaTeX files could streamline the process of entering problems into the system. This involves extracting questions and answers from LaTeX files and storing them in the database.
- **Improved UI/UX** While the current GUI is functional, further improvements could be made to enhance the user experience. This could include additional features such as drag-and-drop functionality for adding files, improved filtering options, and a more visually appealing interface.
- **Generative Adversarial Networks (GANs)** Implementing GANs to generate new problems based on existing problems in the database could be a valuable feature. This would allow instructors to create new problems based on existing ones, maintaining consistency in difficulty and style.

10 Lessons Learned

Throughout the development of the Problem Bank tool, several key lessons were learned:

- **Project Management:**
  - *Time Management:* Adhering to a strict timeline was essential. Weekly milestones and regular check-ins with the client ensured steady progress.
  - *Scope Creep:* Clearly defining the project scope and adhering to it helped prevent delays. Any additional features were carefully considered and prioritized based on their impact. The team faced some scope creep, but was able to manage it effectively, while still delivering a high-quality product, beyond the client's initial expectations.

- **Technical Solutions:**
  - *GUI Design:* Using Python's tkinter library for the GUI interface was effective in creating a user-friendly application. Iterative design and user testing were crucial for refining the interface.
  - *Modular Architecture:* Designing the system with a modular architecture facilitated easier debugging and future enhancements. Each component was independently tested and then integrated.
- **User Feedback:**
  - Regular user feedback was invaluable. Engaging with the end-users throughout the development process ensured the tool met their needs and expectations. Client feedback was the most important aspect of the project, in informing the team of what was needed and what was not.
- **Ethical Considerations:**
  - Ensuring data privacy and security was paramount. Storing data locally and avoiding external services helped maintain the confidentiality and security of exam problems and performance data.

Overall, the project provided valuable insights into project management, technical design, user feedback, and ethical considerations, contributing to a successful outcome. The team is grateful for the opportunity to work on this project and looks forward to applying these lessons to future endeavors.

11 Acknowledgements

The team thanks Dr Terry Bridgman for providing the opportunity to work on this project and for his guidance and feedback throughout the development process. The team also appreciates the support and resources provided by Daniel Greenberg and the CSCI370 course staff, which were instrumental in the successful completion of the project.

Additionally, the team extends their gratitude to the guest lecturers and industry professionals who shared their insights and expertise, contributing to a valuable learning experience. Special thanks to:

- Jesse Garland, Chris Jezek, and Michael Garioto from WWT, for presenting "Building the Right Thing: Understanding the client goals and problems to maximize the value of your team's backlog." [2]
- Matthew McElhaney and Mike Buckner from BPX, for their talk on system architecture. [3]
- Troy Sornson from Salesforce, for his talk on quality assurance. [4]
- Paul Christopher from Google, for presenting "So you have a CS degree, now what?" [5]

12 Team Profile

Brandon Ching
Graduated Mechanical Engineering & Computer Science
Hometown: Coppell, Texas

Landon Gehr
Major: 2nd Year Computer Science
Hometown: Louisville, Colorado

Brandon Lechten
Major: 2nd Year Computer Science
Hometown: Boise, Idaho

Erik Luehrmann
Major: Senior Applied Math and Statistics/Computer Science
Hometown: Austin, Texas

A Glossary of Terms, Abbreviations, and Acronyms

Term	Definition
ACM	Association for Computing Machinery
AMS	Applied Mathematics & Statistics
CSCI	Computer Science Course Code
GANs	Generative Adversarial Networks
GUI	Graphical User Interface
IEEE	Institute of Electrical and Electronics Engineers
LaTeX	A document preparation system
QA	Quality Assurance
SQL	Structured Query Language
UAT	User Acceptance Testing
UI	User Interface
UX	User Experience
UML	Unified Modeling Language

B Project Prompt

COLORADO SCHOOL OF MINES

Background

Each term in the Applied Mathematics & Statistics department consists of 2-3 exams consisting of 8-10 problems per exam, per course. With the advent of Gradescope and Canvas statistics on the students' performance has been recorded per problem over the last 5-7 years. The problems are not identical from term to term though they are somewhat similar. Unfortunately, they may also differ in difficulty. There is a need to archive these problems in such way that a sample exam could be constructed based on problem type and statistical performance. For example, 'I need a problem on Integration by Parts of medium difficulty'.

Project Description

This tool is basically a problem bank. Exam problems over the years would be archived along with student performance. The user would be able to construct a new exam by requesting problem types and difficulty levels. The archive would be maintained and grow with each subsequent terms adding to the problem bank.

The tool needs to support 2 very simple operations:

1. Allow user to add test problems and associated (if any) graphics/files along with topic categories and performance statistics per problem.
2. Allow user to 'build' an exam by specifying problem category and difficulty level.

However, beyond these simple tasks are the following requirements:

1. A majority of the instructors use LaTeX for test creation. Thus, the problems would need to be stored in their raw form, preferably LaTex.
2. Some problems have associated graphics which would also need to be stored and accessed along with the problem.
3. Statistics (e.g., mean/median, recommended points, etc.) needs to be stored and referenced/reported upon request.
4. The category of problems requires flexibility as the list may grow over time.
5. More recent problems also have an associated rubric which should be reported upon problem request.

Desired Skill Sets

- Some database understanding. This project is not necessarily requiring the use of a database application, but some understanding of relational database concepts may aid in the development of the archive structure.
- An understanding of LaTeX. Some problems may have dependencies that are not immediately obvious (i.e., additional packages, user commands/defines, etc.). The developers should have some familiarity with the language.
- Some fundamental understanding of user interface design.

C References

[1] T. Bridgman, "Terry bridgman - applied mathematics and statistics." https://ams.mines.edu/project/bridgman-terry/, 2024.

[2] J. Garland, C. Jezek, and M. Garioto, "Building the right thing: Understanding the client goals and problems to maximize the value of your team's backlog.." Guest lecture at Colorado School of Mines, May 2024. Presentation available at https://cs-courses.mines.edu/csci370/Slides/GuestSpeakers/WWT_GuestSpeaker.pdf.

[3] M. McElhaney and M. Buckner, "Architecture and design." Guest lecture at Colorado School of Mines, May 2024. Presentation available at https://l48prod-my.sharepoint.com/:p:/g/personal/matthew_mcelhaney_bpx_com/EWDzY0oVsVBFto4mL_HSLZEB3l5-Neua6uLnl2ZWwfd0dA?rtime=SsCsFMuL3Eg.

[4] T. Sornson, "Quality assurance and testing." Guest lecture at Colorado School of Mines, May 2024. Presentation available at https://cs-courses.mines.edu/csci370/Slides/GuestSpeakers/Salesforce_GuestSpeaker.pdf.

[5] P. Christopher, "So you got a cs degree ... now what?." Guest lecture at Colorado School of Mines, June 2024. Presentation available at https://docs.google.com/presentation/d/1ay6Lq2mP4UIEGJexwXtL-Yzi7LunuNOpkAfoIQCdhnU/edit#slide=id.p.