Instructor Grading Program

By

Nicholas Aaron Bryant and Vinoth Kumar Manickavasagam

An Honors Capstone

submitted in partial fulfillment of the requirements

for the Honors Diploma

to

The Honors College

of

The University of Alabama in Huntsville

April 24th 2019

Honors Capstone Director: Dr. Richard Coleman

Lecturer, Computer Science Department

Nicholas Aaron Bryant 4-23-2019
Student Date

4-23-2019
Student Date

Richard L Coleman 4-23-2019
Director Date

H S Ranganath 4-23-2019
Department Chair Date

5/1/19
Honors College Dean Date

HONORS COLLEGE
THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

Honors College
Frank Franz Hall
+1 (256) 824-6450 (voice)
+1 (256) 824-7339 (fax)
honors@uah.edu

Honors Thesis Copyright Permission

This form must be signed by the student and submitted as a bound part of the thesis.

In presenting this thesis in partial fulfillment of the requirements for Honors Diploma or Certificate from The University of Alabama in Huntsville, I agree that the Library of this University shall make it freely available for inspection. I further agree that permission for extensive copying for scholarly purposes may be granted by my advisor or, in his/her absence, by the Chair of the Department, Director of the Program, or the Dean of the Honors College. It is also understood that due recognition shall be given to me and to The University of Alabama in Huntsville in any scholarly use which may be made of any material in this thesis.

Nicholas Aaron Bryant

Student Name (printed)

Nicholas Aaron Bryant

Student Signature

Vinoth Kumar Manickavasagam

Student Name (printed)

Student Signature

04-23-2019

Date

Instructor Grading Program

By

Nicholas Aaron Bryant and Vinoth Kumar Manickavasagam

An Honors Capstone

submitted in partial fulfillment of the requirements

for the Honors Diploma

to

The Honors College

of

The University of Alabama in Huntsville

April 24th 2019

Honors Capstone Director: Dr. Richard Coleman

Lecturer, Computer Science Department

_____

Student Date

_____

Student Date

_____

Director Date

_____

Department Chair Date

_____

Honors College Dean Date

Honors College
Frank Franz Hall
+1 (256) 824-6450 (voice)
+1 (256) 824-7339 (fax)
honors@uah.edu

Honors Thesis Copyright Permission

This form must be signed by the student and submitted as a bound part of the thesis.

In presenting this thesis in partial fulfillment of the requirements for Honors Diploma or Certificate from The University of Alabama in Huntsville, I agree that the Library of this University shall make it freely available for inspection. I further agree that permission for extensive copying for scholarly purposes may be granted by my advisor or, in his/her absence, by the Chair of the Department, Director of the Program, or the Dean of the Honors College. It is also understood that due recognition shall be given to me and to The University of Alabama in Huntsville in any scholarly use which may be made of any material in this thesis.

_____

Student Name (printed)

_____

Student Signature

_____

Student Name (printed)

_____

Student Signature

_____

Date

Table of Contents

Abstract

The Instructor Grading Program is a standalone desktop application developed using Java and H2. The application is designed and implemented for use on Windows or OSX based personal computers. The application was designed and developed as part of the Senior Design Project class from January 2019 to April 2019. The application with base feature sets was developed by the entire team while additional requirements were detailed and developed for fulfilling the requirements for the Honors Capstone project. The features developed as part of the Honors Capstone requirements are functionality to export grades to a spreadsheet document, email individual students their grade reports and a question bank which can generate tests, quizzes and homework assignments.

Introduction

The Instructor Grading Program (IGP) provides professional academic instructors with a consolidated, comprehensive software tool for recording and calculating student grades based on a number of user-defined criteria. The workflow and dataflow in IGP was designed keeping in mind the various requirements and syllabus styles that instructors can follow. The GUI was designed recognizing the need to have a clean, consistent and consumer-friendly interface. In order to properly develop the features detailed for the Honors Capstone project, several supporting features were developed. The implementation of features were done using object-oriented programming principles, making use of several well recognized design patterns such as the Singleton Design Pattern and Bridge Design Pattern. As development progressed, the features were unit-tested to ensure that the implementation worked as designed and further tested as part of the integration and UI testing. This paper details the requirements, libraries, tools and design used in the implementation of features as part of the Honors Capstone project.

Requirements

The purpose of the CS-499: Senior Design course is to create a program that allows an instructor to enter and manage information about the courses that he teaches. In addition to the requirements outlined by the professor of CS-499 for the project, we created requirements for the Honors Capstone Project. The additional requirements are listed below:

1. The user can export grades to a spreadsheet document
2. The user can email grade reports to students from the program
3. The user can add questions to a “bank” that can later be used to generate tests and assignments.

These requirements were discussed with the project director and agreed upon as relevant to the IGP project.

Design

While designing the project, the team discussed tools necessary to develop the program, the structure of the database, the structure of the model used in the program, the design of services for exporting and emailing grades, and the design of the graphical user interface (GUI).

Tools

The team decided to use Java to develop the application as all team members had prior experience with it and its ease of portability. We decided to use Java 8 because the team had not had any experience with the newer Java 9 at the time. We also decided to use IntelliJ IDEA as our IDE, as it worked best on all development machines. For our embedded database, we chose to use H2 as it was recommended for its speed and ease of use, as well as offering password protection straight out of the box.

We used various other tools to enable faster and coordinated development:

- **BitBucket** – Online code repository
- **Trello** – Project Management
- **Slack** – Team Communication

Libraries

To develop the features as specified, we made use of several libraries which provided the basic foundation.

JUnit

JUnit is a unit testing framework and is the most commonly used external library in Java. We used JUnit 5 across the project to ensure that all methods worked as designed.

OpenCSV

OpenCSV is a CSV parser library for Java and the goto choice for CSV operation in Java, as Java does not provide native CSV support. OpenCSV supports all the basic CSV-type operations you are want to do. We used OpenCSV to export all term information into a .csv file and all course information (including assignments and grades) into a .csv file.

JavaMail

JavaMail API provides a platform-independent and protocol-independent framework to build and send emails. We use the JavaMail API to construct emails and add attachments to it and send it to all students individually.

PDFBox

PDFBox is an open source Java PDF library for generating and manipulating pdf files. We use PDFBox library to print an array of String elements into a pdf file. However, PDFBox does not provide support for end of line, end of page or single word recognition. We had to use the basic library functions and then implement our own controls on top of it to ensure that printing is accurate and according to the way we desire it.

Database

For the honors requirements, very little database designing needed to be done. The database needed a table for storing questions for the question bank. The “Questions” table used a foreign key to the “Courses” table used in the IGP to link questions to a particular course. Additionally, the table had fields for various properties of the question like the type of assignment the question might be used on, difficulty of the question, estimated time to complete

the question, the type of question, and amount of points that the question is worth. These properties were used as organizational tools for the instructor to sort by when viewing the questions in the IGP. The table also had a field for answers to the question if that question is a multiple choice question.

The other table needed for the honors requirements was the “Settings” table. This table was used to store the email address and password that would be used for emailing grade reports to students. Storing the email address and password in the database was determined to be better than any other method of storage because the database was already password protected along with the rest of the data used for the project.

Model

The model used for the honors requirements consisted of a set of courses and a set of questions. The model would allow one course in the set to be “selected” and use the course’s foreign key to populate the set of questions from the database. The model allowed questions to be added and removed from the set while a course was selected. Additionally, any number of questions in the set could be “selected” to designate them to be used to generate an assignment.

Services

The application was designed in such a way that a lack of add-on services would not affect core functionality of the program. This modularity is very useful when it comes to adding and removing features. Since our Honors Capstone requirements were requirements which were additional to the core functionality, we implemented them under a services submodule and kept them seperate from the core application for better management of code.

There were several services which we implemented for the program and some of them are listed below:

OSService

The OSService class is used to identify the Operating System that the application is running on. Based on the OS, the constructor for this class checks if folders/directories called “Instructor Grading Program” exists in the temporary app data location and desktop location. If the folder/directory do not exists, it creates them at those locations. The folder/directory in the desktop location is used to store csv and pdf files. The folder/directory in the temporary app data location is use to store files used for printing and sending emails, and they are deleted after their use is complete. This class follows the singleton design pattern.

PDFService

The PDFService class was implemented using PDFBox library at its core. This class allows has code to generate pdf files for a blank listing of student, individual student grade reports, overall class grade report and assignment grade report. This methods in this class check for the longevity (temporary or permanent) of the file based on the parameters given to the method and creates the files in the appropriate location. This class follows the singleton design pattern.

PrintService

The PrintService class makes use of the PDFService class and javax.print module to enable sending a file directly to a printer. We use PDFService to generate a pdf file and store it in a folder/directory inside the temporary app data location. The pdf file gets

loaded into a input stream which is used to send the data to the printer as selected by the user the javax.print dialog box. This class follows the singleton design pattern.

CSVService

The CSVService class was implemented using the OpenCSV library at its core. This class has two big methods. The course method allowed all the information about the course, such as the students in the course, the assignments under various categories, grades for each assignment to be exported as a .csv file. The term method allowed all the information about the term and the list of courses in that term to be exported as .csv file. This class follows the singleton design pattern.

EmailService

The EmailService class is designed to send out emails to entire courses or to individual students. The PDFService class is used to generate password protected files which are added as attachments to the emails.This class follows the singleton design pattern.

GUI

The question bank required several views to be added to the program. The main view was added to allow the user to see a list of all of the questions in a course. A form view was added to allow the user to enter information about a question whenever the user wanted to add a question to a course or edit a question already in the course. Finally, a view for generating an assignment was added that allowed the user to print out an assignment.

The main view consisted of a table showing the set of questions defined in the model, a combobox that allowed the user to select a course from the set of courses defined in the model,

and buttons for generating an assignment and editing, deleting, and adding questions. The table of questions allowed the user to sort the questions by any of the fields defined in the database and select any of the questions in the table with a checkbox so that they would be used in generating an assignment. The table was designed to be sortable on all of the question fields so that the user could easily find questions using its organizational properties. A screenshot of the main view is shown in Figure 1.

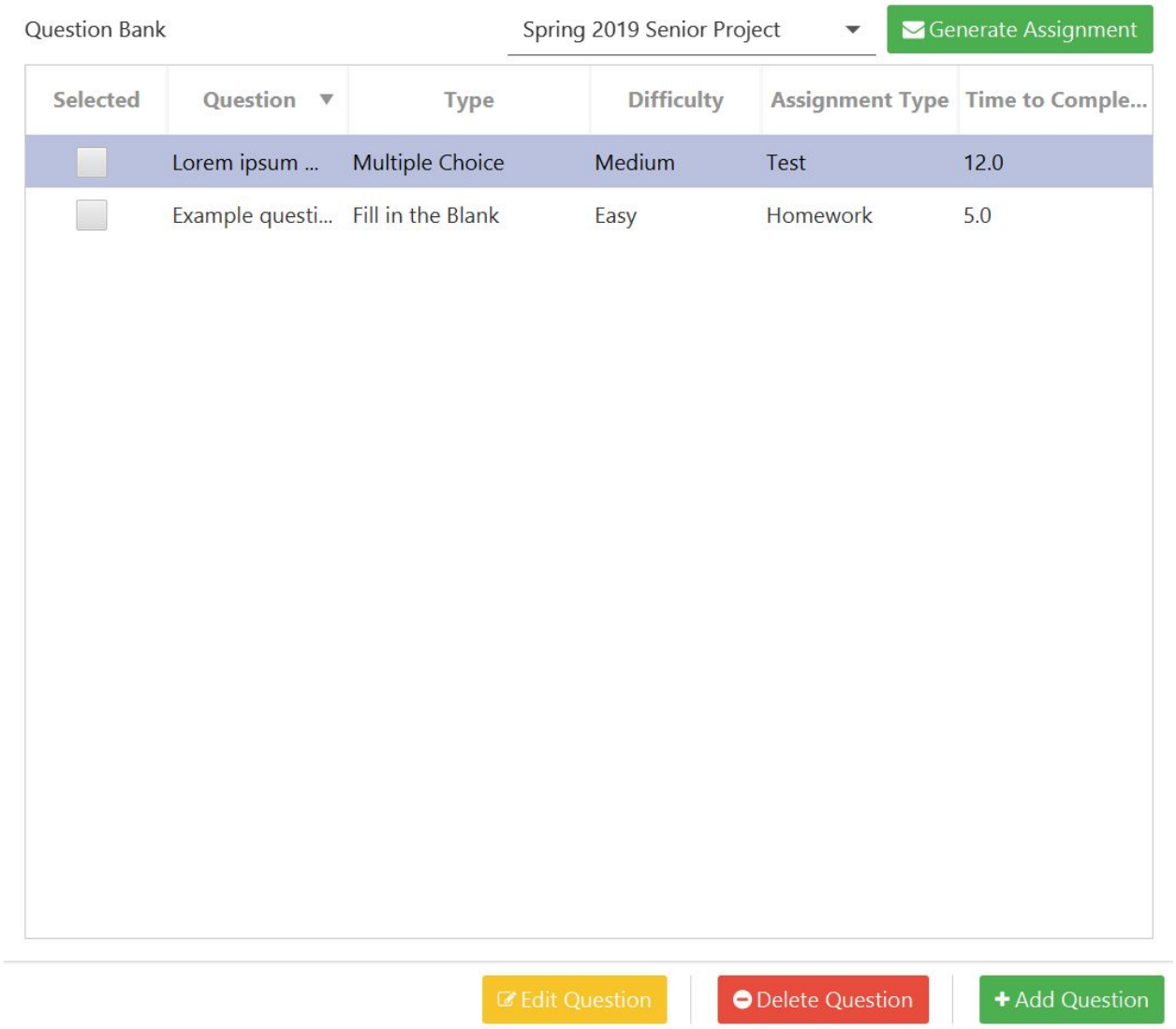

Figure 1: Question Bank Main View

The form view for adding and editing questions consisted of fields for entering the question text, possible answers for the question if any, and the other previously mentioned question fields. A combobox was added to allow the user to pick what kind of question is being added. If the user picks “multiple choice,” fields for entering answers into the form are added as well as buttons for adding answers and deleting answers. A screenshot of the form view is shown in Figure 2.

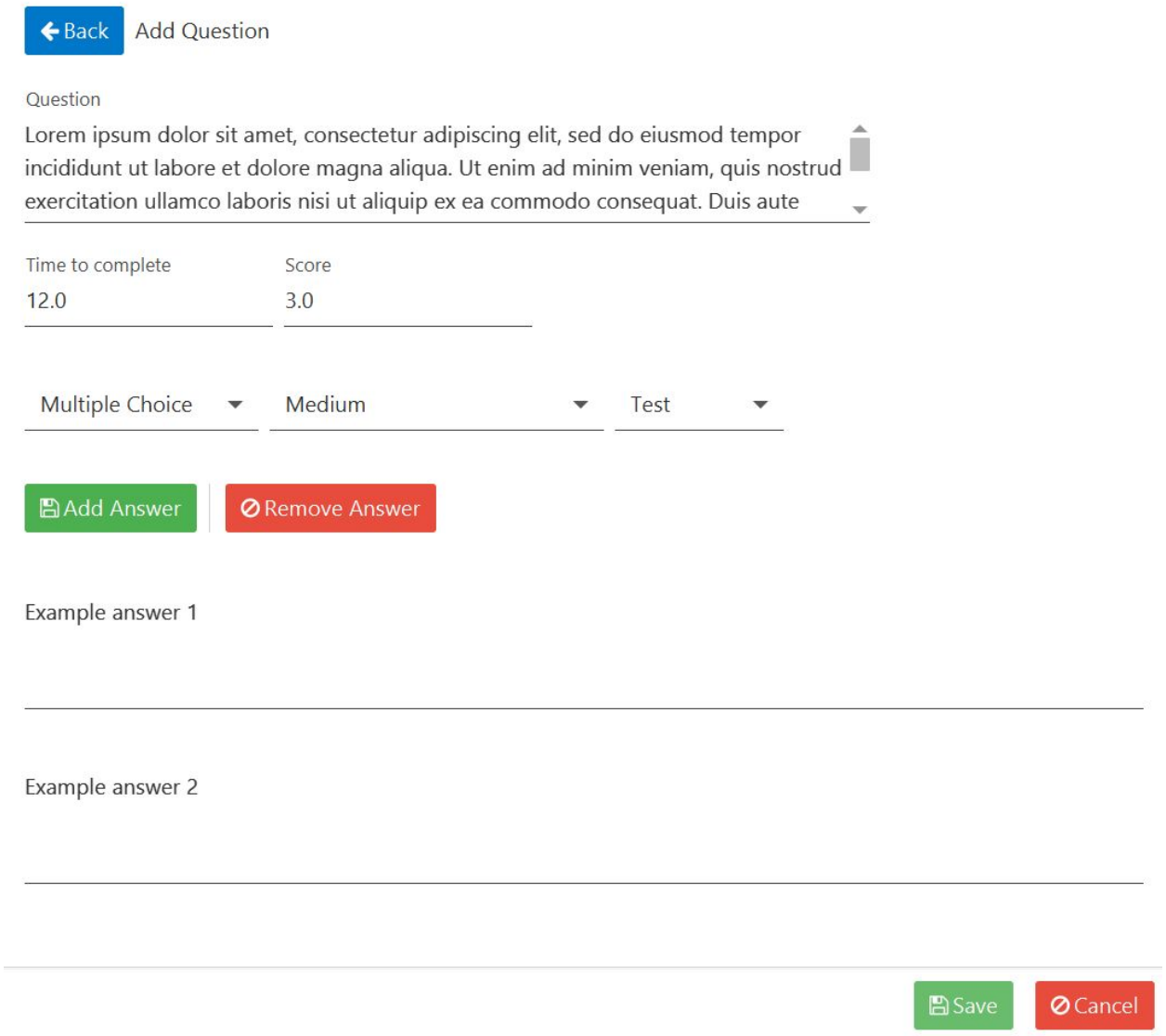

Figure 2: Question Form View

The view for generating an assignment allowed the user to choose to associate the questions with either a new assignment that is then added to the course or an existing assignment that is already in the course. The program would then show a dialog box for printing the generated assignment. After printing the assignment, the user would be taken to the view for the course so that the user could edit the assignment that the questions were associated with. Screenshots of the generate assignment view are shown in Figure 3 and Figure 4.

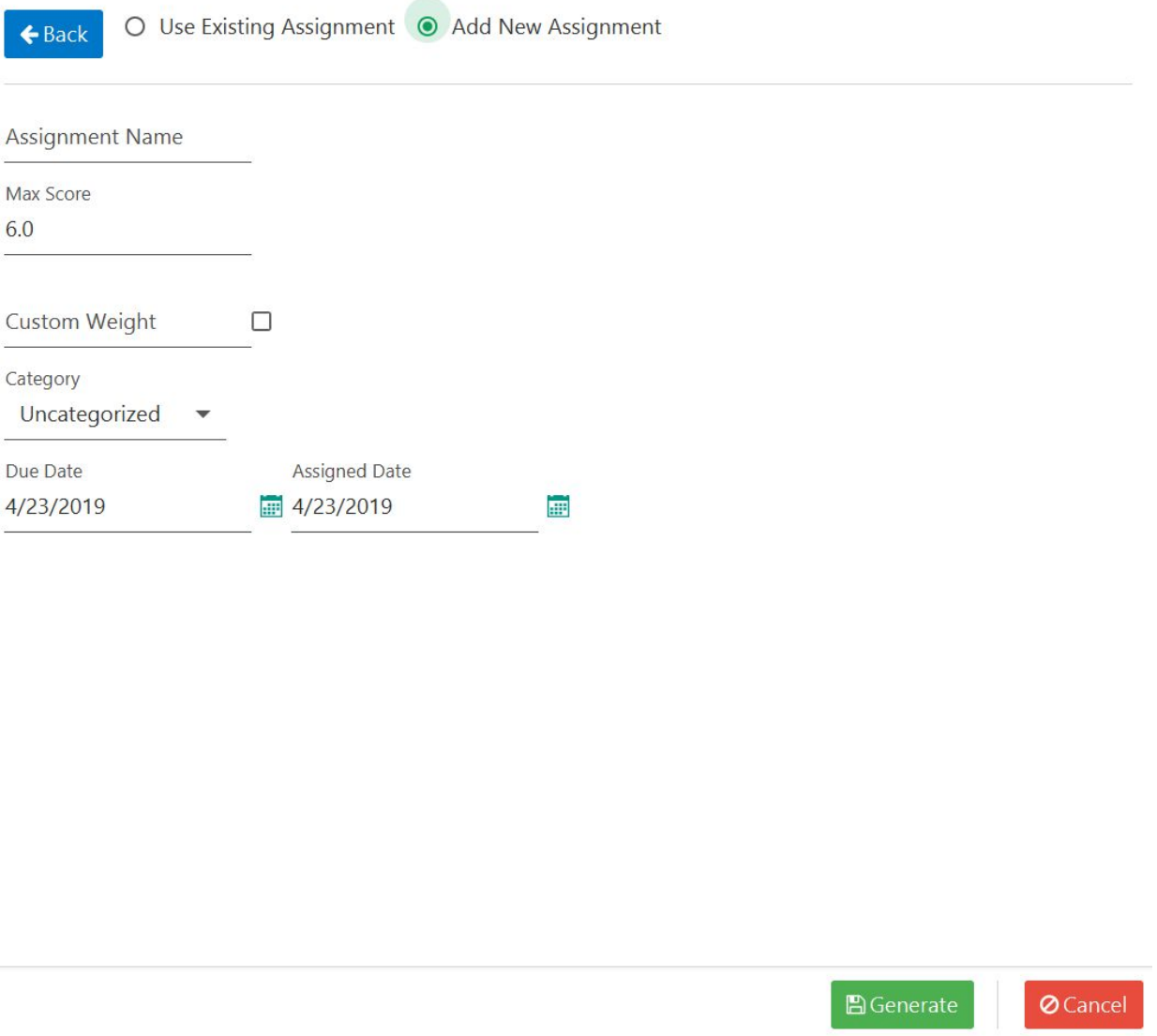

Figure 3: Generate Assignment View

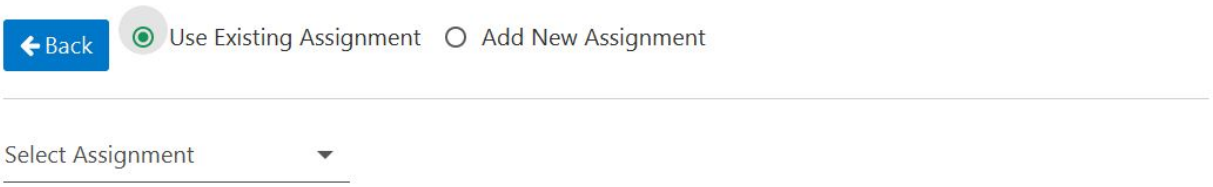

Figure 4: Generate Assignment View

A view was added for the email service that allows the user to enter an email address and a password for the email address. Controls were added to course and student views from the basic requirements of the project that allow the user to use the email and export services. Additionally, dialog boxes were added to notify the user of any errors with the email service

whenever it is used. A screenshot of the settings view for entering an email and password is shown in Figure 5.

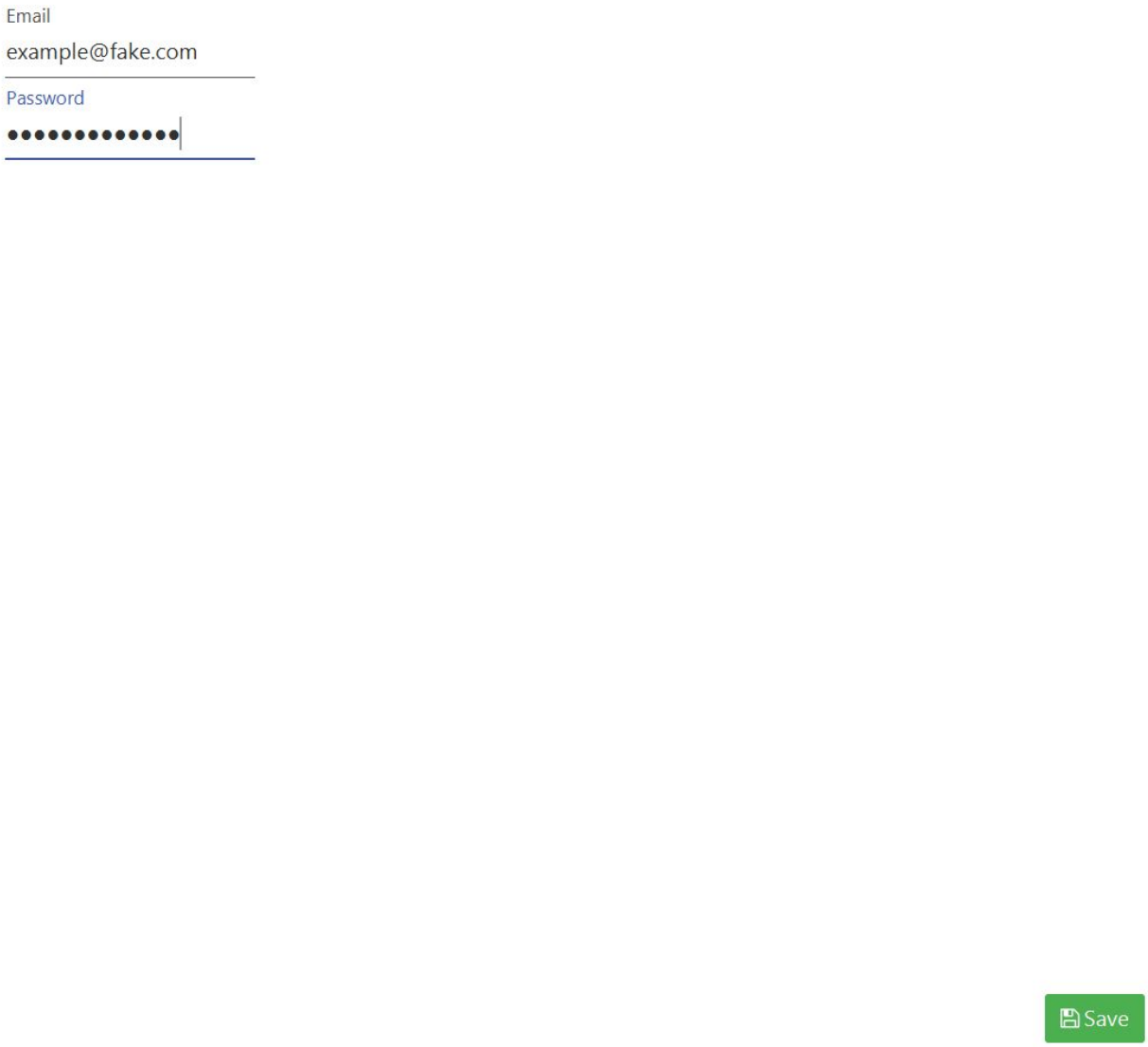

Figure 5: Settings View

Process

Question Bank

During the process of creating the program, several changes were made to the design of the question bank. Due to the team's inexperience with databases, the data type for storing the question text and answer text was poorly chosen. Initially, the table used a "Varchar" field to store the question and answer texts, but it was found to perform poorly with longer questions and answers. Additionally, the appropriate method of storing a variable number of answers per question was difficult to decide on. The initial decision was to use a "Varchar" field that separated answers using some delimiter, but this method required that whatever character was chosen to delimit answers could not appear in the text of an answer. The delimited string solution would also require much more string processing than desired and have the same issue that the lengthy question text field would have. While researching how to fix these issues, the honors members of the group found that the H2 database system supported fields of "Text" and "Array" types. The "Text" type was found to improve the performance of larger "Varchar" fields, and the "Array" type would solve the issue of variable number of fields for each question. The reason that these data types were not known about is due largely to the team's inexperience with the H2 database system and insufficient research time before designing the database.

Emailing Grade Reports

The biggest problem with the email service that became apparent to the honors members of the team was that the grade reports being emailed were not necessarily secure. The email service sends a grade report PDF to the student's email, but if the student uses a library computer and forgets to sign out of the email, that PDF attachment could be viewed by anyone at the

computer. This vulnerability was eventually addressed by using password protected PDF files rather than normal PDF files. The consensus of the honors members of the team was that the most secure password for a student's file was that student's ID. For UAH, the student's ID would be the A number of the student, which should not be known by anyone other than the student and school staff. The grade reports were then changed to have the student's ID as the password for the PDF when emailing them. The password prompt for the PDF also indicates that the password is the student's ID so that students are aware of how to view their reports.

While implementing the email service, the honors members of the group decided that the program should be able to email all grade reports for a course at once. Even though this functionality was not specifically referenced in the honors requirements, the tedious nature of emailing each student in the class their grade report by clicking through views made it apparent that a quality of life feature was needed. A way to email each student in a course their own individual grade reports was added to make the email service useful to an instructor with a large class.

Exporting Grades

The most significant change that was made to the way that grades are exported for a course is that the team decided it might be useful for the instructor to export all information for a course to a spreadsheet document rather than just the grades of the course. Using OpenCSV, defining how to export a course and all of the information related to the course was simple and would take about the same amount of development time as implementing the export feature for the grades in the course. Since this feature that encompassed more than the honors requirement

cost a similar amount of time to implement as the honors requirement, the team implemented the ability to export all information for a course to a spreadsheet document.

Conclusion

Extensive research was carried out to understand the functionings of several external libraries. With a mature programming language such as Java, there are several competing libraries available and some are better at certain tasks and others at others. It was necessary to understand what the libraries had to offer and choose one which made the most sense. More research should have been put into the database system being used especially considering nobody on the team had any prior experience using embedded databases.

The honors requirements for the IGP were met and integrated with the rest of the project well. Even after meeting the requirements agreed upon, some quality of life additions were made to make using the email service, exporting service, and question bank more user friendly.

Appendix

Links to libraries used:

H2:
- https://www.h2database.com/html/main.html

JFoenix:
- http://www.jfoenix.com/documentation.html

FontAwesome:
- https://fontawesome.com/?from=io
- https://www.jensd.de/wordpress/?p=2588

PDFBox:
- http://www.pdfbox.org/
- https://www.tutorialspoint.com/pdfbox/

Javax.print:
- https://docs.oracle.com/javase/8/docs/api/index.html?javax/print/package-summary.html

OpenCSV:
- http://opencsv.sourceforge.net/
- https://www.baeldung.com/opencsv

JavaMail:
- https://www.oracle.com/technetwork/java/javamail/index.html
- https://www.tutorialspoint.com/javamail_api/javamail_api_gmail_smtp_server.htm

JUnit:
- https://junit.org/junit5/