Software Project Information File

PDF Grader

Version 1

Contents

1. REVISION HISTORY:

Rev.	*Date*	*Author*	*Comments*
1	4/12/2022		Initial release.

2. PURPOSE

Test grading is a time consuming task that professors have to go through. More and more often tests are being submitted through web based programs leading to digital files being graded. In order to more efficiently grade it is proposed that a piece of software can help a professor grade a PDF (scanned or digital) file while also speeding the process up by presenting the question in a more efficient order.

3. SCOPE

The program will take in one pdf file with multiple tests in it. This means, for a five page test, every five pages is the fifth/last page of each test. You will input how many pages are in a test, and how many questions there are. The professor will then go through each page and drag a box over each question/answer location for the entire test. Using that box, it goes question by question (all the 1s, then all the 2s, etc) of each test for grading.

The grading is not automatic, the professor will be shown the question that is being graded and have an interface where he can type specific feedback for a question and how many points they're going to lose to be reused on other students answering the same question. There will also be a variety of "canned" answers, but these will be very general. Out of bounds support is needed, so if an answer is trailing off the specified box, they can go to the entire page of that question and then return to the test grading software.

After an initial release more features may be added like allowing two or more people to run this software at the same time and thus be able to grade with a TA, as well as a barcode recognition to be able to automatically gather information.

Once the tests have been graded, the software will generate a table of scores as a pdf, and a pdf for each student. This pdf for the student will be as many pages as were questions in the test, and list their feedback that they received on each question (1 question per page).

4. REFERENCES

Software Development Process
Project Plan - PDF Grader.docx
SDD - PDF Grader.docx

5. CODING STANDARD

Java Files
Location: src/main/java/com/ezgrader/pdfgrader/
Name Convention: CapitolCase, scene controllers use format [scene name]`Controller.java`
Variable Convention: camelCase

FXML Files
Location: src/target/classes/com/ezgrader/pdfgrader/
Name Convention: lowercase, in format [scene name]`.fxml`
Tag ID Convention: camelCase

CSS Files
Location: src/main/java/com/ezgrader/pdfgrader/
Name Convention: lowercase, in format [scene name]`.css`

Style Class / ID Convention: camelCase (as opposed to typical dash-case to match w/ controller variable names)

6. SOFTWARE PROJECT REPOSITORY LOCATION

Description: The code repository is on a private github, where each team member is a collaborator on the project.
Location: https://github.com/colbehr/PDF-Grader
Login: N/a

Version Control Workflow File Name: Source_Code_Control_Workflow.docx
Version Control Workflow File Location:

7. SOFTWARE DESIGN NAME and LOCATION

Functional Requirements

File Name: Section 2.3 of *Project Plan - PDF Grader.docx*
File Location: /documents/*Project Plan - PDF Grader.docx*

Specification

File Name: SDD - PDF Grader.docx
File Location: /documents/SDD - PDF Grader.docx

8. SOFTWARE SYSTEM TEST PLAN NAME and LOCATION

As of right now we have not implemented a test plan. The document will be updated as soon as a testing framework is created and implemented.

However, at the end of the Spring ‘22 quarter, we plan to do practical testing with our client, Aran Clauson, by grading real student exams to evaluate the workflow of the software. This will also test all of the basic essential functionalities of the software. This includes importing a test, setting up questions, entering scores and feedback, and finally exporting graded tests and test overview. This real-world usage will establish a baseline for what features need further development and testing.

File Name: N/a
File Location: N/a

9. SOFTWARE BUILD INSTRUCTIONS

Required Build Tools

- IntelliJ IDEA (or other IDE)
- Maven
  - Maven should download all needed libraries, but if manually downloading, the libraries are:
    - JavaFX
    - Apache PDFbox

Build Instructions

- Add the repo and clone the project using File>New>Project From Version Control or using the Get from VCS button on the home page.
    - This should auto populate if you log into Github.
- Add a new 'Application configuration under Run>Edit Configurations... or at the top right under Add Configuration...
- Confirm you are using Java 17 and choose a main class using the icon on the right, should be com.ezgrader.pdfgrader.Main
    - The main class should auto populate, if it does not there may be a problem.
- Add a name at the top and hit Apply or OK.
- The project should now run.

10. PROGRAMMING THE DEVICE

N/a

Required Software

N/a

Required Hardware

N/a