Project Backlog

ADVANCED SUBMISSION SYSTEM

Anthony Goeckner; Saurav Khanna; Max Molnar; Krutarth Rao; Harold Smith
CS 40800 TEAM 15 | PURDUE UNIVERSITY

Problem Statement

The “turnin” system used by many Computer Science courses is awkward to use and poorly documented. We propose a system that is easier to use and has more clearly visible warnings and safeguards preventing accidental loss of data or lack of submission. Additionally, the current system does not provide a way to view grades of submitted assignments. Therefore, we will also develop two interfaces: one for students to view their own grades and statistics, and another for instructors to edit grades and view class statistics. This new, combined system will be highly popular with both students and instructors.

Background Information

Using the current “turnin” system, mistakes can easily be made that result in the loss of data. This includes overwriting an existing submission with an empty one. It is also relatively easy to submit the wrong files or to submit files to the wrong destination (e.g. to submit a file to the wrong project). All control of the program is done using the command line, and the printed output is difficult to interpret. Most computer science majors use the turnin system at some point during their academic career, and for many it is one of their first experiences with command-line tools. Therefore, an easier-to-use system is required.

Development Environment

Our "turnin" system will be written in C++, and will be compiled with GCC. The files will be stored in relevant folders in Purdue's filesystem, to alleviate the need for an external database. Since the system will involve a command-line user interface, it will require the ncurses library. We will also need a JSON parser for config files, the Linux tar program, and a library for file storage/management.

Functional Requirements

Backlog ID	Functional Requirement	Hours	Status
1	As a student, I would like to submit a file or folder.	6	Completed in Sprint 2.
2	As a student, I would like to check what I have already submitted.	4	Completed in Sprint 2.
3	As a student, I would like to be warned if I am submitting an assignment when there is already a previous submission.	4	Completed in Sprint 2.
4	As a student, I would like to be warned if I am trying to submit an empty file.	6	If time permits.
5	As a student, I would like to be warned if I am submitting an assignment to the wrong destination.	5	Completed in Sprint 1.
6	As a student, I would like to check the grades for my submissions.	14	Completed in Sprint 2.

7	As a student, I would like to check the feedbacks for my submissions.	7	Completed in Sprint 2.
8	As a student, I would like to check the grade statistics for each assignment.	16	If time permits.
9	As a student, I would like to submit team assignments.	13	If time permits.
10	As a student, I would like to submit regrade email requests.	7	If time permits.
11	As a professor, I would like to create an assignment.	16	Completed in Sprint 2.
12	As a professor, I would like to delete an assignment.	13	Completed in Sprint 2.
13	As a professor, I would like to add a limit to the number of times an assignment can be submitted.	7	Completed in Sprint 2.
14	As a professor, I would like to add a date past which the assignment will not be accepted.	7	Completed in Sprint 2.
15	As a professor, I would like to view students' submissions from a class.	12	If time permits.
16	As a professor, I would like to view assignment submission times.	10	If time permits.
17	As a professor, I would like to edit students' grades for each assignment from a class.	12	Completed in Sprint 2.
18	As a professor I would like to view team assignments.	15	If time permits.
19	As a professor, I would like to grade team assignments.	6	If time permits.
20	As a professor, I would like to have access to submissions and grades from multiple classes.	6	Completed in Sprint 2.
21	As a professor, I would like to provide feedback for each assignment submitted.	6	Completed in Sprint 2.
22	As a professor, I would like to view the statistics for each assignment from a class.	10	If time permits.
	Total:	202	

Non-Functional Requirements

The requirements below focus on how the system will function.

Backlog ID	Functional Requirement
1	As a developer, I would like the code to be well documented and commented so that others can add functionality to the program quickly.
2	As a developer, I would like the system to be modular to allow for white box testing techniques and unit testing.

3	As the developer, I would like the program to hold student's assignments in a secure format.
4	As a developer, I would like the system to provide files for grading to instructors in less than 2 seconds.
5	As a developer, I would like the program to be in the Python language.
6	As a developer, I would like the program to facilitate team submissions and team grading.
7	As a Customer, I would like to grade assignments in a user friendly interface.
8	As a Customer, I would like to have access to the portal that is responsive and free of lag.
9	As a Customer, I would like to have a secure way to login to the grading portal.
10	As a Customer, I would the changes made in the grades visible to students in less than 5 seconds.
11	As a Customer, I would like the users to read the graders comments in an organized format.
12	As a User, I would like the program to give me helpful hints while submitting the assignment if I am new to the system.
13	As a Customer, I would like the system to store grades in a secure location only visible to the student and the instructor.
14	As a Customer, I would like to recheck my assignments and open a read-only version on a later date in case of an error during grading.
15	As a developer, I would like the program to run in debug mode and output the necessary meta data to identify errors.
16	As a developer, I would like the program to compile to an executable that can undergo black-box testing.

Use Cases

Student Use Cases

Case: Viewing Grades

Action:

1. Enter class folder
2. Open grades.txt

System Response:

1. Changes directory to the class directory and opens the student's personal folder
2. Opens the grades text file and displays it on the terminal

Case: Uploading assignments

Action:

1. Make a directory and put all files for submission in that directory
2. Enter command to submit directory for grading

System Response:

1. A new directory is made and all files for submission are added to it
2. A message is display either indicating success or failure

Case: Overriding previous submissions with a new submission

Action:

1. Make a directory and put all files for submission in that directory
2. Enter command to submit directory for grading
3. Enter "y" or "yes" to confirm overriding the previous submission

System Response:

1. A new directory is made and all files for submission are added to it
2. Ask for confirmation to override the previous submission
3. A message is display either indicating success or failure

Case: Request a regrade on an assignment

Action:

1. Enter the command to request a regrade for an assignment
2. Enter "y" or "yes" to confirm sending a regrade request

System Response:

1. Display message asking for confirmation to submit a regrade request for the specified assignment
2. Display a message either indicating success or failure

Professor Use Cases

Case: Adding projects, labs, and homework for individual assignments

Action:

1. Change directory to class directory
2. Type command to make a new individual assignment directory

System Response:

1. Class directory is opened and displays current assignment directories
2. A new directory is created for the assignment
3. Inside the new directory, a directory is created for each student in the class
4. A message is displayed indicating success or failure in creating the new assignment

Case: Adding projects, labs, and homework for team assignments

Action:

1. Change directory to class directory
2. Type command to make a new team assignment directory

System Response:

1. Class directory is opened and displays current assignment directories
2. A new directory is created for the assignment
3. Inside the new directory, a directory is created for each team listed in a provided list of teams
4. A message is displayed indicating success or failure in creating the new assignment

Case: Deleting projects, labs, and homework assignments

Action:

1. Change directory to class directory
2. Type command to delete an existing assignment
3. Type "y" or "yes" to confirm deleting the assignment

System Response:

1. Class directory is opened and displays current assignment directories
2. A message is displayed to confirm deleting the assignment
3. The directory is deleted and any scores listed in statistics are removed

Case: Modifying and assigning grades

Action:

1. Change directory to class directory
2. Change directory to the assignment directory
3. Change directory to student or team directory
4. Type command to grade and assignment

System Response:

1. Class directory is opened and displays current assignment directories

2. The assignment directory is opened and displays all the student or team folders
3. The team or student folder is opened and displays all the files submitted
4. A new text file is created called "grade.txt" that contains the grade entered for the assignment
5. A check is run to see if all student or team directories have a grade.txt file
6. If all directories contain a grade.txt then each grade.txt is updated with class statistics for the assignment

Case: Modifying grades after initial grade

Action:

1. Change directory to class directory
2. Change directory to the assignment directory
3. Change directory to student or team directory
4. Type command to modify and existing grade

System Response:

1. Class directory is opened and displays current assignment directories
2. The assignment directory is opened and displays all the student or team folders
3. The team or student folder is opened and displays all the files submitted
4. The text file with the grade is updated
5. A check is run to see if all student or team directories have a grade.txt file
6. If all directories contain a grade.txt then each grade.txt is updated with class statistics for the assignment

Case: Looking up student grades

Action:

1. Change directory to class directory
2. Type command to lookup a grade for a specific student or team

System Response:

1. Class directory is opened and displays current assignment directories
2. Enters the student or team directory and displays the grade in grade.txt
3. If no grade.txt exists then a message is displayed indicating no grade has been assigned