Doornail Technologies, Inc.

System Requirement Specifications for the Grade Book System

Aaron Danielson | Luis Hidalgo | Timothy Jutras | Graham Williamson

CS3300 Software Engineering

12.03.2012

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Release Date | Responsible Party | Major Changes |
| 1.0 | 12.03.2012 |  | Release for design |

Table of Contents

[1.0 Introduction 3](#_Toc319178975)

[1.1 Purpose 3](#_Toc319178976)

[1.2 Scope 3](#_Toc319178977)

[1.3 Definitions, Acronyms, and Abbreviations 3](#_Toc319178978)

[2.0 General Description 3](#_Toc319178979)

[2.1 Product Perspective 3](#_Toc319178980)

[2.2 Product Function Summary 3](#_Toc319178981)

[2.3 User Characteristics 4](#_Toc319178982)

[2.4 General Constraints 4](#_Toc319178983)

[2.5 Assumptions and Dependencies 4](#_Toc319178984)

[3.0 Specific Requirements 4](#_Toc319178985)

[3.1 Functional Requirements 4](#_Toc319178986)

[3.2 External Interface Requirements 5](#_Toc319178987)

[3.3 Use Case Descriptions 5](#_Toc319178988)

[3.4 Performance Requirements 5](#_Toc319178989)

[3.5 Design Constraints 5](#_Toc319178990)

[3.6 Quality Characteristics 5](#_Toc319178991)

[4.0 Supporting Information 6](#_Toc319178992)

[4.1 Appendix A 6](#_Toc319178993)

1. **Introduction**
   1. **Purpose**

The purpose of this document is to clarify and specify the capabilities of the grade book program known as "Dream Crusher". It will provide specific information regarding the functionality and typical use cases, and will be used by the developers as a guide to maintain a common end product goal.

* 1. **Scope**

The grade book application will offer the user a graphical user interface in order to perform the following system functionalities:

* + 1. Manage one or more grade books for multiple classes, which may contain a variable number of students and assignments.
    2. A grade book must be able to support zero or more assignments; the user may add class assignments at any time.
    3. Save all data in CSV files to allow displaying data as spreadsheets in Excel.

A grade book allows an instructor to manage the individual student scores for class assignments, for which all the students are expected to complete.

* 1. **Definitions, Acronyms, and Abbreviations**
     1. CSV Comma-separated values. This is a file format used to store tabular data in plain-text form. Entries are separated by commas. Typically, a CSV file is translated into a spreadsheet of rows and columns and is used in a program like Excel.
     2. Excel Microsoft Office Excel Program.
     3. GUI Graphical User Interface. This is the interface between the user and the software that will allow the user to tell the program what to do and to see the results displayed to the monitor.
     4. OS Operating System.
     5. SRS System Requirements Specification. This is used in reference to this specific document as a whole.

1. **General Description**
   1. **Product Perspective**

Our product is independent and totally self-contained. It is not a component of a larger product, and no databases will be accessed or needed for our product to function. The program manages its own functions of reading and saving CSV files, allowing the user to enter data manually, and exporting the data via CSV files for use in Excel.

* 1. **Product Function Summary**

The purpose of our product is to prepare a simple grade book layout for use in Excel. Dream Crusher will allow the user to import a CSV file containing a list of student names that will be distributed in a single column. The user can enter in assignment or exam titles as separate columns and can also enter formulas (as separate columns) for use in calculating the student grades. These formulas can be copied into each row containing a student name. Once the user has finished filling in the assignment titles and/or formulas, the program will save the data as a CSV file. The user can then open this CSV file in Excel, and the data will be spread into the appropriate columns.

The end result of the layout should contain a column with the student names, additional columns with the assignments and exams, and some formula columns for calculating the student grades. The assignment and exam columns will be left blank (except for their titles); no data will be written into these columns (i.e. no grades will be entered) via this program. Rather, these columns will receive data in Excel. However, the program is responsible for copying a formula from a formula column into each row containing a student name. The sole purpose of the program is to prepare a simple layout to be used in Excel, not to store data such as student grades.

* 1. **User Characteristics**

The grade book user will require only minimal computer literacy skills with the Microsoft Windows line of operating systems. This includes mouse and keyboard usage, starting and exiting a program, and selecting files from a directory. The program will also provide specific details on how to use the grade book application via the help menu.

* 1. **General Constraints**
     1. The grade book application shall execute on all standard PC platforms running Microsoft Windows versions XP through Windows 7.
     2. The program shall not allow multiple instances of itself to run. This is done in order to ensure that no conflicts exist when exporting and saving data from the application.
     3. The program shall be written in the C# 4.0 NET framework, produced by Microsoft.
     4. The program shall be distributed as an install file, which can easily be run by double-clicking the file.
  2. **Assumptions and Dependencies**
     1. The professor is the only person who administers the grade book application; there is no student involvement.
     2. Data encryption and decryption techniques are not required since a password protected grade book system is not needed.

1. **Specific Requirements**
   1. **Functional Requirements**
      1. The program will take a CSV file as input, which contains the student roster, and distribute the student names in a single column.
      2. The program will allow the user to create columns for assignments and exams, which will eventually contain the individual student scores. The scores will be entered through Excel, not the program (i.e. Dream Crusher).
      3. The program will not allow the user to enter the student grades for each assignment or exam. These cells will be left blank.
      4. The assignments and exams will not be imported as a list into the application (i.e. via a CSV file). They must be entered manually through the programs prompting.
      5. Assignment and exam titles are editable. The user can delete or change their titles.
      6. The program will allow the user to create columns for arithmetic formulas to calculate the student overall scores and/or calculate weighted assignments.
      7. The formulas will not be imported as a list into the application (i.e. via a CSV file). They must be entered manually through the programs prompting.
      8. The arithmetic formulas can be copied into each row containing a student name.
      9. The arithmetic formulas are editable. The user can delete or change the formulas.
      10. The program will save the data created by the user into a CSV file and store the CSV file into a user-specified directory.
      11. When the user saves the data to a CSV file, the program will prompt the user to name the CSV file.
      12. The program can open a previously created grade book (i.e. a CSV file) from the directory and allow the user to edit any of the grade book assignment, exam, or formula columns.
      13. The program will allow the user to open a CSV file containing a student roster and merge this file with another opened CSV file containing the layout of a previously created grade book.
      14. The program will include an option, which will allow the two CSV files, explained in requirement 3.1.13, to be merged together. A potential name for this is GENERATE.
      15. A CSV file containing a grade book layout and a student name CSV file must both be opened in order for the option, mentioned in requirement 3.1.14 to function.
      16. The application must display the name of the CSV files currently in use.
   2. **External Interface Requirements**
      1. The program will contain a GUI that will be the mediator between the user and the program. It will allow the user to give commands to the program and will display the contents of the grade book to the monitor.
      2. The GUI will contain all of the options which allow the user to manipulate or interact with the grade book application. These options will be accessible via a menu strip.
      3. The GUI will contain an option to exit the application.
   3. **Use Case Descriptions**
      1. Import the CSV-formatted student roster into the grade book application.
      2. Import an existing CSV-formatted grade book layout into the application.
      3. Create one or more new class assignments.
      4. Create one or more formula columns.
      5. Save the data to a new CSV file.
      6. Modify any existing formula columns.

Refer to Appendix A (4.1) for more information.

* 1. **Performance Requirements**
     1. The grade book application must load within 10 seconds after the initial startup.
     2. The initial importing of the CSV file containing the list of student names must finish within 10 seconds.
     3. Saving the data to a new CSV file must finish within 10 seconds.
  2. **Design Constraints**

The design of the grade book program must conform to the following constraints:

* + 1. All external files for use with the application will be stored as a CSV file.
    2. The design will be as object-oriented, in order to strive for easier maintainability.
  1. **Quality Characteristics**

The grade book will be designed and implemented according to the following quality standards:

* + 1. Formatting - The program will ensure that the CSV files are correctly formatted, and can be imported into Microsoft Excel without error. This is critical because the user should not be forced into personally scanning for formatting errors.
    2. Reliability - The program will be designed to be robust, minimizing any actions or events which could cause any crashes or loss of data.
    3. The program will adhere to the performance requirements mentioned in the performance section of this document.

1. **Supporting Information**
   1. **Appendix A: Use-Case Descriptions**
      1. Function: Import Student Roster
2. Actors: User
3. Description: The user will import the CSV-formatted student roster into the grade book application.
4. Pre-Conditions: The user has a CSV file containing a roster of student names, and has launched the program.
5. Post-Conditions: The user will create assignments and formulas for the students in the grade book.
6. Import Flow:
   1. User selects an option in the File menu to open a CSV roster.
   2. A directory window will pop up, prompting the user to select the CSV file to import.
   3. The user selects ‘Open’ and the program will import the student names to the form accordingly.
      1. Function: Import Existing Grade Book Layout
7. Actors: User
8. Description: The user will import an existing CSV-formatted grade book layout into the application.
9. Pre-Conditions: The user has an existing .CSV file which represents the grade book layout consisting of assignment and formula columns.
10. Post-Conditions: The user will have merged the student roster CSV file with the grade book layout CSV file.
11. Import Flow:
    1. User selects an option in the File menu to open a CSV grade book template.
    2. A directory window will pop up, prompting the user to select the CSV file to import.
    3. The user selects ‘Open’ and the program will import the grade book layout.
    4. The user will select an option to ‘generate’ another CSV file, which will merge the student roster CSV file with the grade book layout CSV file.
       1. Function: Create New Class Assignments
12. Actors: User
13. Description: The user will create new class assignments.
14. Pre-Conditions: The user has an open grade book.
15. Post-Conditions: The user will either create assignments for the assignment column or save existing data as new CSV file.
16. Create New Assignment Flow:
    1. User selects option to ‘Create New Column’.
    2. The user is prompted with a ‘Create New Column’ wizard.
    3. The user will select the option ‘Data Entry Column’.
    4. User completes the wizard with the appropriate assignment information.
    5. User selects ‘Finish’ in the wizard, and the form is populated accordingly.
       1. Function: Create New Formula Column
17. Actors: User
18. Description: Create one or more formula columns to represent scores.
19. Pre-Conditions: User must have an open grade book and one or more assignment columns.
20. Post-Conditions: User will save all data to a CSV file.
21. Create New Formula Flow:
    1. User selects option to ‘Create new Column’.
    2. The user will be prompted with the ‘Create New Column’ wizard.
    3. The user will select the option ‘Formula Column’.
    4. User completes the wizard with the appropriate information, including the formula to be used.
    5. User selects ‘Finish’ and the form is updated accordingly.
       1. Function: Saving Data to a CSV File
22. Actors: User
23. Description: Save all the current data as a CSV file.
24. Pre-Conditions: User must have an open grade book with one or more assignments and/or formula columns.
25. Post-Conditions: User will open the saved CSV file with Excel.
26. Saving Data to a CSV File Flow:
    1. User selects option to ‘Save’.
    2. A ‘Save As’ window will appear.
    3. User is prompted for the name and location of the file to be saved.
    4. User selects ‘Save’, and the file is saved to the chosen directory.
       1. Function: Editing an Existing Formula Column
27. Actors: User
28. Description: Editing the formula on an already existing formula column.
29. Pre-Conditions: User must have an open grade book with one or more assignments and at least one formula column.
30. Post-Conditions: User will save the file.
31. Editing an Existing Formula Column Flow:
    1. User selects option to ‘Modify Formula’.
    2. A ‘Modify formula’ wizard will appear.
    3. The wizard will prompt users to identify the formula column to be changed.
    4. The wizard will prompt for the new formula to be entered.
    5. User selects ‘Update’.
    6. New formula is displayed in the form accordingly.