Software Requirements Specification

February 28, 2020

Team 3-- Owen, Abby, Katie, Ana, Brad, Ben, and Gracie

Introduction:

This report describes Team 3's Software Requirements for its project to develop a mastery grade book dashboard for Davidson College.

The format will follow the format shown in the textbook, Chapter 3, Section 3.3

1. **Introduction**

   1.1. Purpose

   The purpose of our application is to provide a dashboard that allows professors to input grade data for students and allows students to view their individual grades.

   1.2. Scope

   This document defines requirements for the entire gradebook dashboard system, including the backend database and front end web applications. Because our team is using the Agile method, we will be defining most but not all requirements in this document. We plan to add requirements as necessary throughout the lifecycle of this project.

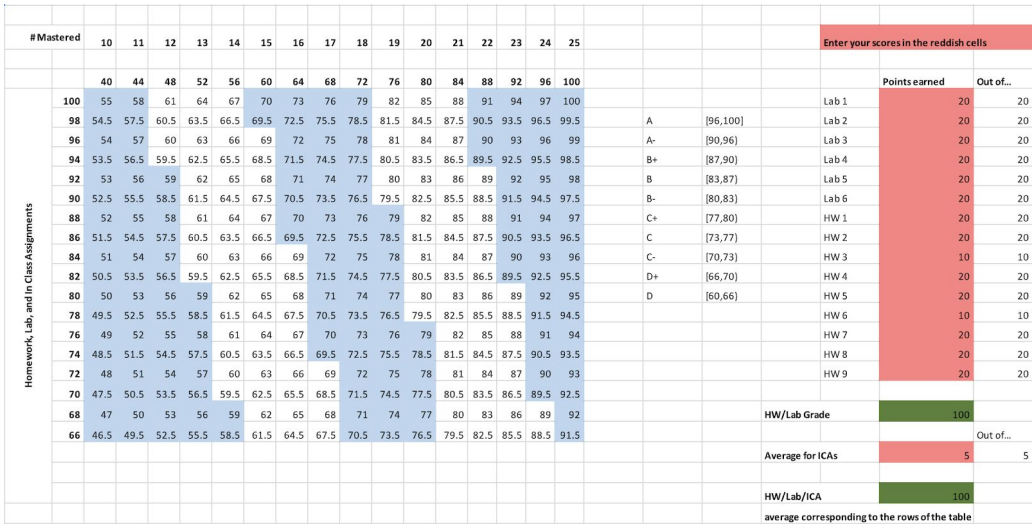
   1.3. Definitions, Acronyms, and Abbreviations

   - Mastery Grading System (MGS): A grading system in which students must achieve a complete understanding of a topic before receiving full credit for that topic. The final grade is determined by the number mastered over the total number of concepts in the class.
   - At Davidson, the mastery levels are M (master), J (journeyman), and A (apprentice). A master has correctly completed an assessment on a particular topic. A journeyman has attempted a mastery, but missed a few small things. Finally, an apprentice has attempted a topic, but did not make significant progress.

   1.4. References

   A basic definition of the MGS: **https://en.wikipedia.org/wiki/Mastery_learning**

An example MGS gradebook:



#Mastered	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
	40	44	48	52	56	60	64	68	72	76	80	84	88	92	96	100
100	55	58	61	64	67	70	73	76	79	82	85	88	91	94	97	100
98	54.5	57.5	60.5	63.5	66.5	69.5	72.5	75.5	78.5	81.5	84.5	87.5	90.5	93.5	96.5	99.5
96	54	57	60	63	66	69	72	75	78	81	84	87	90	93	96	99
94	53.5	56.5	59.5	62.5	65.5	68.5	71.5	74.5	77.5	80.5	83.5	86.5	89.5	92.5	95.5	98.5
92	53	56	59	62	65	68	71	74	77	80	83	86	89	92	95	98
90	52.5	55.5	58.5	61.5	64.5	67.5	70.5	73.5	76.5	79.5	82.5	85.5	88.5	91.5	94.5	97.5
88	52	55	58	61	64	67	70	73	76	79	82	85	88	91	94	97
86	51.5	54.5	57.5	60.5	63.5	66.5	69.5	72.5	75.5	78.5	81.5	84.5	87.5	90.5	93.5	96.5
84	51	54	57	60	63	66	69	72	75	78	81	84	87	90	93	96
82	50.5	53.5	56.5	59.5	62.5	65.5	68.5	71.5	74.5	77.5	80.5	83.5	86.5	89.5	92.5	95.5
80	50	53	56	59	62	65	68	71	74	77	80	83	86	89	92	95
78	49.5	52.5	55.5	58.5	61.5	64.5	67.5	70.5	73.5	76.5	79.5	82.5	85.5	88.5	91.5	94.5
76	49	52	55	58	61	64	67	70	73	76	79	82	85	88	91	94
74	48.5	51.5	54.5	57.5	60.5	63.5	66.5	69.5	72.5	75.5	78.5	81.5	84.5	87.5	90.5	93.5
72	48	51	54	57	60	63	66	69	72	75	78	81	84	87	90	93
70	47.5	50.5	53.5	56.5	59.5	62.5	65.5	68.5	71.5	74.5	77.5	80.5	83.5	86.5	89.5	92.5
68	47	50	53	56	59	62	65	68	71	74	77	80	83	86	89	92
66	46.5	49.5	52.5	55.5	58.5	61.5	64.5	67.5	70.5	73.5	76.5	79.5	82.5	85.5	88.5	91.5

Homework, Lab, and In Class Assignments

A	[96,100]	
A-	[90,96)	
B+	[87,90)	
B	[83,87)	
B-	[80,83)	
C+	[77,80)	
C	[73,77)	
C-	[70,73)	
D+	[66,70)	
D	[60,66)	

Enter your scores in the reddish cells

	Points earned	Out of...
Lab 1	20	20
Lab 2	20	20
Lab 3	20	20
Lab 4	20	20
Lab 5	20	20
Lab 6	20	20
HW 1	20	20
HW 2	20	20
HW 3	10	10
HW 4	20	20
HW 5	20	20
HW 6	10	10
HW 7	20	20
HW 8	20	20
HW 9	20	20
HW/Lab Grade	100	
		Out of...
Average for ICAs	5	5
HW/Lab/ICA	100	

average corresponding to the rows of the table

1.5. Overview

Overall, we hope to make a dynamic web application for both professors and students to use. That being said, our application will consist of two main interfaces: one for students and one for professors.

For the students, we need our application to display their grades and their progress in the course. In the MGS, it can be complicated for a student to keep track of their progress on each individual topic. Therefore, we aim to solve this problem for students through our application.

On the other hand, our application will allow professors to create a gradebook for their courses. The professors can define topics and enter grades for each student. This way, professors can monitor each student's progress throughout the course.

2. Overall Description

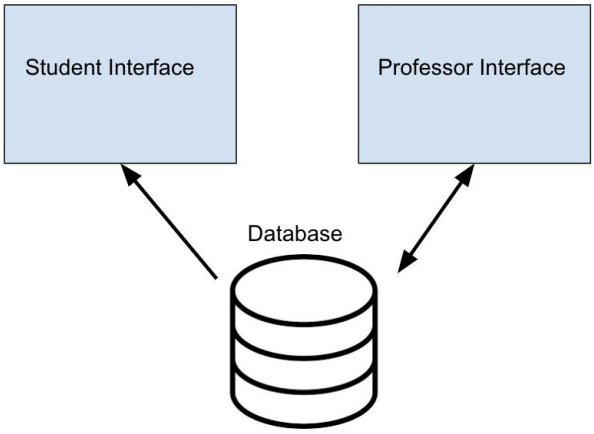
2.1. Product Perspective

Lucky for us, our product will be developed as a stand-alone application and it does not need to fit into an existing code base or greater system. Additionally, we require no special hardware to implement or use this product. That is, our product is quite simple from both a system and hardware perspective.

From a user perspective, this product will simplify a difficult task. From a student's perspective, this product will be a tool that solves the problem of keeping track of their grades in a MGS course. From a professor's perspective, this product will make it easier for them to teach a MSG course.

From a network perspective, our product will rely on a database hosted on a server. The users will act as clients and connect to our server to read from/write to the database.

2.2. Product Functions



We have three major pieces of the system:

1. the student interface
2. the professor interface
3. the database

Both of the interfaces will interact with the database. The professor side of the application will read and write to the database. However, the student side will only read from the database. The professor interface and student interface will exchange data through our database.

2.3. User Characteristics

We have two main types of users: teachers/professors and students. Our target student users are those enrolled in a course that uses the MGS. The students are aware of the MGS and want to actively track their progress in the course. The teachers/professors using our application have an understanding of MGS and a list of concepts they want to track. Our application will not help the professors/teachers design their course, but will help them track grades for an already planned course.

2.4. General Constraints

Because we are all students here at Davidson, we are constrained by a lack of technical resources. That is, we will not be hosting a centralized server with a database that anyone can connect to in order to use our application. However,

we hope to have a product that is accessible to professors and students within Davidson College. To do this, we will talk to Dr. Mendes about setting up a database server to store all of our data. This database is constrained by the resources of the Davidson Computer Science Department.

2.5. Assumptions and Dependencies

We are assuming that our users and our developers have a basic understanding of the MGS. Additionally, we are assuming that our users and developers have the same definition of the MGS and share a standard grade book structure. Specifically, we will be implementing a gradebook that the way in which professors here at Davidson use the MGS. We will not allow the user to modify the general style of the gradebook in our application. (However, if a professor wanted to modify the style of the gradebook, they could do so by modifying the source code on GitHub.)

3. Detailed Requirements

3.1. External Interface Requirements

3.1.1. User Interfaces: We want to implement a chart for visualizing grade data (course progress, number of masters per topic, grades) for both professors and students modeled after the gradebook in the References section of this document. Additionally, our app will have a sidebar menu to navigate the page and toggle between courses as well as picker submission boxes used for data entry and storage.

3.1.2. Hardware Interfaces: For the scope of this class, the dashboard will be launched from a local machine and hosted on a Shiny.io server and will not interact with any specific hardware. The web app will be compatible with all standard web browsers.

3.1.3. Software Interfaces: The MGS is a stand alone application that does not interact with any other software applications. We are building a simple website using R, a widely used development language, so it will be compatible with other software if our system is expanded in the future. Our application will interact with a database hosted on our Shiny.io server over any kind of internet connection.

3.1.4. Communication Interfaces: Out MGS dashboard is not designed to be a communication tool. The professor can leave comments on individual assignments and we will include the professor’s email on the side, but no communication will take place within the site, though these features may be added in the future. The professors will also share the student’s grades through the database.

3.2. Functional Requirements

You will list each major item in your product and the requirements for each. For each actual requirement, use the word “shall” if the implementation must do it, “may” or “should” if it’s not mandatory, but just desirable. Most requirements should use “shall.” Remember, every requirement must be verifiable. Check that you have met the 6 characteristics in section 3.3.1 of good requirements – correct, complete, unambiguous, verifiable, consistent, ranked for importance and/or stability.

3.2.1. Professor Side

3.2.1.1. The professor shall be able to create a course, add at least 35 students to the course by Last Name, First Name, and create mastery concepts for the course through the application.

3.2.1.2 The professor shall be able to input grade data for each student, modify grade data for each student, and modify the mastery concepts through the application.

3.2.1.3 The professor shall be able to leave comments on individual assignments for each student, only visible by the individual student.

3.2.1.4 The professor shall be able to look up students individually by Last Name, First Name.

3.2.1.5 The professor may be able to download all of the student’s grades (or an individual student's grades) to a PDF or CSV file.

3.2.1.6 There shall be at least 1 professor with administrative control of each class (aka the ability to make modifications as described in 3.2.1.2).

3.2.1.7 The professor shall be able to toggle between all of their courses using the menu sidebar.

3.2.2. Student Side

3.2.2.1. Students shall be able to view the concepts mastered and the mastery level for these concepts (aka their grades in the course).

3.2.2.2 Students may be able to view and track their progress in the course through data visualization tools like graphs and charts.

3.2.2.3 Students shall be able to toggle between different courses they are currently enrolled in through a sidebar menu.

3.2.2.4 Students may be able to download their grade data for a specific course to a PDF or CSV file.

3.2.3. Database

3.2.1.1. The database shall be able to store user data for the student (Last Name, First Name, test grades, and additional field as requested) for both professors and students.
3.2.1.2. The database shall be able to store the grade data for each course (all concepts, level of mastery for each concept for each student, comments associated with each level of mastery and each student).
3.2.1.3. The database shall allow professors to create, modify, or delete any course data. (Professors will have full read and write access to their associated courses)
3.2.1.4 The database shall allow students to view their grades for each of their associated courses. (Students will have read only access their their own grade data)

3.3. Performance Requirements

3.3.1 We shall have a response time of less that 1 second for any user interactions with our database and the fastest response time that the free plan of the shiny apps server will provide.
3.3.2 Our throughput will also be as fast as the shiny apps server will provide

3.4. Design Constraints

Our dashboard is not dealing with large amounts of data, so we do not anticipate generating large files or encountering memory limitations. Overally, we have not identified any design constraints.

3.5. Attributes

We do not plan on developing a log-in system at this point, but this is a feature we hope to develop in the future. All other features are well described in the preceding sections.

3.6. Other Requirements

Because we are dealing with personal grade data for students, it is important for us to define security requirements for our application.

3.6.1 Security Requirements

3.6.1.1 Our application shall not allow students to view any grade data besides their own.

3.6.1.2 Our application will not allow students to write to the database and modify their grades.

3.6.1.3 Our application will not allow professors to view or modify any data from a course they are not registered with.

3.6.1.3 Our application will not allow anyone who is not a professor or a student to access the database.