Software Requirements Specification

For

JGradeBook

7/3/2017

[Version 1.0]

Prepared by:

Scott Blick

**Table of Contents**

[Revision History 2](#_Toc486947618)

[1 Introduction 4](#_Toc486947619)

[1.1 Overview 4](#_Toc486947620)

[1.2 Goals and Objectives 4](#_Toc486947621)

[1.3 Scope 4](#_Toc486947622)

[1.4 Definitions 4](#_Toc486947623)

[1.5 Document Conventions 5](#_Toc486947624)

[1.6 Assumptions 5](#_Toc486947625)

[2 General Design Constraints 5](#_Toc486947626)

[2.1 Product Environment 5](#_Toc486947627)

[2.2 User Characteristics 5](#_Toc486947628)

[2.3 Mandated Constraints 6](#_Toc486947629)

[2.4 Potential System Evolution 6](#_Toc486947630)

[3 Nonfunctional Requirements 6](#_Toc486947631)

[3.1 Usability Requirements 6](#_Toc486947632)

[3.2 Operational Requirements 6](#_Toc486947633)

[3.3 Performance Requirements 6](#_Toc486947634)

[3.4 Security Requirements 7](#_Toc486947635)

[3.5 Safety Requirements 7](#_Toc486947636)

[3.6 Legal Requirements 7](#_Toc486947637)

[3.7 Other Quality Attributes 7](#_Toc486947638)

[3.8 Documentation and Training 7](#_Toc486947639)

[3.9 External Interface 7](#_Toc486947640)

[3.9.1 User Interface 7](#_Toc486947641)

[4 Functional Requirements 8](#_Toc486947642)

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Name** | **Description** |
| 1.0 | 7/3/2017 | Scott Blick | First Iteration of the Software Requirements Specification Document |
|  |  |  |  |

# Introduction

## Overview

This document is used to provide specification of the requirements for the JGradeBook project. This document will help clearly define the project and the tasks that need to be accomplished.

JGradeBook is a subsystem of a LMS to keep track of student information and a record of their grades. Its intended use is for both students to view their own information, and administrations, that can view, add, and edit each students’ information. All this must be accomplished with a GUI and some form of database system.

## Goals and Objectives

There are two main goals for the JGradeBook project:

1. An easy to use program so students can view their grades.
2. Provide a basis for the administration to edit student as well as view each student’s information

## Scope

This program will store a student’s id, name, and the courses they are taking. Each course will have its own set of exam scores along with it. The program will be able to calculate the student’s GPA based on their equally weighted test scores. The student will be able to log in with their student id and password, but can only view their own information and cannot edit. The administration on the other hand, can log in using a admin id and password view all students’ information and edit them, as well as add and delete students.

## Definitions

**LMS** – Learning management system.

**GPA** – Grade point average.

**Administration** – refers to both the system admin and the school administration, since they will likely be one and the same in regards to using this software

**Java** – The programming language that will be used to develop this project.

**Use case** – describes a goal-oriented interaction between the system and an actor. A use case may define several variants called scenarios that result in different paths through the use case and usually different outcomes.

**Scenario** – one path through a use case

**Actor** – user or other software system that receives value from a use case.

**Role** – category of users that share similar characteristics.

**Product** – what is being described here; the software system specified in this document.

**Project** – activities that will lead to the production of the product described here. Project issues are described in a separate project plan.

**Shall** – adverb used to indicate importance; indicates the requirement is mandatory. “Must” and “will” are synonyms for “shall”.

**Should** – adverb used to indicate importance; indicates the requirement is desired but not mandatory.

**May** – adverb used to indicate an option. For example, “The system may be taken offline for up to one hour every evening for maintenance.” Not used to express a requirement, but rather to specifically allow an option.

**Controls** – the individual elements of a user interface such as buttons and check boxes.

## Document Conventions

This section describes presentation conventions use in the document.

***Example:***

Portions of this document that are incomplete will be marked with TBD. Each TBD item will have an owner and estimated date for resolving the issue.

## Assumptions

It is assumed that the target machine can run Java and have the programs necessary to view all documentation.

# General Design Constraints

## Product Environment

The product environment is simple, and only requiring that the target has java to run it and a mouse and keyboard to interact. Touch screens will not be tested for user input.

## User Characteristics

The first set of users are students, which can be considered part of the general public, since even though this is designed for higher education, it can be used in most school systems. No technical knowledge can be assumed from this set of users. The program must therefore be straightforward and easy to use.

The administration audience will either have a college or be in an IT position. This means we can assume that they will be able to use the more complicated editing part given proper instructions.

## Mandated Constraints

This software must have two distinct operating modes, one for normal user, or students, and one for the administration. The student cannot be allowed to make changes but the admin must be able make changes for every student.

## Potential System Evolution

There are several potential areas of expansion, such as attendance tracking. However, post-delivery maintenance is not the responsibility of this development team. We will try to make objects as cohesive and independent to help facilitate any future development.

# Nonfunctional Requirements

This software must have a GUI interface for both users.

This software must use some kind of external database. SQL databases or simple text files in folders count towards this.

## Usability Requirements

Test must readable without the user having to readjust the window sizes. Navigation must be done with clearly marked and visible buttons. The goal is to have the user put as little effort as possible to use this software. A user manual will accompany the program, and help dialog buttons may be added to the software directly.

## Operational Requirements

Any powered system loaded with Java and have a screen, mouse, and keyboard attached.

## Performance Requirements

As this is a small program, its performance should reflect that. The size of the program should almost be entirely dependent on the size of the database, and the time to perform should hardly be noticeable.

## Security Requirements

Security only applies to the software and not to the text files they access. A username and password is required for both students and administrators. A student user can only view, but not edit, their own information, and no one else’s.

## Safety Requirements

There are currently no safety concerns with this software.

## Legal Requirements

Student information should not be visible by other students. Only the administration and the one associated user can view a student’s information.

## Other Quality Attributes

Due to the fact that Java is very portable, this program will also be portable, as it only uses standard libraries.

## Documentation and Training

The following documents will be included alongside this program:

* User Manual
* Use-Case and UML Diagrams
* SPMP
* Source Code with documentation
* Test Cases

## External Interface

### User Interface

Login Interface: This will be used by both users. They will enter their user name and password in the program and click Login. The password text field should not show what characters are typed. If the login is incorrect, it should deny access and pop up window will display an error message.

Student Interface:

A simple interface that should display the name, id, courses, and grades for that student. There should also be a button to calculate GPA and display it in a pop-up window.

Admin Interface:

The admin interface should list all the students and provide and option for the admin to select one, or create a new. There should be a separate interface for creating a new student, allowing the admin to fill out all the fields. There will also be a specialized student view, allowing the admin to edit or delete student information.

# Functional Requirements

The following are a list of features that will be included in the product.

There needs to be two class objects. These will represent the data that will be stored in the text file databases

1. Courses

* Course name
* An array of exam scores

1. Student

* Name
* Student
* Array of courses with exam scores
* GPA calculation

There must also be various window that perform the required functions.

1. Login Window

* Allow user to enter a username and password
* Must be able to search for a user in the database
* Must be able to compare password with stored password in database
* Reject and give error message for incorrect login
* Accept correct login and display appropriate window for students and administrators

1. Student Window

* Display all student information from database, including name, student id, courses, and grades.
* Calculate and display the student’s GPA

1. Admin Window

* Display a list of all students
* Allow the admin to select a student to view their information in a new window
* Allow the admin to create a new student file in a new window

1. Admin Add Window

* Have fields for the admin to enter new data for a new student
* Create a new file with all the info saved

1. Admin Student Window

* Show a student’s information
* Give option to edit data with a new window
* Calculate and display GPA

1. Admin Edit Window

* Fetch student info and display it in an editable format
* Allow admin to make changes
* Replace student file with the new edited if saved
* Allow admin to delete student file