**Student Graduation Processing Back-end**

**for**

**Graduation Rule Assessment Data Systems (GRADS)**

**Version 1.0**

**Prepared by Blake Edwards, Benjamin Kronemeyer, and Colter Bodinot**

**CSCE 247, Software Engineering**

**02/04/19**

# Introduction

## Purpose

### Create a processing back-end for the graduation rule auditing project. This system will allow two types of users, students and graduate program coordinators to access student data and view the student’s degree progress. This system is to be put in place because the current paper-based degree program form is laborious manual task that can be expedited through this back-end.

## Scope

### This document will cover the overall description of the back-end system being built, requirements for user types, requirements for displayable information per user type, requirements for user actions per user type, and interfacing requirements.

## Definitions & Acronyms

### GRADS – Graduation Rule Assessment Data System

### GPC – Graduate Program Coordinator

### CS – Bachelor of Science in Computer Science

### CE – Bachelor of Science in Computer Engineering

### CIS – Bachelor of Science in Computer Information Systems

## References

### CSCE 247 – Project 1 – Draft Requirements

## Overview

### The document will begin with an overall description of the product. It will describe the type of product being built and for whom. It will also describe constraints placed upon the engineers building the system. Then the document will move into the Requirements section in which the required details of the product will be outlined. These requirements will be organized by user class. Finally, an appendix and index will follow for further referencing and searching.

# Overall Description

## Product Perspective

### This product will be a back-end system that can be accessed by various front-ends in a universal manner in order to view student progress within the Computer Science, Computer Engineering, and Computer Information Systems majors.

## Product Functions

### The GRADS system will allow a few user specific functions to be performed. For the student, it will allow them to view their degree progress. It will also allow them to select possible classes to see that classes impact to their progress, however they will not be able to save these schedules. The GCP will be able to view all students within the programs and view the same information the student can. However, the GCP will also be able to manually update grades, insert classes, and leave notes on the student profile.

## User Characteristics

### Students

#### Base user with no amendment rights

#### May only view their own student property data report including:

##### Name, Student ID, Degree

##### Degree progress

##### GPA

##### Advisor

##### Starting Semester

#### May enter prospective courses to view impact on degree.

### GPC

#### User with amendment rights

##### May update any student fields except Student ID, Network ID, First Name, and Last Name

##### May leave notes on student profiles visible to both the GPC and Student

#### Full access to all students and ID numbers

##### May view raw transcript data for any student

###### Includes all individual courses via course names, semester taken, and final grade

###### Cumulative GPA

## Constraints

### Do not “Gold Plate” the GRADS system. Stick to customer desires.

## Assumptions

### A front-end system is already in place and does not need to be created.

# Requirements

## External Interface Requirements

### User Interfaces

### Hardware Interfaces

### Software Interfaces

#### Various Front-end interfaces

##### Access point should be universal

### Communication Interfaces

#### Access points will transfer Java Objects using JSON formatting

## Functional Requirements

### Login

## Performance Requirements

## Design Constraints

## Other Requirements

### Test

# Appendices

## Any relevant information that can assist in understanding the requirements.

# Index

# Section 3.b., Functional Requirements can be organized in multiple ways, three recommendations are:

## 1: Organized by User Class

### ● User Class 1

### ○ Requirement 1.1

### ○ Requirement 1.2

### ○ …

### ○ Requirement 1.m

### ● User Class 2

### ● ….

### ● User Class N

## 2: Organized by Feature

### ● Feature 1

### ○ Requirement 1.1

### ○ Requirement 1.2

### ○ …

### ○ Requirement 1.m

### ● Feature 2

### ● ….

### ● Feature N

## 3: Organized by Mode

### ● Mode 1

### ○ Requirement 1.1

### ○ Requirement 1.2

### ○ …

### ○ Requirement 1.m

### ● Mode 2

### ● ….

### ● Mode N