Autograder

Group 3 - Python

Bryce Bjorkman, Shane Brandl, David Edwardson, Nolan Forehand, Eli Kolb, Zachary Moldenhauer, Tyler Reski, John Snyder

**Software Requirements Specification**

**Document**

**Version: Sprint 1** **Date: (09/18/2019)**

**Table of Contents**

**1. Introduction**

*1.1 Purpose*

*1.2 Scope*

**2. The Overall Description**

*2.1 Product Perspective*

2.1.1 System Interfaces

2.1.2 Interfaces

2.1.3 Hardware Interfaces

2.1.4 Communications Interfaces

2.1.5 Memory Constraints

2.1.6 Operations

2.1.7 Site Adaptation Requirements

*2.2 Product Functions*

*2.3 User Characteristics*

*2.5 Assumptions and Dependencies*

*2.6 Apportioning of Requirements*

**3. Specific Requirements**

*3.1 External interfaces*

*3.2 Functions*

*3.3 Performance Requirements*

*3.4 Logical Database Requirements*

*3.5 Design Constraints*

3.5.1 Standards Compliance

*3.6 Software System Attributes - Non-functional Requirements*

**4. Change Management Process**

**1. Introduction**

## **1.1 Purpose**

## The purpose of the Software Requirements Specification Document is to outline the functional and nonfunctional requirements of Group 3’s proposed academic programming assignment grading solutions software. The program will be able to grade Python language based assignments.

## **1.2 Scope**

## Within this document and upon release, Autograder will be the name of the proposed software product. Autograder should provide organized views and grading suggestions for sets of academic software assignments via a computer science professor’s personal computer. Autograder is intended to evaluate well-configured source files, and not to build project directories from flat source. Autograder should suggest grades to the user, but not finalize grades in any official academic system.

# **2. The Overall Description**

## **2.1 Product Perspective**

## Autograder should be a self-contained desktop application.

### **2.1.1 System Interfaces**

### The system should run on Mac, Windows, and Linux operating systems, using their respective application program interfaces to gain access to standard native computing and storage resources.

### 

### **2.1.2 Interfaces**

### Autograder should provide a graphical user interface to the user.

### **2.1.3 Hardware Interfaces**

### The system should not require hardware interfaces beyond those that are standard for a typical personal computer.

### **2.1.4 Communications Interfaces**

### The system should not require communications interfaces beyond access to user operating system application program interfaces for managing file systems and serving graphical content.

### **2.1.5 Memory Constraints**

### The system should run on as little as 4GB of RAM so as to function on a common laptop, and should use at most 8GB of RAM so as to limit secondary storage consumption of the user’s machine.

### **2.1.6 Operations**

### The user must unzip any compressed directories containing any files that need to be graded.

### **2.1.7 Site Adaptation Requirements**

### The system’s setup process should consist solely of copying the appropriate operating system’s executable of Autograder to the desired installation location.

## **2.2 Product Functions**

## 1). The major function of this software is to grade student submitted assignments. It will accomplish this in the following ways:

* Static code analysis
* Comparison of output values vs. expected output values
* Reporting of a final score with the suggested grade, and the areas where the student lost points (it's the professor’s decision after this point)

2). The professor will configure the system, or load a user created configuration saved. When the desired grading scheme is configured, static code analysis will be done, followed by a run of the program being graded. The results of the analysis and code run will be used to calculate a grade based on the professors grading configuration. A report will then be generated containing statistics on individual assignments, or the whole classes.

## **2.3 User Characteristics**

## Autograder users are expected to be computer science professors and/or teaching assistants. The users must have a basic knowledge of file management in order to import files to be graded.

## **2.5 Assumptions and Dependencies**

# **3. Specific Requirements**

## **3.1 External Interfaces**

## **3.2 Functions**

## The system shall:

* Allow versatility to evaluate folders OR singular files
* Deny grading of files other than text-based files (.py, .txt, etc.)
* Present its graphical user interface with appropriate buttons grayed out until the user has uploaded a file/folder
* Check the number of comment blocks vs methods in the document
* Check for the placement of docstrings at heads of documents and following definitions
* Check the existence of method names anywhere within the document
* Retrieve/regenerate code from .txt document and allow user to plug in different variables

## **3.3 Performance Requirements**

The system should support a single user evaluating as many as at least 1000 assignments in a single session, and grade each assignment in under 30 seconds plus the execution time of any user provided test cases. The system should accept file sizes up to 1GB.

## **3.4 Logical Database Requirements**

## This requirement is not applicable. There will be no database interaction. The application will run with access to provided files.

## **3.5 Design Constraints**

### **3.5.1 Standards Compliance**

### The system should log which assignments have been graded by the user, including the student associated with the assignment, when they were graded, the grade assigned, any test outputs, and any system errors encountered during evaluation.

## **3.6 Software System Attributes - Non-functional Requirements**

1. External Requirements
2. Efficiency Requirements
3. Dependability Requirements
4. Security Requirements
   1. These requirements are not applicable to the scope of this application. Any security issues are deferred to the local operating system.
5. Ethical Requirements
   1. Considering that any user will be a professor or teaching assistant working on assigning formal grades, the system should not export any identifying information regarding the students whose assignments are being evaluated.
6. Usability Requirements
7. Environmental Requirements
8. Operational Requirements
9. Development Requirements
   1. Autograder will be developed using the Python programming language.
   2. IDEs used include Pycharm and Visual Studio.

# **Change Management Process**

At the start of each sprint of development the team will evaluate the SRS and discuss any proposed changes with the client.