Writing
User Requirements
and
System Specification
Documents

Document Version: 0.9
Status: Under revision

Dept. of Computer Science & Engineering, York University

October 31, 2008

Revision History

Name	Date	Reason For Changes	Version
Jonathan S. Ostroff	Oct-31-2008	Initial Draft	0.9

Table of Contents

1. Introduction 1
1.1 Purpose 1
1.2 Intended Audience 1
1.3 Reading Suggestions 1
1.4 References 2
2. User Requirements Document 4
2.1 Introduction 4
2.1.1 Purpose 4
2.1.2 Intended Audience 4
2.2 Overview of the Problem Domain 4
2.3 Environment – description of ePost inputs 6
2.3.1 Distribution File 6
2.3.2 Grading Unit File 6
2.4 User Requirements 7
2.5 Example Scenarios 7
2.5.1 Distribution File: CSE3311E 7
2.5.2 Single Grading Unit File: test.ePost 8
2.5.3 Single Grading Unit Report: assignment.csv 9
2.6 Mathematical Description of Calculation of Averages 9
3. System Specification Document 12
3.1 Command Syntax 12
3.1.1 Creation of Single Grading Unit Reports 13
3.1.2 Error Handling 13
4. Traceability (Validation & Verification) 14
5. Glossary (Incomplete) 14
6. Issues List 14

1. Introduction

1.1 Purpose

This document provides a template for writing *User Requirements* and *System Specification* Documents. A small example is used to illustrate these two types of documents. *ePost* is a system used in the Department of Computer Science and Engineering. It allows Instructors to distribute grades electronically and securely to students. The sample requirements and specification documents describe a small extension to ePost. The extension provides instructors with the ability to view all the information associated with a single grading units (such as an assignment, test, exam) including statistical information such as the GPA average.

1.2 Intended Audience

This document is intended for students in the CSE4312 requirements engineering course.

1.3 Reading Suggestions

Document	**Domain**	**View**	**Role**
User Requirements Document	Problem domain	Stakeholders View	State *what* the stakeholders want to achieve. Avoid reference to particular solutions
System Specification Document	Solution Domain	Requirements Engineering Team & Software Analysts	State abstractly how a proposed *System* will meet stakeholder requirements. Avoid reference to detailed design.
Architectural Design Document	Solution Domain	Software Analysts & Developers	State how a proposed design will meet system specifications.
…			
Code	The final product		

The *User Requirements Document* describes the user's goals and needs in the given *problem domain*. The main task of user requirements is to help the stakeholders express their needs in writing.

The *System Specification Document* describes a *solution* that will satisfy the needs of the users. The solution in this case is a future application that is under construction (*dPost*). The User Requirements Document describes results (external properties) desired by the user. In contrast, the System Specification Document describes, abstractly, an application (a Machine *M*) that will achieve those results. It is abstract because it does not describe all the details of the application (e.g. it does not describe internal implementation details). Rather, it describes only the external input-output behavior of the application. Specifications must solve every part of the problem described in the Requirements Document and must, thus, trace back to the user requirements.

Requirements and Specifications must be *testable*, i.e. they must define the desirable results, properties or behaviors with enough precision and detail so that tests can be written to check that they are satisfied.

Traceability tables and *example scenarios* (see Section 2.5) help to validate the specification as described by Software Engineering theory (described at the beginning of the course), i.e.

> **Validation of the Specification**: ENV **and** SPEC(*M*) ➔ REQ
> **Verification of the Implementation**: *IMP*(*M*) ➔ spec(M)

Together, Validation & Verification entail the required behavior: ENV **and** *IMP*(*M*) ➔ REQ. This means that the behaviors of the implemented software application in the given problem domain (ENV) satisfies the requirements. The current document is thus part of SPEC(*dPost*).

Note the following while reading the document.

- This document is an example of how to write design specifications (and, for that matter, user requirements).
    - A hierarchical organization helps the reader to understand the overall structure.
    - Specifications are broken down into *atomic* statements that state the precise result to be achieved in enough detail to be *testable*.
    - Each atomic statement is *unique*, i.e. is a single, traceable, element with a unique identifier.
    - Each atomic statement is *clear, precise, consistent,* and *complete*.
    - A statement of specification (or requirement) is *abstract*, i.e. does not impose implementation or design detail relevant to a lower level.
    - A statement must be *feasible* (e.g., "the system must be 100% reliable" is not practical).
    - It is often useful to illustrate a specification statement with an example, where suitable. Examples and other details are cross-referenced in the Comments Column.
    - Example Scenarios are the basis for testing the requirements.
    - Consistent terminology and examples are used. A glossary helps.
    - The document avoids ambiguity and let-out clauses that weaken the requirements. Repeated use of conjunctions in an atomic statement, may make the statement assert too many requirements at the same time.
    - Avoid rambling; conciseness is a virtue; we are not writing a novel;
    - Avoid speculations and wishful thinking and infeasible requirements
- This document is a design specification for the part of dPost you must implement in Phase3 of the Project.
- The Traceability Table (section 4) and example scenarios (section 2.5) will help you to see how to validate specifications and verify that the final implementation satisfies the requirements (as required by SE-theory). You will need to provide many more Example Scenarios of your own to test all the different use cases.

1.4 References

See Section 4, *Writing Better Requirements*, Telelogic, MN-DR-ED-GTR80-05-01. See also E. Hull et. al. *Requirements Engineering*, Springer 2004.

User Requirements Document

for

ePost Grading Unit Reports

Document Version: 1
Status: Approved

Dept. of Computer Science & Engineering, York University
October 31, 2008

Revision History

Name	Date	Reason For Changes	Version
Jonathan S. Ostroff	Oct-31-2008	Initial Draft	1.0

2. User Requirements Document

2.1 Introduction

2.1.1 Purpose

ePost is a system used in the Department of Computer Science and Engineering. It allows Instructors to distribute grades electronically and securely to students. This document is a partial description of some of the User Requirements for a small extension to ePost. The extension provides instructors with the ability to view all the information associated with a single grading units (such as an assignment, test or exam) including statistical information such as the GPA average. This document does not describe the wider needs of Instructors but limits itself to the need to view all the data associated with a single Grading Unit

2.1.2 Intended Audience

This document is intended for students in the CSE4312 requirements engineering course. It provides them with an example of a *User Requirements Document*.

***Note**: This document is by no means, complete. See the References in Section 1.4 for more detail.*

2.2 Overview of the Problem Domain

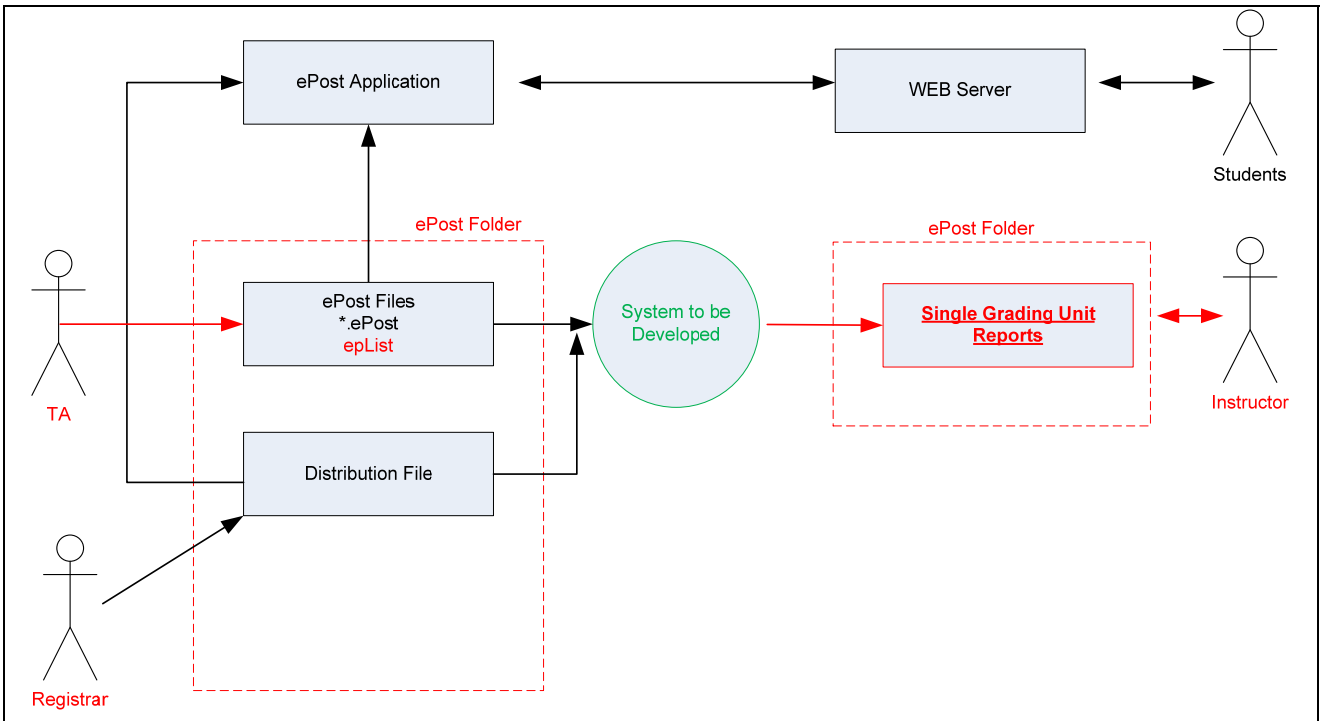


Figure 1 Context Diagram

The current ePost system allows entry of grades as *.ePost files (e.g. *exam.ePost*) in the following format[1]:

```
+ Grades for the Exam for CSE3311E
cs654321        86
cse12345        76
+ CPS weight=30%: max=100: due=Oct. 17, 07: available=Nov. 9, 07
```

ePost allows the instructor to distribute marks securely to students, and to generate consolidated files with all units in Excel CSV format for final grade calculations, although it does not do the actual calculation of grades. To do this, Instructors place the ePost files in an ePost folder in a special course directory allocated by the Department.

Instructors may add arbitrary meta-data by prefixing a line with a "+". Students see the meta-data lines as well as the specific entries relevant to them (as determined by the Prism login).

The Prism login (e.g. cse12345) is checked against a distribution file to see if the student is registered in the course. Grades are only reported for those students who are registered as reported in a special Distribution File (obtained from the Registrar) for that course.

[1] http://www.cse.yorku.ca/~roumani/ePost/

2.3 Environment – description of ePost inputs

2.3.1 Distribution File

ID	Description	Actor	Comments
E1	**Distribution File**. The Distribution File, obtained from the Registrar, lists all students officially registered in the course.	Registrar	See Section 2.5.1 for an example of a Distribution File.
E1.1	The precise syntax of a Distribution File is described in Section 2.5.1. For each student, information such as the *Student Nam*e, *Student Number*, and *Prism Login* is provided.		
E1.2	Instructors place the latest version of the Distibution File in the ePost Directory	Instructor	The ePost Directory is a directory on the Departmental servers where grading information is recorded.

2.3.2 Grading Unit File

E2	**ePost files.** An ePost file (prepared by Instructors and Teaching Assistants) records the grades obtained by students in a given Grading Unit. The file maps each student (as indentified by their Prism login) to a Raw Mark score.	Instructor, TA	For an example, see Section 2.5.2
E2.1	Instructors may record additional information for students using the meta-data prefix "+". The ePost system supports a standard CPS meta-data entry using the key words *weight* of the Grading Unit and the *max* (i.e. maximum) Raw Mark Score that can be obtained by students in this unit.		For example, `+ CPS weight=20%: max=100: due=Sep. 25, 07: available=Oct. 11, 07`
E2.2	In order to support new reporting functionality (see sequel), Instructors have agreed to enter a new meta-data Cutoffs line in the ePost file using the keyword *cutoff* (see comment for an example). The Cutoffs line specifies a map from Raw Marks obtained by students in the Grading Unit to a Letter Grade. The maps (e.g. A+=88) are assigned by the Instructor (and need not		For example, `+ cutoff:A+=88,A=77,B+=74,B=70,C+=65,C=60,D+=55,D=52,E=40,F=0`

	be standard (such as A+=90).		
E2.3	An ePost Grading Unit file must have the ePost extension in its file name		Example ePost file names: `assignment.ePost`, `exam.ePost`

2.4 User Requirements

ID	Description	Status	Comments
	Single Grading Unit Report		
R1	Given an ePost file for a Grading Unit, an Instructor shall be able to generate (from the ePost file) a Single Grading Unit Report.	Approved	For an example of the Single Grading Unit Report, see Section 2.5.3
R1.1	The Single Grading Unit Report shall report results for all students registered in the course as determined from the course Distribution File. For each student, the Report will contain the student number, Prism login, raw mark score, and equivalent letter grade.	Approved	
R1.2	The Single Grading Unit Report shall contain a calculation of the average for this Unit. Section 2.6 explains how the average is calculated.	Approved	
R2	**Error handling**: *To be done*.		Missing Files, Missing meta-data etc.

2.5 Example Scenarios

2.5.1 Distribution File: CSE3311E

The syntax is

```
<PrismLogin> <YUlogin> <StudentNumber> <LastName>,<FirstName>
```

There is whitespace between entries and line termination may be either Windows or linux format. For example,

```
<NO-CS-ACCT>  <NO-YU-ACCT>    206394662   Ahmad, Alex
cs233222      <NO-YU-ACCT>    204084968   Badwal, Hasitha Pathirage Dhananjaya
cs233922      bderek          206518567   Benda, Bunarith K
cse73842      cailing         206029458   Sotoudeh-Hosseini, Xiaoling
cs233012      fordcraw        206677890   Crawford Sriling, Jack
cs234444      mashtin09       206012345   Crotes, Butchin Aubrey
cs212345      bits0$kid       207474406   Vishanth, Alexander
cs987657      rosejack        207555555   Gao, Ivan H
cs222445      agloryday       123456789   Stonehead, Trudel
cs233018      speers08        206860050   Speers, Andrew
cs233567      tallingron      776654356   Tal, Bunarith
cs123457      junket          888999666   Tan, Ron
```

2.5.2 Single Grading Unit File: test.ePost

```
+ Assignment Raw mark Scores
+ See Course website for grade details
cs234444  135
cs212345  120
cs987657  112.5
cs233222  105
cs233922  97.5
cse73842  90
cs123456  149
cs233012  82.5
cs222445  75
cs233018  60
cs233567  30
+ cutoff:A+=90,A=80,B+=75,B=70,C+=65,C=60,D+=55,D=50,E=40,F=0
+ CPS weight=20%: max=150: due=Sep. 25, 07: available=Oct. 11, 07
```

2.5.3 Single Grading Unit Report: assignment.csv

```
CSE3311Z Single Grading Unit Report for: test.ePost
Login,User ID,ID,Full Name,assignment
cs233222,<NO-YU-ACCT>,204084968,"Badwal, Hasitha Pathirage Dhananjaya",105,A
cs233922,bderek,206518567,"Benda, Bunarith K",97.5,A+
cse73842,cailing,206029458,"Sotoudeh-Hosseini, Xiaoling",90,A+
cs233012,fordcraw,206677890,"Crawford Sriling, Jack",82.5,A
cs234444,mashtin09,206012345,"Crotes, Butchin Aubrey",135,A+
cs212345,bits0$kid,207474406,"Vishanth, Alexander",120,A+
cs987657,rosejack,207555555,"Gao, Ivan H",112.5,A+
cs222445,agloryday,123456789,"Stonehead, Trudel",75,B+
cs233018,speers08,206860050,"Speers, Andrew",60,C
cs233567,tallingron,776654356,"Tal, Bunarith",30,F
,
GPA Average,,7.3,A
Raw Mark Average,,90.75,A+
,
Maximum,150
Weight,20
test.ePost,CutOff,"A+=90,A=80,B+=75,B=70,C+=65,C=60,D+=55,D=50,E=40,F=0"
```

Instructors may view the above CSV report using a spreadsheet application.

Login	User ID	ID	Full Name	test	
cs233222	<NO-YU-ACCT>	204084968	Badwal, Hasitha Pathirage Dhananjaya	105	A
cs233922	bderek	206518567	Benda, Bunarith K	97.5	A+
cse73842	cailing	206029458	Sotoudeh-Hosseini, Xiaoling	90	A+
cs233012	fordcraw	206677890	Crawford Sriling, Jack	82.5	A
cs234444	mashtin09	206012345	Crotes, Butchin Aubrey	135	A+
cs212345	bits0$kid	207474406	Vishanth, Alexander	120	A+
cs987657	rosejack	207555555	Gao, Ivan H	112.5	A+
cs222445	agloryday	123456789	Stonehead, Trudel	75	B+
cs233018	speers08	206860050	Speers, Andrew	60	C
cs233567	tallingron	776654356	Tal, Bunarith	30	F
GPA Average		7.3	A		
Raw mark average		90.75	A+		
Maximum	150				
Weight	20				
test.ePost	CutOff	A+=90,A=80,B+=75,B=70,C+=65,C=60,D+=55,D=50,E=40,F=0			

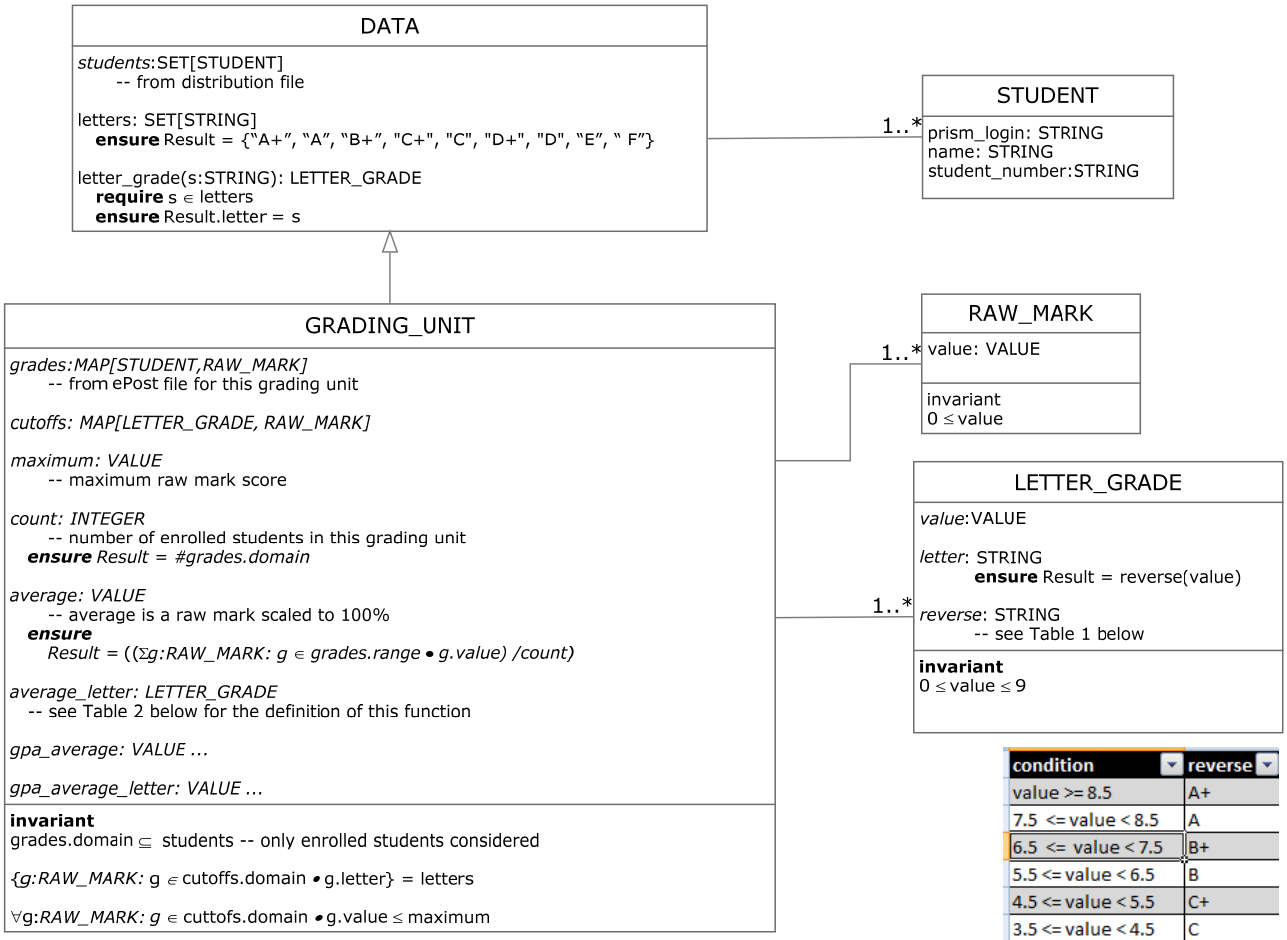
Figure 2 Viewing a CSV file in a spreadsheet application

2.6 Mathematical Description of Calculation of Averages

The UML class diagram (with contracts) in Figure 3 provides a mathematical description of how to calculate the GPA and Raw Mark averages for the Single Grading Unit Report.

For example, in the Single Grading Unit report of Section 2.5.3, the Raw mark Average is 90.75 and the associated Letter Grade is A+. The GPA average is 7.3 and associated Letter Grade is an A.

See the definition of the operations *average*, *average_letter*, *gpa_average* and *gpa_average_letter* in class GRADING_UNIT.



condition	reverse
value >= 8.5	A+
7.5 <= value < 8.5	A
6.5 <= value < 7.5	B+
5.5 <= value < 6.5	B
4.5 <= value < 5.5	C+
3.5 <= value < 4.5	C
2.5 <= value < 3.5	D+
1.5 <= value < 2.5	D
0.5 <= value < 1.5	E
value < 0.5	F

Table 1

condition	average_letter
cutoffs(letter_grade("A+")).value <= average	A+
cutoffs(letter_grade("A")).value <= average < cutoffs(letter_grade("A+")).value	A
cutoffs(letter_grade("B+")).value <= average < cutoffs(letter_grade("A")).value	B+
cutoffs(letter_grade("B")).value <= average < cutoffs(letter_grade("B+")).value	B
cutoffs(letter_grade("C+")).value <= average < cutoffs(letter_grade("B")).value	C+
cutoffs(letter_grade("C")).value <= average < cutoffs(letter_grade("C+")).value	C
cutoffs(letter_grade("D+")).value <= average < cutoffs(letter_grade("C")).value	D+
cutoffs(letter_grade("D")).value <= average < cutoffs(letter_grade("D+")).value	D
cutoffs(letter_grade("E")).value <= average < cutoffs(letter_grade("D")).value	E
0 <= average < cutoffs(letter_grade("E")).value	F

Table 2: function average_letter

Figure 3 Calculation of Statistical Averages for a Single Grading Unit

System Specification Document

for

ePost Grading Unit Reports

Document Version: 0.9
Status: Not yet approved

Dept. of Computer Science & Engineering, York University
October 31, 2008

Revision History

Name	Date	Reason For Changes	Version
Jonathan S. Ostroff	Oct-31-2008	0.9	0.9

3. System Specification Document

This document describes the input/output behaviour of a Linux command line application, *dPost*, that satisfies the User Requirements outlined in Section 2.

3.1 Command Syntax

ID	Description	Status	Comments
S1	The syntax of the dPost command shall be: *dpost <DistributionFile> –e <GradingUnitFile> [-sa]*	Approved	For example, *dpost CSE3311E –e test.ePost*
S1.1	Runtime behavior of dPost shall be independent of switch order.	Approved	The following commands are the same: *dpost CSE3311E –e test.ePost* *dpost –e test.ePost CSE3311E*
S1.3	dPost shall support the handling of input files whether these files were created on Linux or Windows.	Approved	Line termination characters (e.g. using Carriage Return CR and Line Feed LF) are handled differently by Linux and Windows. See http://usertools.plus.net/tutorials/id/22

3.1.1 Creation of Single Grading Unit Reports

ID	Description	V	Status	Comments
S2	The dPost command with the e-switch (see S1) generates a Single Grading Unit Report (in CSV format) as described in the User Requirements Document.			For an example, see Section 2.5.3
S2.1	The Single Grading Unit Report shall be stored in a file <*GradeUnitFile*>.*cvs*, located in the ePost folder.	1	Approved	For example, the command *dpost CSE3311E –e test.ePost* generates: *test.csv*
S2.2	The Single Unit Report shall be sorted by the raw mark column from highest raw mark to lowest raw mark score.	1	Approved	This is the default sorting order.
S5	dPost shall sort Single Unit Reports by family name (i.e. last name and then first name) when dPost executes with the *sa* switch: *dpost <DistributionFile> –e <GradingUnitFile> -sa*	1	Approved	Example: For example, *dpost CSE3311E –e test.ePost -sa*

3.1.2 Error Handling

ID	Description	V	Status	Comments
S3	dPost shall print “Error invalid input parameters” to STDERR and print help to STDOUT when dPost executes with wrong parameters	1	Approved	Example: *dpost –b*
S3.1	dPost shall be print “dPost cannot find the file specified” in STDERR when dPost executes with a non existent Data Distribution or Grading Unit Files	1	Approved	Example: *dpost somenonexistingfile*
S3.2	For all other errors enumerated in the Requirements Document, dPost shall output the error to STDERR. Errors shall also be printed to a file *log.txt* in the ePost directory.	1	Approved	

4. Traceability (Validation & Verification)

In this section, we illustrate an (incomplete) Traceability Table to validate that the system specification satisfies the user requirements.

Traceability Table

Requirements	Specification	Comment	Test Status
R1			
R1.1	S2 (together with E1 and E2)	Test with Example Scenario in Section 2.5.3	Test not yet run
R1.2			
R2			

Ultimately, we should be able to show how each user requirement is satisfied by a corresponding system specification. When the final system is implemented, we may use the Example Scenarios to generate acceptance tests for each user requirement.

5. Glossary (Incomplete)

Cutoff: A map from raw mark scores to letter grades.

ePost Directory: is a directory on the Departmental servers where grading information is stored for distribution to students via the Web.

Dpost, dPost: "dPost" in this document refers to the implanted software application. "dpost" is used in the command line to invoke the dPost application.

Grading Unit: The components used to evaluate the final grade of a student such as assignments, tests and exams.

STDOUT: Standard Output

STDERR: Error Output

6. Issues List

There is only one scenario example. Many more scenario examples should be added in order to provide Acceptance Tests for all use cases. Glossary is incomplete. Also, add Priority/Version/Risk attributes for Triage. Description of Grade calculations is incomplete
Error handling requirements is incomplete.