Software Requirements Specification

for

KSU CSWS

**Version 1.0 approved**

**Prepared by Yeojin Kim, Shiva Voruganti, Nidhi Shah, Junan Zhao, Jean-Jacques Muteteke**

**Senior Project**

**9/21/2016**

**Table of Contents**

**Table of Contents**

**Revision History**

**1. Introduction**

1.1 Purpose

1.2 Document Conventions

1.3 Intended Audience and Reading Suggestions

1.4 Product Scope

1.5 References

**2. Overall Description**

2.1 Product Perspective

2.2 Product Functions

2.3 User Classes and Characteristics

2.4 Operating Environment

2.5 Design and Implementation Constraints

2.6 User Documentation

2.7 Assumptions and Dependencies

**3. External Interface Requirements**

3.1 User Interfaces

3.2 Hardware Interfaces

3.3 Software Interfaces

3.4 Communications Interfaces

**4. System Features**

4.1 System Feature 1

4.2 System Feature 2 (and so on)

**5. Other Nonfunctional Requirements**

5.1 Performance Requirements

5.2 Safety Requirements

5.3 Security Requirements

5.4 Software Quality Attributes

5.5 Business Rules

**6. Other Requirements**

**Appendix A: Glossary**

**Appendix B: Analysis Models**

**Appendix C: To Be Determined List**

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason For Changes** | **Version** |
|  |  |  |  |
|  |  |  |  |

# Introduction

## Purpose

*<Identify the product whose software requirements are specified in this document, including the revision or release number. Describe the scope of the product that is covered by this SRS, particularly if this SRS describes only part of the system or a single subsystem.>*

Our objective is to make the processes of course substitution a lot better in terms of performance, time and flexibility between advisors and department coordinators. Our objective would be use digital and web technology to create and implement a better system where timeframe to process substitution forms would be much lower. As if now, the process takes at least 3 weeks to update in student's transcript profile. We want to create a system where students can log in and electronically fill out the substitution form and advisors and departments chairs can't take necessary actions such as approve, disapprove, comments and request other information.

## Document Conventions

*<Describe any standards or typographical conventions that were followed when writing this SRS, such as fonts or highlighting that have special significance. For example, state whether priorities for higher-level requirements are assumed to be inherited by detailed requirements, or whether every requirement statement is to have its own priority.>*

## Intended Audience and Reading Suggestions

*<Describe the different types of reader that the document is intended for, such as developers, project managers, marketing staff, users, testers, and documentation writers. Describe what the rest of this SRS contains and how it is organized. Suggest a sequence for reading the document, beginning with the overview sections and proceeding through the sections that are most pertinent to each reader type.>*

*This document is intended for all the members of CSWS team to use for references. This document is also intended for all the stakeholders such as advisors, Deans and department chairs who will be users our product. It is also intended for our professor Dr.Wagner.*

## Product Scope

*<Provide a short description of the software being specified and its purpose, including relevant benefits, objectives, and goals. Relate the software to corporate goals or business strategies. If a separate vision and scope document is available, refer to it rather than duplicating its contents here.>*

*Our product scope can be deduced from section 1.1.*

## References

*<List any other documents or Web addresses to which this SRS refers. These may include user interface style guides, contracts, standards, system requirements specifications, use case documents, or a vision and scope document. Provide enough information so that the reader could access a copy of each reference, including title, author, version number, date, and source or location.>*

# Overall Description

## Product Perspective

*<Describe the context and origin of the product being specified in this SRS. For example, state whether this product is a follow-on member of a product family, a replacement for certain existing systems, or a new, self-contained product. If the SRS defines a component of a larger system, relate the requirements of the larger system to the functionality of this software and identify interfaces between the two. A simple diagram that shows the major components of the overall system, subsystem interconnections, and external interfaces can be helpful.>*

## Product Functions

*<Summarize the major functions the product must perform or must let the user perform. Details will be provided in Section 3, so only a high level summary (such as a bullet list) is needed here. Organize the functions to make them understandable to any reader of the SRS. A picture of the major groups of related requirements and how they relate, such as a top level data flow diagram or object class diagram, is often effective.>*

## User Classes and Characteristics

*<Identify the various user classes that you anticipate will use this product. User classes may be differentiated based on frequency of use, subset of product functions used, technical expertise, security or privilege levels, educational level, or experience. Describe the pertinent characteristics of each user class. Certain requirements may pertain only to certain user classes. Distinguish the most important user classes for this product from those who are less important to satisfy.>*

## Operating Environment

*<Describe the environment in which the software will operate, including the hardware platform, operating system and versions, and any other software components or applications with which it must peacefully coexist.>*

## Design and Implementation Constraints

*<Describe any items or issues that will limit the options available to the developers. These might include: corporate or regulatory policies; hardware limitations (timing requirements, memory requirements); interfaces to other applications; specific technologies, tools, and databases to be used; parallel operations; language requirements; communications protocols; security considerations; design conventions or programming standards (for example, if the customer’s organization will be responsible for maintaining the delivered software).>*

## User Documentation

*<List the user documentation components (such as user manuals, on-line help, and tutorials) that will be delivered along with the software. Identify any known user documentation delivery formats or standards.>*

## Assumptions and Dependencies

*<List any assumed factors (as opposed to known facts) that could affect the requirements stated in the SRS. These could include third-party or commercial components that you plan to use, issues around the development or operating environment, or constraints. The project could be affected if these assumptions are incorrect, are not shared, or change. Also identify any dependencies the project has on external factors, such as software components that you intend to reuse from another project, unless they are already documented elsewhere (for example, in the vision and scope document or the project plan).>*

# External Interface Requirements

## User Interfaces

*<Describe the logical characteristics of each interface between the software product and the users. This may include sample screen images, any GUI standards or product family style guides that are to be followed, screen layout constraints, standard buttons and functions (e.g., help) that will appear on every screen, keyboard shortcuts, error message display standards, and so on. Define the software components for which a user interface is needed. Details of the user interface design should be documented in a separate user interface specification.>*

## Hardware Interfaces

*<Describe the logical and physical characteristics of each interface between the software product and the hardware components of the system. This may include the supported device types, the nature of the data and control interactions between the software and the hardware, and communication protocols to be used.>*

## Software Interfaces

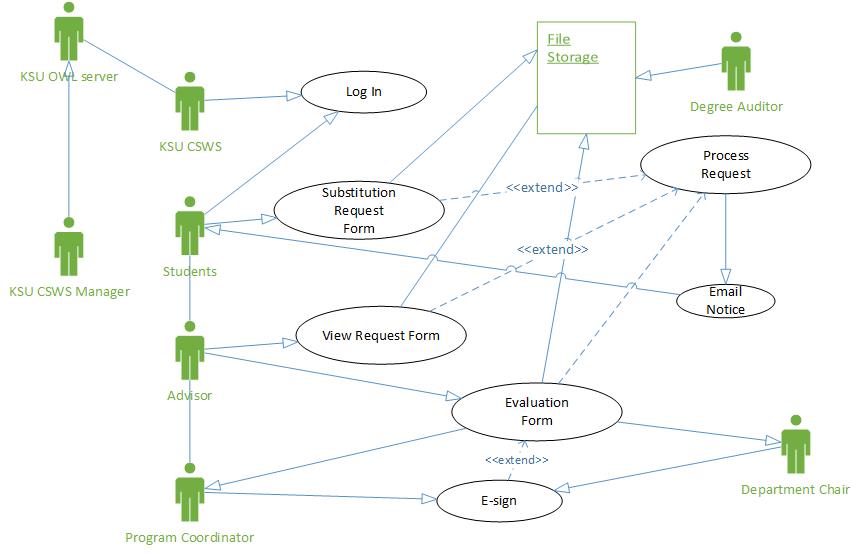
*<Describe the connections between this product and other specific software components (name and version), including databases, operating systems, tools, libraries, and integrated commercial components. Identify the data items or messages coming into the system and going out and describe the purpose of each. Describe the services needed and the nature of communications. Refer to documents that describe detailed application programming interface protocols. Identify data that will be shared across software components. If the data sharing mechanism must be implemented in a specific way (for example, use of a global data area in a multitasking operating system), specify this as an implementation constraint.>*

## Communications Interfaces

*<Describe the requirements associated with any communications functions required by this product, including e-mail, web browser, network server communications protocols, electronic forms, and so on. Define any pertinent message formatting. Identify any communication standards that will be used, such as FTP or HTTP. Specify any communication security or encryption issues, data transfer rates, and synchronization mechanisms.>*

# System Features (Use Cases) - YEOJIN KIM

## UML Diagram



## Figure #. UML Diagram

Figure #. The UML diagram is used to display actors and their interactions with the system. In this UML diagram we have a list of actors (KSU OWL server, KSU CSWS Manager, KSU CSWS, Student, Advisor, Program Coordinator, and Department Chair), and note that actors such as KSU OWL server and Degree Auditor are existing systems. There are a number of use cases in our system, and they serve as functions of our system. Use cases can be independent functions, they can “uses” another use case to complete their task, or they can “extends” another use case, meaning they are a subtask of that function. The file storage is a saved document of the course substitution forms from student and faculty.

## 

## Student Login

**ID and name**: UC-1 Student Login

**author:**

**data created:**

**actors:**

**summary:**

**trigger:**

**preconditions:**

**postconditions:**

**flow of events:**

**priority:**

**assumptions:**

*As a student i should be able to login to this product the same way i log in to any kennesaw application. (high priority)*

*When the student clicks the login button. if the password is correct the student will see the main page.*

*if the student puts an invalid username and/or password then the student will see an error notification.*

When a student presses the login button, if username and password are correct they should be directed to the home screen

When a student presses the login button, if username and/or password are incorrect they should see an error notification

## Faculty Login

**ID and name**: UC-2 Faculty Login

**author:** Yeojin Kim

**data created:** 9/29/2016

**actors:** KSU CS Adviser, Program Coordinator, Department Chair

**summary:** This is for KSU CS faculties who would like to login so that they can see all the course substitution requests and approve or deny the course substitution requests.

**trigger:**User presses login button with user ID and password.

**preconditions:**User must be KSU Computer Science faculty (adviser, program coordinator, and department chair). Faculties must enter correct user ID and password. Account validation must happen.

**postconditions:**The System should log the faculty in

**flow of events:**

* **normal flows:**

1. User presses “Login with Faculty” button
2. Faculty enters user ID and password
3. Account validation occurs
4. User is successfully logged in and can now see all the course substitution requests from students.

* **alternative flows:**

1. User presses “forgot password”
2. User enters user ID
3. Password is emailed

* **exceptional flows:**

1. User presses “Login with Faculty” button
2. Faculty enters incorrect/invalid user ID/password
3. Account validation occurs
4. User ID or password is incorrect and user is not logged in

**priority:** 9/10

**assumptions:** User is KSU CS Faculty

## Course Substitution Request (Student)

**ID and name**: UC-3 Course Substitution Request

**author:** Yeojin Kim

**data created:** 9/29/2016

**actors:** Students

**summary:** Users can see the “New Request” button. After users click the button, they can see a web page with various text inputs. They should fill in all the required information such as KSU student ID, E-mail, and Course name... etc. After they finish filling out the form, they click “Submit” button. Now the course substitution request form will be stored in the file storage.

**trigger:** Users press the “New Request” button. After they’ve done, press the “Submit” button.

**preconditions:** Users logged in as a student account, and they will see the categories: my requests, new requests, help, and logout.

**postconditions:** After submit the new request, users will receive the email that notifies them “You have submitted the course substitution request form successfully!”

**flow of events:**

* **normal flows:**

1. “New Request” page is opened
2. Students type in all the required informations to create a course substitution evaluation form.
3. Students should sign by E-signature feature (UC-7 E-Sign).
4. The Course Substitution Request Form will be generated when students press the “Submit” button.
5. The Course Substitution Request Form will be stored in the section of “Active” list on the File Storage feature (UC-6 File Storage).

* **alternative flows:**

2a. Students press “clear all” button to re-type in the new form.

4a. Students press “Cancel” button.

4b. “Are you sure you want to cancel the request form?” message is popped up.

4c. Student can chooses Yes or No

* **exceptional flows:**

4a. Error message popes up when students leave blank on the one of the

required

**priority:** 10/10

**assumptions:** User knows about their course requirements and course curriculum, and they have provided supplemental information about the completed course.

When the user finishes filling out the form, he clicks the submit button. If there are no errors he gets a confirmation that his form has been submitted.

if there is an error he/she will be notified.

## Course Substitution Evaluation (Faculty)

**ID and name**: UC-4 Course Substitution Evaluation

**author:**

**data created:**

**actors:**

**summary:**

**trigger:**

**preconditions:**

**postconditions:**

**flow of events:**

**priority:**

**assumptions:**

Identical to feature 4.4 except this can be done by the advisor, dean or department chair

## View Course Substitution Request and Records (Faculty only)

**ID and name**: UC-5 View CSWS History

**author:**

**data created:**

**actors:**

**summary:**

**trigger:**

**preconditions:**

**postconditions:**

**flow of events:**

**priority:**

**assumptions:**

## File Storage

**ID and name**: UC-6 File Storage

**author:**

**data created:**

**actors:**

**summary:**

**trigger:**

**preconditions:**

**postconditions:**

**flow of events:**

**priority:**

**assumptions:**

* + 1. Description and Priority

Every sub. form allows students or any of the faculty members to upload necessary files that may be needed for the course substitution form to be accepted. Such as transcripts, syllabus, etc..

* + 1. Stimulus/Response

when the student or faculty member is in the process of creating a sub. form. they can click on the upload extra files button.

the button will open a file selection window and the student or faculty member will choose the necessary file(s).

if the total file size is greater then some to be determined number, the user will receive an notification that the file is to big.

if the total file size is less than or equal to some to be determined number, the file(s) will be uploaded/attached to the application.

* + 1. Functional requirement

[R#] when the user presses the upload extra files button they should see the file selection window

[R#] when the user chooses the files and clicks ok to select those files, if the total file size is greater than maximum size, show warning else submit those files.

## E-Sign

**ID and name**: UC-7 E-Sign

**author: Yeojin Kim**

**data created:**

**actors:**

**summary:**

**trigger:**

**preconditions:**

**postconditions:**

**flow of events:**

**priority:**

**assumptions:**

## Processing Course Substitution Request

**ID and name**: UC-8 Processing Course Substitution Request

**author:**

**data created:**

**actors:**

**summary:**

**trigger:**

**preconditions:**

**postconditions:**

**flow of events:**

**priority:**

**assumptions:**

## Email Notification

**ID and name**: UC-9 Email Notification

**author:**

**data created:**

**actors:**

**summary:**

**trigger:**

**preconditions:**

**postconditions:**

**flow of events:**

**priority:**

**assumptions:**

## 

# Other Nonfunctional Requirements----Andy

## Performance Requirements

*<If there are performance requirements for the product under various circumstances, state them here and explain their rationale, to help the developers understand the intent and make suitable design choices. Specify the timing relationships for real time systems. Make such requirements as specific as possible. You may need to state performance requirements for individual functional requirements or features.>*

*The system GUI could change its theme color according to user’s preference.*

*The system should process the user’s login information in no more than one second.*

*The system should take no more than one second to logout user from system.*

*The system should take no more than five seconds to generate a PDF version of the substitution application after student’s submission.*

*The system should take no more than ten seconds to send an application notification email to advisor’s email account.*

*The system should take no more than one second to pop up the PDF application after advisor clicks on any one application item in the application table.*

## Safety Requirements

*<Specify those requirements that are concerned with possible loss, damage, or harm that could result from the use of the product. Define any safeguards or actions that must be taken, as well as actions that must be prevented. Refer to any external policies or regulations that state safety issues that affect the product’s design or use. Define any safety certifications that must be satisfied.>*

## Security Requirements

*<Specify any requirements regarding security or privacy issues surrounding use of the product or protection of the data used or created by the product. Define any user identity authentication requirements. Refer to any external policies or regulations containing security issues that affect the product. Define any security or privacy certifications that must be satisfied.>*

Because the system’s purpose is serving KSU students, so people who are not KSU students

shouldn’t have access to the system. If is possible, we plan to put the system into KSU Owl Express so that other people cannot log in.

After student submit his/her substitution application, the product should guarantee that no one could make modification on the original application form, even advisor user or department chair user. User other than student could only write comments or make judgement on the application, but never change its contents.

## Software Quality Attributes

*<Specify any additional quality characteristics for the product that will be important to either the customers or the developers. Some to consider are: adaptability, availability, correctness, flexibility, interoperability, maintainability, portability, reliability, reusability, robustness, testability, and usability. Write these to be specific, quantitative, and verifiable when possible. At the least, clarify the relative preferences for various attributes, such as ease of use over ease of learning.>*

*adaptability*

*Availability: the system only opens to KSU students, advisor, program coordinator and department chair. No other people have access. This is realized by using login page.*

*Correctness*

*Flexibility*

*Interoperability*

*Maintainability*

*Portability*

*Reliability*

*Reusability*

*Robustness: the system should have the robustness against invalid input. In the login page, if user input some information (wrong KSU ID or wrong name) which doesn’t exist, the system should be able to output the error notification and back to the login page again. After login, dropdown list will be used as many as possible to avoid invalid input.*

*Testability:*

*Usability:*

## Business Rules

*<List any operating principles about the product, such as which individuals or roles can perform which functions under specific circumstances. These are not functional requirements in themselves, but they may imply certain functional requirements to enforce the rules.>*

Student can login in the system and fill in a form, then submit or cancel the substitution application.

From advisor angle, there is a file storage of all the applications in the system, advisor could click on any of these application to see the detail content of the application. And advisor could add comments and forward the application to program coordinator/department chair or deny.

Program coordinator could forward the application to department chair or deny.

Department chair reserves the final right to approve or deny, with electrical signature.

# Other Requirements

*<Define any other requirements not covered elsewhere in the SRS. This might include database requirements, internationalization requirements, legal requirements, reuse objectives for the project, and so on. Add any new sections that are pertinent to the project.>*

**Appendix A: Glossary**

*<Define all the terms necessary to properly interpret the SRS, including acronyms and abbreviations. You may wish to build a separate glossary that spans multiple projects or the entire organization, and just include terms specific to a single project in each SRS.>*

**Appendix B: Analysis Models**

*<Optionally, include any pertinent analysis models, such as data flow diagrams, class diagrams, state-transition diagrams, or entity-relationship diagrams*.>

**Appendix C: To Be Determined List**

*<Collect a numbered list of the TBD (to be determined) references that remain in the SRS so they can be tracked to closure.>*