CSC 131
Computer Software Engineering

Instructor: Doan Nguyen Ph.D

Fall 2016

Deliverable #2: Software Req. Specification
Team Name: Interactive Interface

Due date: 10/04/2016

{SECTION 1} SRS – INCREMENTAL VERSION:
1.1 Customer Statement and Requirements

Goals: Our goal for this project is to design a web application that keeps track of student attendance. This web application will be compatible with SacCT in order for professors to utilize the attendance records in grading. Sign-in availability will be limited to those present in the classroom as well as removing the time it takes to get class attendance taken care of.

Problem Statement: Currently, if professors want to keep track of student attendance, they have to interrupt lecture time in order to call each student off of rollcall and proceed to mark whether they are present, late, or even absent. Since topics covered in lecture should be the priority of lecture, it would be essential if there was a more efficient way to take attendance without deviating from the lecture materials.

Proposed Solution: Our web application will help the professor mitigate this process out of the lecture timeframe and allot more time for lecture material. Professors only need to issue an authorization key, in which the students present in lecture will input into the application to automate the attendance process.

We are designing a proposed solution to this existing problem through our web application, where students will have to login into our system. Then they will be allowed to submit their student ID number, as well as the unique authorization key generated by the professor for that lecture session. A receipt will be printed to students who have successfully utilized the web application, notifying them that their attendance has been recorded for this lecture session. For professors, they will be able to receive and modify an xml spreadsheet withholding data entered into the web application in case the instructor needs to manually enter a student into the system. The advantage of having all of the students take self-attendance is to promote showing up to lecture periods and removing the paperwork required for professors in recording rollcall.

1.2 Functional Requirements

Identifier	Priority	Description
REQ_1	5	The system shall be able to accept inputted student ID and professor’s key.
REQ_2	4	The system shall have the ability to limit process inside the classroom only.
REQ_3	2	The system shall be able to generate a confirmation box notifying the student that they entered the professor’s key successfully.
REQ_4	4	The system shall be able to make an attendance sheet on SacCT available for professors.
REQ_5	5	The system shall have the ability to make the attendance sheet editable manually by the professor.

REQ_6	1	The system shall be able to be accessible by other professors.

In order for our application to be successful, our most important requirement is for the system to be able to validate the information that the student has entered, including the location and time of when the student entered the professor's key (REQ_1, REQ_2, REQ_4). Also, in case of internet failure at the classroom or students are unsuccessfully able to input information, the professor must be able to manually input attendance (REQ_5). This is why we have made these requirements to be the highest priority, since they are the basic building blocks of our application.

1.3 Nonfunctional Requirements

Identifier	**Priority**	**Description**
REQ_3_1	2	The system shall be able to validate the successful confirmation within an allotted timeframe of one minute.

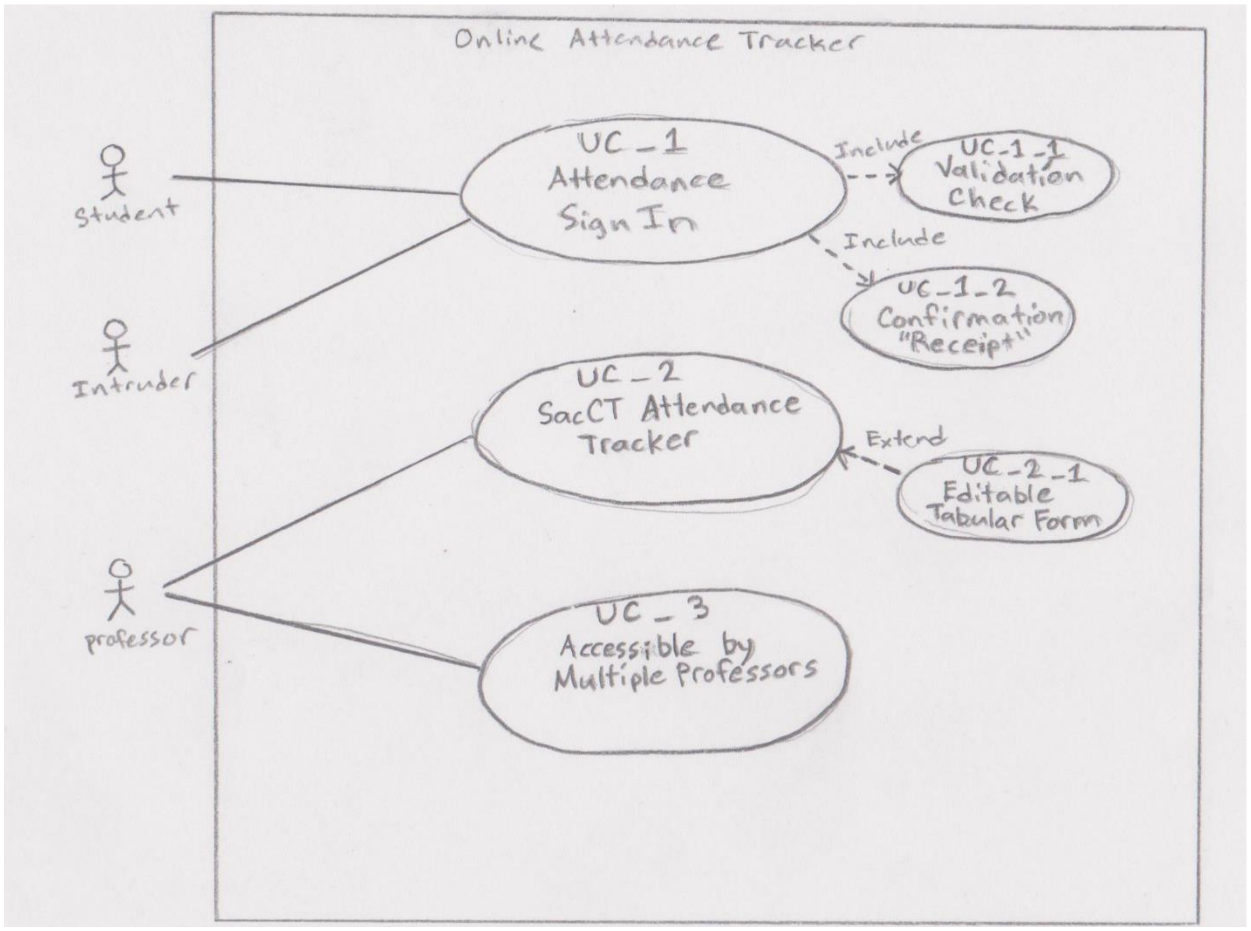
Our nonfunctional requirements stem from creating a positive user experience. They are highly tied with the User Interface, since this is what the user will be interacting with. Users will receive feedback from the system, letting them know when the system is doing work and when it is done authenticating their sign in.

1.4 Functional Requirements Specification

Stakeholders: Because of the nature of this project (being a class assignment for CSC 131), the main stakeholders are the developers: Isaac Israel, Daniel Komac, Justin Rucker, and Cheng Thor. Another stakeholder is our professor, Doan Nguyen, who will be responsible for assessing our application. In this context, Professor Nguyen and our classmates are the customers, since we will be demonstrating our application to them during our presentation.

Actors: During our use cases, we have used three types of users: Student, Intruder, and Professor. A Student is someone who is currently within the classroom and has been instructed by the Professor to input their key into the web application. They will be doing this within the timeframe of the lecture session. An Intruder is someone who is trying to access the web application and input the professor's key from outside the lecture session. A Professor on the other hand has elevated rights, such as viewing the record of student attendance in tabular form and granted editing rights on the attendance sheet.

Use Cases: Below are our use cases. The first figure is the use case diagram, which gives the reader a summary of our use cases. Then, we have a table with more details on what each use case entails.


Online Attendance Tracker
Student
Intruder
Professor
UC_1 Attendance Sign In
Include
UC-1_1 Validation Check
Include
UC_1_2 Confirmation "Receipt"
UC_2 SacCT Attendance Tracker
Extend
UC_2_1 Editable Tabular Form
UC_3 Accessible by Multiple Professors

Use Case #	Use Case Name	Use Case Details
UC_1	Attendance Sign-In	**<Actors>** Student Intruder **<Goals>** To register class attendance. **<Pre-conditions>** (1) Online attendance sheet already created. (2) Attendance key already prepared and provided by teacher. (3) Teacher provides the URL needed to take attendance. **<Post-conditions>** (1) Attendance sheet updated. (2) Student receives confirmation. (3) Student cannot submit another attendance request.

		<Extension Points> (1) Includes UC_1_1: Validation Check. (2) Includes UC_1_2: Confirmation “Receipt”. **<Description>** The student can register to class attendance. **<Basic Flow of Events>** 1) Student goes to the given URL. 2) The web application displays information about what needs to be inputted: Student ID and provided key. 3) The student presses “go” on the student side of the web application. 4) Student inputs their ID and key, with an additional option to put their names, and then clicks the “Sign in” button. 5) Include Validation (see UC_1_1). 6) If validation returns TRUE, the web application updates the attendance sheet at the current date. 7) Include Confirmation (see UC_1_2). **<Alternative Events>** 1) If the student does not have a device that can access the internet, they are to report to the professor so that the professor can manually update the student’s attendance, provided the student gives the professor their ID. 7) If validation returned FALSE, the web application will prompt the student about invalid information. Then restart at Step 3 in **<Basic Flow of Events>**.
UC_1_1	Validation Check	**<Actors>** Student Intruder **<Goals>** To validate user information. **<Pre-conditions>** (1) Student ID and key already inputted. **<Post-conditions>** (1) Student input has been checked for validation. **<Extension Points>**

		(1) Included by UC_1: Attendance Sign-In. **<Description>** This checks the student's inputted information as well as the time and location of the student for validity. **<Basic Flow of Events>** 1) The web application checks the student's ID and key. 2) The web application checks the current time with the class hours. 3) The web application checks the student's IP address with the classroom's IP address. The web application returns with a validation status (TRUE or FALSE).
UC_1_2	Confirmation "Receipt"	**<Actors>** Student **<Goals>** To output a confirmation "receipt". **<Pre-conditions>** (1) The attendance sheet has been updated. **<Post-conditions>** (1) The confirmation "receipt" has been shown by the web application. (2) The student has been barred from making another attendance request for the day for the class. **<Extension Points>** (1) Included by UC_1: Attendance Sign-In. **<Description>** The web application shows the student a confirmation "receipt" with their information. **<Basic Flow of Events>** 1) The web application gathers the student's ID, current date, and class. 2) The web application shows the student the confirmation "receipt" with the following: "<Student Name> of ID<Student ID> has been recorded as attending <Class> on <Date>. Please screenshot this page and save it for future references".

UC_2	SacCT Attendance Tracking	**<Actors>** Professor **<Goals>** To allow professor to upload student attendance to SacCT. **<Pre-Conditions>** (1) Spreadsheet has already been created. (2) Professor is on SacCT. **<Post-Conditions>** (1) Spreadsheet has already been created. (2) Display confirmation that attendance was downloaded by creating a .csv file. **<Description>** Web application lets professor download the spreadsheet file. **<Basic Flow of Events>** (1) Professor will download .csv file from web application. (2) Professor will upload file to SacCT.
UC_2_1	Editable Tabular Form	**<Actors>** Professor **<Goals>** Professor will be able to edit spreadsheet from the web application. **<Pre-Conditions>** (1) Professor is signed into the web application. (2) Professor already has pre-existing spreadsheet. **<Post-Conditions>** (1) Spreadsheet successfully updates. **<Description>** Web application lets professor modify and save changes to the spreadsheet file. **<Basic Flow of Events>** (1) Professor accesses web application. (2) Professor edits spreadsheet. (3) Professor saves changes to the spreadsheet.

UC_3	Accessible by Multiple Professors	**<Actors>** Professor **<Goals>** To allow multiple professors access to the web application and make editions. **<Pre-Conditions>** (1) Have a Google account. (2) Have a unique key for students. (3) Type in the URL. **<Post-Conditions>** (1) Successfully access the web application. **<Description>** Multiple professors may generate their own attendance spreadsheets for use in their lecture sessions. **<Basic Flow of Events>** (1) Professor accesses URL of web application. (2) Professor downloads or edits spreadsheet.

1.5 Traceability Matrix

Below is a traceability matrix, which shows the relationship between the use cases and the functional requirements.

	UC_1: Attendance Sign-In	UC_1_1: Validation Check	UC_1_2: Confirmation "Receipt"	UC_2: SacCT Attendance Tracking	UC_2_1: Editable Tabular Form	UC_3: Accessible by Multiple Professors
REQ_1: Student Attendance Check	X			X	X	X
REQ_2: Attendance Check Limitations		X				
REQ_3: Confirmation "Receipt"		X	X			

REQ_4: SacCT Spreadsheet Access	X			X		X
REQ_5: Excel Access	X			X	X	X
REQ_6: Web Service Available.	X	X	X	X	X	X

<u>{SECTION 2} DESIGN – INCREMENTAL VERSION:</u>

For our system architecture, we are following a widely used pattern for web application: the Model View Controller (MVC). In this architecture, the system is observed as having 3 major parts. The View is the user interface, which is the part of the software that users will be interacting with. In our case, this will be our web UI, which will be the part that users interact with. The controller contains the application logic, which in our case, refers to taking appropriate actions after a student or professor input. A student input if valid, would generate a confirmation box and log the student ID and time of input into a spreadsheet. On the flipside, a professor input grants permission to edit the spreadsheet and gives access rights to upload it to SacCT. Finally, the model handles the data, which in our case will involve the use of REST API in order to read and write data. Figure 1 contains our system architecture.

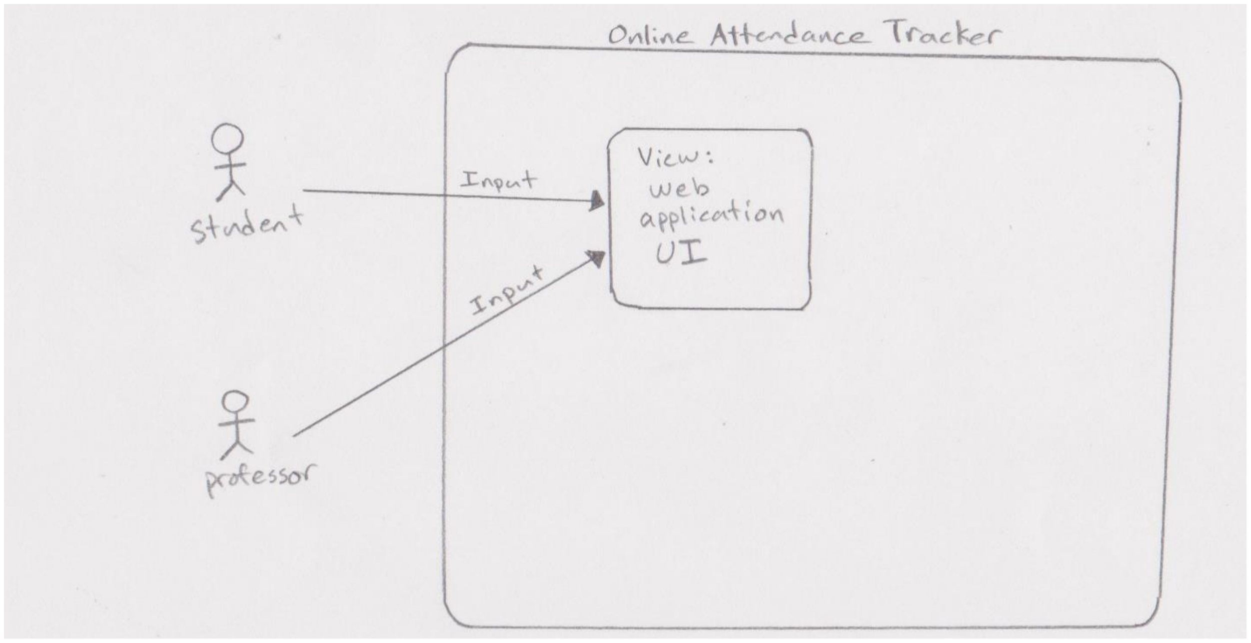


Figure 1: System Architecture (1st Iteration)

{SECTION 3} TESTING – INCREMENTAL VERSION:

As we have only prepared the website's exoskeleton, no use cases have yet to be satisfied with this iteration. Thankfully, we have a greater understanding of where we need to be now and have the experience needed to further pursue our mission in providing the online attendance tracker.

{SECTION 4} CONCLUSION:

In this incremental release, we have delivered an outline of a working web application in which the system will accept student inputted ID numbers and professor keys. None of the functionality is there yet, and the next steps would be accomplishing and finalizing the sign in process. This means UC_1, UC_1_1, and UC_1_2 are going to be worked on for the next iteration to complete the basic attendance tracking functionality.

{SECTION 5} TEAM CONTRIBUTIONS:

Title
Interactive Interface

Team
Isaac Israel, Daniel Komac, Justin Rucker, Cheng Thor

Team Summary Time Log

Team Member	Time (in minutes)	Activities (description)
Isaac Israel	595	● 9/21 Attended small group meeting ○ Installed Python ○ Installed PIP tool for Python ○ Studying Google Sheet API ○ Testing out Google Sheet API for Python ○ Recorded the meeting (via paper) ● 10/1 Worked on web application via HTML ● 10/3 Attended group meeting ○ Discussed plan of action ○ Worked on web application via HTML ○ Developed UC_2, UC_2_1, and UC_3 for the table.
Daniel Komac	240	● 9/21 Attended small group meeting ○ Studying Google Sheet API ○ Recorded the meeting (via paper) ● 10/3 Attended group meeting (Skype)

		○ Offered input and double checked work being done on multiple drafts of deliverable.
Justin Rucker	540	● 9/21 First Draft of Incremental Release SRS Form ○ Formatting Document ○ Studying Google Sheet API ○ Reviewed the small group meeting ● 10/1 Second Draft of Incremental Release SRS Form ● 10/3 Attended group meeting ○ Discussed plan of action ○ Third, Fourth and Final Drafts of Incremental Release SRS Form ○ Multiple Revisions to tables, diagrams, word choices ○ Putting the project into descriptive words ○ Documentation Formatting ○ Leading team to victory
Cheng Thor	515	● 9/21 Attended small group meeting ○ Studying Google Sheet API ○ Recorded the meeting (via paper) ● 9/21 Installed Python ○ Testing out Google Sheet API for Python ● 10/3 Attended group meeting ○ Discussed plan of action. ○ Finalized diagrams for deliverable #2. ○ Developed UC_1, UC_1_1, and UC_1_2 for the table. ○ Double checked Traceability Matrix.
TEAM TOTAL	**1890**	Total time needed to produce this deliverable.

Why are we doing this Deliverable
We are doing this deliverable to give a general guideline as to how our personal project has been progressing throughout these last two weeks. We are learning many things about Python programming, Google Spreadsheets and API, organizational schematics as well as proper documentation of large-scale projects.

Project files for this Deliverable
No project files for this deliverable, just the website in which we have designed our base skeleton to the online attendance tracker. URL: *athena.ecs.csus.edu/~israeli/Test1.html*

Method / Process
The overall process to completing this iteration simply lied in studying the Google API. This helped us discover what was needed to designing the exoskeleton of our system; the user interface. Python is also going to be used for the back-end operations of our web application.

Results
We produced a web application with a unique URL which shows an “About” page, a “Student” page, a “Teacher” page, and a “Home” page. No obstacles have been encountered in accomplishing this project yet. The success was a working web page with buttons that redirect the user (Student/Professor) to the specified pages.