CS1530 Compiled Documentation

Group 15

MILESTONE 1

1.0 Scope (Product overview and summary. Evolves into Sec. 1 of requirements spec)

- It is statistically shown that people perform better when provided a familiar environment to work in. To more effectively draw out potential, our program will provide a way to increase the stability in the classroom and with that, the productivity.

  The program will focus on the seating arrangements of the classroom, allowing the professor to designate a classroom layout and assign seats, or allow the students to choose their preferred seats. With this, students will no longer have to worry about their seats getting taken or accidentally taking someone else's seat, and professors will no longer have scan the classroom for a stray student while taking attendance. This program will hopefully streamline classroom performance as a whole.

  1.1 Functions (What can this product do? Evolves into Sec. 3 of requirements spec)

  - Login for professors and student (allows for the selection of either professor or student)
  - Allows professors freely create and customize classrooms based on their preferences/ their actual counterparts
  - Allows students to choose/reserve their own seats if the professor does not assign them a seat
  - Allows the professor to change/edit student seats if there is conflict
  - Professor can upload a classroom roster/ students will be added to the roster when choosing a seat

- Allows professor to add notes to students (not visible to students)

1.2 Performance (How well can this product do? Evolves into Sec. 4 of requirements spec.)

- Does not involve any major overhead and processing power
- Might take some server memory
- Overall, there should be no problems with any computer running this program

1.3 Limitations (What this product cannot do? Included in Sec. 8 of requirements spec.)

- CSV file upload very specific format
- No exporting seating arrangement formats in csv (Easily fixed by using PrtSc button instead)
- Cannot provide any other in-classroom functions besides seating and making notes

2.0 Tasks (From developer's viewpoint, what are the tasks? For example tasks may include user interface, database manager, data mining, profile management, and so on. Cost analysis is based on tasks.)

- Implement simple website skeleton
- Implement basic database
- Implementing professor and student login
- Create data structures for each student/ professor
- Create separate UIs for student/ professor
- Create interactable and customizable table for professor
- Create interactable table edited by professor for students
- Optimize and streamline website

3.0 Resources

- Because the vision of our product is still in its stages of infancy, it is hard to say what we might need. However, we can say what we have to use right now, and hopefully we will not need much more in the future.

  3.1 Hardware (for development and for deployment)

  - 5 laptops
  - 2 desktops

  3.2 Software (what do you need for the development environment?)

  - SQL
  - Python
  - Javascript
  - HTML
  - Photoshop / paint for mockups and planning
  - A web domain
  - A database program

MILESTONE 2

1. Product Overview and Summary

Essentially what our project wants to do is to streamline the chaos of choosing seats in the classroom. In order to do this, we have decided to create a program to counteract this unorderly business by allowing the professor to set the classroom layout of their choice and

allowing the student to choose their own seats. This will not only make attendance easier to handle but also increase productivity of students by providing them with a consistent learning environment. This program will also allow professors to keep better track of their students.

2. Information Description

2.1. User interface (A Preliminary User manual)

We want the user interface to be easily accessible and interactable to all kinds of students, meaning it should be both simple and intuitive. The interface will not only be color coded, but each marked box (which will determine which seats are reserved, unreserved, or blocked) will have its own pattern so that even people with colorblindness can use the program.

- A green box with a circle in it will represent an unreserved seat
- A red box with an "X" through it will represent a reserved seat
- A black box with dashed lines through it will represent a blocked seat and also walls and desks

Student side:

There will be a login prompt right as the student visits the website. Afterwards, the student will have access to their list of classrooms where their seat is reserved. The student will also be able to add a new classroom to join, if needed.

If the student wants to add a new classroom, prompt them to enter in a room code to join.

If the student wants to access a classroom they already have signed up for, they should click on the classroom name they would like to access and click on a seat that is unreserved (green box with a circle in it) to reserve the seat.

If the seat has not been taken, the program will prompt the student for their name and email, so the professor or other students know that the seat is taken by the specified statement and how to contact them.

If the student wants to unreserve a seat they should click on the seat they've reserved before. They will be given the option to change their name, email, or unreserve their seat.

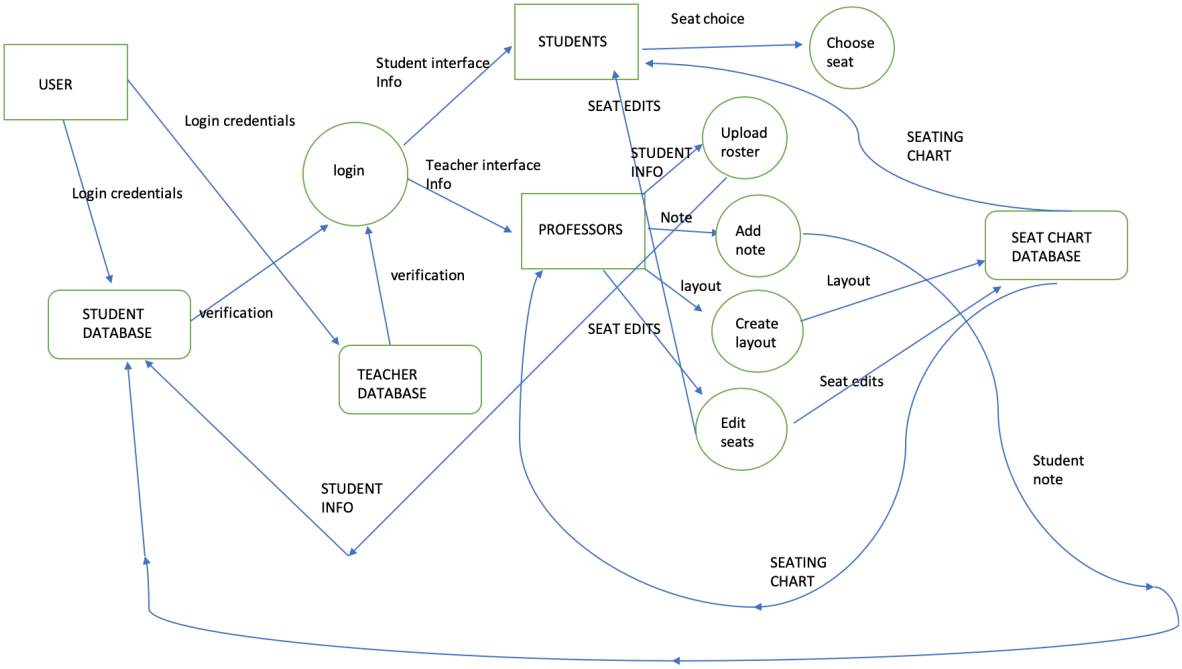
Professor side:

When accessing the site the professor will be prompted to sign in. Afterwards, they will be given an option to create a new classroom or access a classroom they've already made.

If they make a new classroom, they will be given an interactable interface with an adjustable table and boxes that change the cursor tool when clicked on. This way, the professor can easily customize the table given to resemble the likeness of their classroom.

If they access an already made classroom, a grid will show up (similar to when a classroom is made) and it will display all the seats and if any students reserved them. The professor can click on the seat to reveal more information about the reservee. This mode still allows editing with the click of an edit button.

2.2. High level data flow diagram



3. Functional Description

3.1. Functions (you should specify functions using IC cards)

IC Card IC Name: choose seats

Description: students choose seats
Interaction Pattern:

By Myself no Interacton
My Task: choose a seat
Time Critical Condition: none
Name of Other IC: none
Message to Other IC: none
Other IC's Task: none
Card 1of 1 (If necessary please use several IC cards to describe an IC)

IC Card IC Name: add notes

Description: add notes to a student
Interaction Pattern:

By Myself no Interacton
My Task: add notes to a student
Time Critical Condition: none
Name of Other IC: student database
Message to Other IC: fill in notes in student database
Other IC's Task: none
Card 1of 1 (If necessary please use several IC cards to describe an IC)

IC Card IC Name: create layout

Description: create a layout of the classroom
Interaction Pattern:

By Myself no Interacton
My Task: create a layout of the classroom
Time Critical Condition: none
Name of Other IC: seat chart database
Message to Other IC: fill in seat chart database
Other IC's Task: none
Card 1of 1 (If necessary please use several IC cards to describe an IC)

IC Card IC Name: Login

Description: Log in to the system
Interaction Pattern:

By Myself with Interaction

My Task: Log into the system
Time Critical Condition: none
Name of Other IC: student or teacher database
Message to Other IC: authenticate my credentials
Other IC's Task: direct to student or professor landing page
Card 1of 1 (If necessary please use several IC cards to describe an IC)

IC Card IC Name: upload roster

Description: professor uploads class roster
Interaction Pattern:

By Myself with Interaction

My Task: upload class roster
Time Critical Condition: none
Name of Other IC: student database
Message to Other IC: populate student database
Other IC's Task: none
Card 1of 1 (If necessary please use several IC cards to describe an IC)

IC Card IC Name: edit seats

Description: edit seat layout of a classroom
Interaction Pattern:

By Myself with Interaction

My Task: change seat layout of a classroom
Time Critical Condition: none
Name of Other IC: students
Message to Other IC: update your seat position
Other IC's Task: change their seat position
Card 1of 1 (If necessary please use several IC cards to describe an IC)

3.2. Processing narrative

(The processing narrative describes in words what the functions do. For each function, there is a paragraph (or several paragraphs if necessary) describing this function. For example, the take-test function of a distance learning system allows a student to take a test, by first logging in, then entering course information, and finally taking an online test. The test scores are stored in the student record.)

- Login ("user" becomes student or professor, or loop back to user if invalid credentials)
    - User is prompted to login as a student or a professor. The user is then matched to the student or professor database depending on which option they selected with their login credentials.
    - If the login succeeds, the user is taken into their classroom list to further proceed
    - If the login fails, the user is taken back to the login screen with an error message
    - If user does not have a login yet, they will be able to sign up, and the username and password will be stored in the student database
- Classroom selection
    - User is asked to choose the classroom they want to access (or if they want to access a new classroom).
    - If user wants to access a new classroom, they will be prompted to enter in the classroom code, which is stored in the seat database.

        - If the classroom code is invalid, user is taken back to classroom selection page
        - If classroom code matches one in seat database, user moves to the choose seats page
- Choose seats
    - User is prompted to choose an available seat in the classroom they have accessed.
        - If seat has been taken in between them accessing the classroom layout and reserving a seat, return to classroom layout page
        - If seat is available, prompt user to enter name and email
            - Once successfully reserved, the name and email the user entered will be saved in the student database, along with user's login data
            - Seat will be saved as reserved with student's name and email into the seat database
- Upload roster
    - If user login is a professor, after classroom selection, user has more in depth options.
        - In order to make seating easier, professor can upload a roster in XML format
            - If formatting fails, user will return to the classroom layout page with an error message
            - If formatting succeeds, user will return to the classroom layout page with a roster and names that the professor can click and drag to seats, if needed.

- Add notes
    - Professor user will be able to click on a reserved seat to add notes. These notes will only be able to be seen by professor users
        - If the note is too long, return an error notification
        - If the note successfully saves, save note to seat database along with the seat
- Create layout
    - After professor user selects a classroom, they will be provided an expandable and customizable table to change and design to their tastes.
        - Once the table is finished being designed, the user can save it. The table will then be saved in the seat database.
    - If a professor chooses a classroom with a created layout already, the professor can still edit the layout to change anything that needs to be fixed.
        - Once the table changes are finished, the user can click save to update that specific classroom layout to the seat database.
- Edit seats
    - After professor user selects a classroom, they will be able to access the classroom layout including the seats that are reserved and unreserved. The professor can:
        - Kick a student out of a seat
        - Move a student to another open seat
            - Upon performing one of these actions and clicking save, the seat database will update the seat and user information.

3.3. Design constraints

(The design constraints specify what the functions cannot do. For example, the take-test function of a distance learning system supports only multiple-choice type of tests, and other type of tests are not supported.)

- Login
    - Unable to provide security questions
    - Unable to provide two factor authentication
    - Unable to use anything other than username / password
- Classroom selection
    - Unable to select more than one classroom at the same time
    - Unable to remove classrooms (yet)
    - Unable to upload external tables/ classroom templates
- Choose seats
    - Unable to negotiate in-program with someone who has a seat you want
    - Unable to reserve more than one seat
    - Unable to select a seat that is blocked or not specified
- Upload roster
    - Unable to upload roster in anything besides XML format
- Add notes
    - Unable to add longer notes.
    - Unable to allow students to see notes
- Create layout

    - Unable to curve classrooms
    - Unable to create very specifically detailed classroom layouts
- Edit seats
    - Unable to edit multiple table squares at the same time

Cost and Schedule:

4.1 Labor Cost (LOC Technique):

Function	Optimistic	Most	Pessimistic	Expected	Deviation	Line/Month	Months
Interface	200	600	800	566	100	400	1.1
Login	50	100	120	95	12	50	1.1
Choose Seats	100	150	175	145	13	80	1
Roster Upload	50	100	120	95	12	50	.5
Add notes	50	100	130	96	13	50	.25
Edit seats	50	70	100	71	8	40	.25
Total	500	1120	1445	1068	158	670	4.2*

*since we have multiple people, many functions will be developed at the same time and will be finished a lot sooner than 4.2 months (see schedule)

4.2 Labor Cost (Man-Hours Task Technique) :

Function	Requirement	Design	Coding	Test	Subtotal
Interface	20	60	60	60	200
Login	20	50	80	50	200

Choose Seats	20	60	50	50	180
Roster Upload	10	20	40	20	90
Add notes	10	10	10	10	40
Edit seats	10	20	10	10	50
Total	90	220	250	200	760

5. Schedule:

week	1	2	3	4	5	6	7	8	9
interface	o				x				
login		o				x			
Choose seats				o			x		
Roster upload					o		x		
Add notes						o	x		
Edit seats						o	x		
testing							o		x

MILESTONE 3

2.3 Data Structures

1) Student (<u>sid</u>, fname, lname, s_username, s_password_hash)

2) Professor (<u>pid</u>, fname, lname, p_username, p_password_hash)

3) Courses (cid, title, semester, pid, classroom_id, section)
4) Course_taken (cid, sid, grade, notes)
5) Classroom (classroom_id, classroom_name, dimensionX, dimensionY)
6) Seat (classroom_id, cid, X_coordinate, Y_coordinate, status, sid)

2.4 Data Dictionary

1) Student:
    a) sid: Data type: INT. Stands for Student ID. It is the primary key of the Student table. It identifies the students, and it is generated by the software.
    b) fname: Data type: char(15). Contains the first name of the student.
    c) lname: Data type: char(15). Contains the last name of the student.
    d) s_username: Data type: char(16). It is the string that the students create for login.
    e) s_password_hash: Data type: INT. It is the hash code of the student's password for login.
2) Professor:
    a) pid: Data type: INT. Stands for Professor ID. It is the primary key of the Professor table. It identifies the professor, and it is generated by the software.
    b) fname: Data type: char(15). Contains the first name of the professor.
    c) lname: Data type: char(15). Contains the last name of the professor.
    d) p_username: Data type: char(16). It is the string that the professors create for login.
    e) p_password_hash: Data type: INT. It is the hash code of the professor's password for login.

3) Courses:
    a) cid: Data type: INT. Stands for course id. Primary key of the Courses table. It identifies the courses, and it is generated by the software. Note that for the same course in different semesters/different sections, they have different cid.
    b) title: Data type: char(40). It is the title of the course.
    c) semester: Data type: char(12). It denotes which semester the course is in. It has a format of "yyyy Spring/Summer/Fall". For example, "2019 Fall".
    d) pid: Foreign key from Professor table, denoting the professor of this course.
    e) classroom_id: Foreign key from Classroom table, denoting the classroom of the course.
    f) section: Data type: INT. It indicates different sections of the course in a semester.
4) Course_taken:
    a) cid: Foreign key from Courses table, denoting the course that the student takes.
    b) sid: Foreign key from Student table, denoting the student who selects this course.
        i) Cid and sid consist the primary key of the table Course_taken
    c) grade: Data type: char[2]. Contains the letter grade of the course. For example, "A-".
    d) notes: Data type: TEXT. It is generated and uploaded by the professor to denote each student's behavior and performance. The notes are only visible to the professor who teaches this course.
5) Classroom
    a) classroom_id: Data type: INT. Primary key of the Classroom table. It identifies the classrooms and is generated by the software.

b) classroom_name: Data type: char(20). Contains the name of the classroom. For example, “Sennott Square 5129”

c) dimensionX: Data type: INT. Indicates the width of the classroom, in the unit of foot.

d) dimensionY: Data type: INT. Indicates the length of the classroom, in the unit of foot.

6) Seat:

a) classroom_id: Foreign key from Classroom table, denoting the classroom of the seat.

b) cid: Foreign key from Courses table. It indicates which course the classroom is for.

c) X_coordinate: Data type: INT. Denotes the position of the seat in the classroom.

d) Y_coordinate: Data type: INT. Denotes the position of the seat in the classroom.

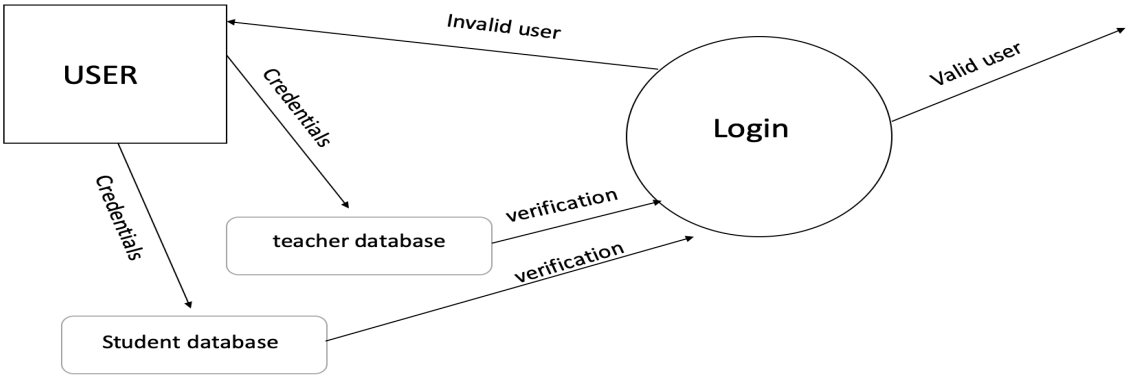
e) status: Data type: char(10). Indicates the different status of the seat. For example, “taken”, “empty”, “blocked”, “reserved”.

f) sid: Foreign key from Student table, denoting the student who takes the seat.

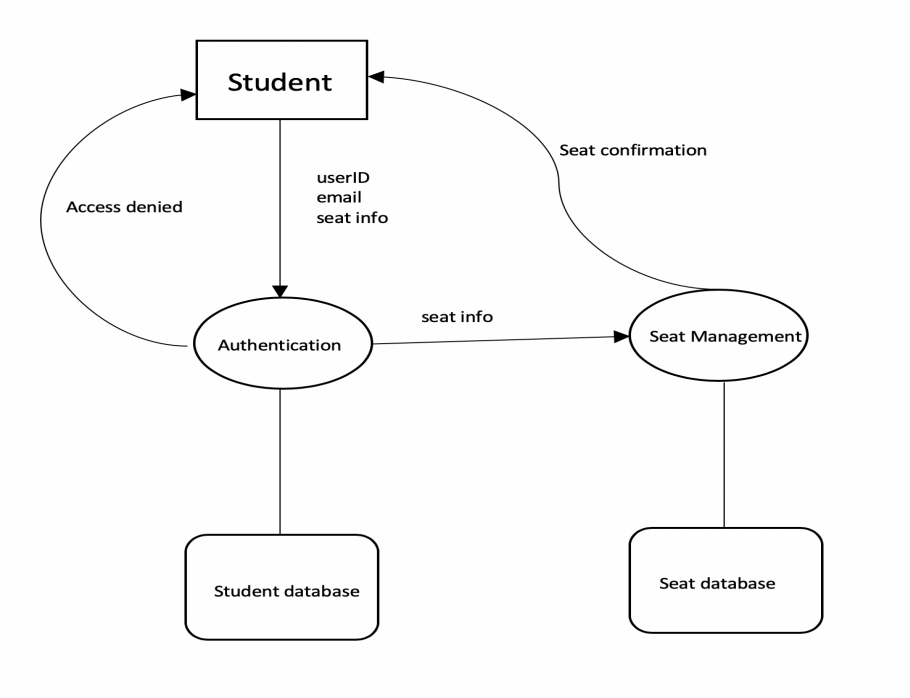
3.4. Diagrams

(There should be a detailed data flow diagram for each function)

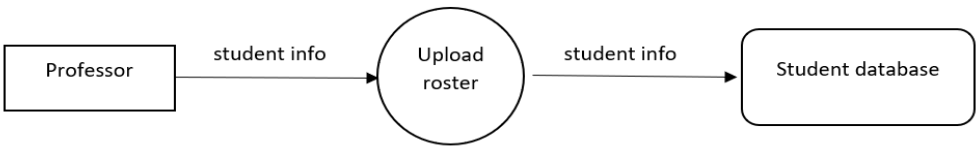
- Login



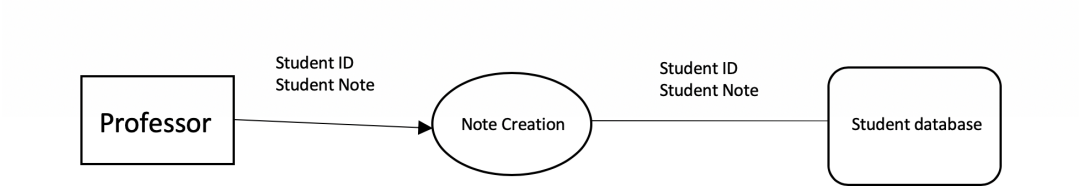
- Choose seats



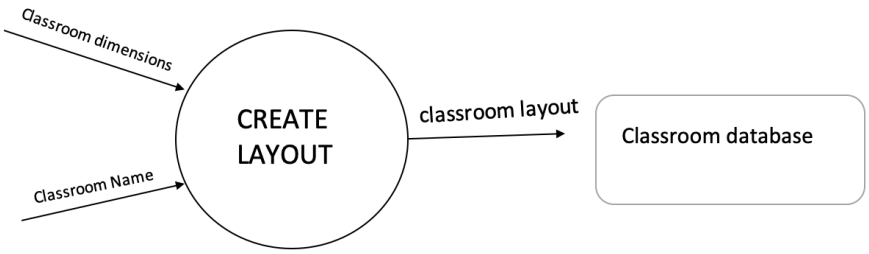
- Upload roster



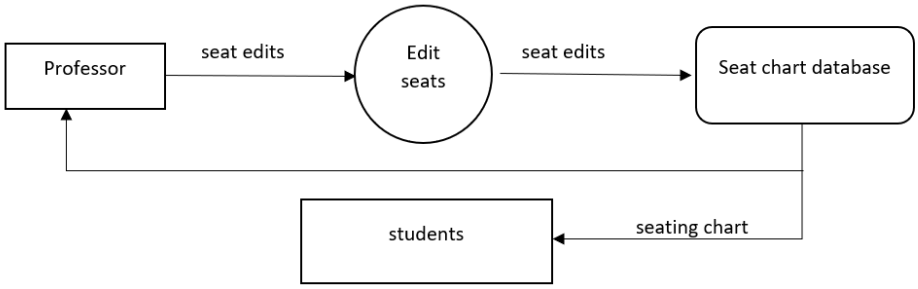
- Add notes



- Create layout



- Edit seats



4. Performance requirements

(As an example, the system should handle at least two hundred

concurrent users, five hundred thousand user profiles and

be able to process a job placement inquiry in less than one second.)

- System should be able to handle at least 100 concurrent users (traffic will be highest when the semester starts)
- System should be able to store and return data from 4 databases

    - Student database should be able to store at least 5000 profiles
    - Professor database should be able to store at least 1000 profiles
    - Classroom database should be able to store at least 3000 classroom layouts
    - Classroom directory database should be able to store 6000 classroom lists
- Data from databases should be returned to users in no less than a second

5. Exception conditions/exception handling

(For example, if a company has a crash job and suddenly needs to

hire five hundred temp workers, how to deal with it?)

- If there is an issue with the domain, we must quickly secure a new domain at any cost or have a backup domain just to start with
- If there is a sudden issue with the databases, we may need to have a big database as backup, which will hold all the information. This will be inefficient to fetch from, since it is not split optimally.
- If a programmer suddenly gets sick, the only thing we can do is increase effort, since we cannot increase cost.

6. Implementation Priorities

(As an example, due to resource (time and personnel) limitation

we will first implement the core system supporting

the following basic functions: (list).

If there are additional resources (time and personnel)

we will implement the following functions: (wish list).

For the final product we will implement all the basic functions

plus some functions in the wish list by iteration following

agile incremental approach.)

Due to time and personnel limitation, our list of priorities will look something like this:

These are essential functions that must be added:

1) Securing a domain:
    a) We will first try to secure a domain so that we may test our program and also have something to link our work to.
2) Creating databases
    a) We will try to create the databases first so that we can have a place to store all that data that we will be testing and storing later on. This is the very skeleton of our project and we cannot continue without it. Therefore it is integral to get this done first so we may continue with the other features of the project
3) Create login
    a) In order to link information such as classrooms, seats, and classroom templates, we must be able to create profiles with encrypted passwords first. This should not take long but it is important
4) Create Classroom Layout Customizer
    a) This will be the bulk of the work. Since this is basically the entire purpose of our project, we must get to this point as soon as possible after taking care of the absolute essentials. This can be interchangeable with " 3) Create Login".
5) Create Classroom Directory
    a) This will be used to link up a saved classroom table with a saved login profile so that a user may access their classrooms. This is essential for navigation and must

be finished right after the layout customizer so that we may access stored layouts faster to see if they work.

6) Create Seat Reservation Function
    a) A sub-database for seat reservations will be linked to the classroom layout. This is the second essential part of our program and must be working after all of the essential functions it needs are finished, but should not take too long by itself to complete.
7) Generate classroom codes
    a) A classroom code generator will be created to students may type in the code and be added to the classroom to perform seat reservation. This will complete the basic functionality of our program.

These are the functions that we will add if we have extra time:

1) Classroom roster
    a) The professor may upload a CSV file and assign seats based on what shows up. The professor may also input names manually. This will be a very nice quality of life for all parties involved especially if something goes wrong and a student cannot access the classroom or sign in. This should not take long to implement.
2) Student notes
    a) The professor will be able to add notes to a student's seat. This will allow to professor to identify students better. Should not take long to implement.

7. Foreseeable modifications and enhancements

- We may change what databases we need and add extra functions based on what we think may be better for the user interface. As such, we are keeping our “needed functions” to more of a minimum while we flesh out our project more.
- We are thinking about adding basic classroom templates as an option when a professor decides to create a classroom
- Adding more roster upload formats other than only csv files
- In-program messaging for exchanging seats?

8. Acceptance criteria

Functional and performance tests

Documentation standards

(what tests will you do to accept this product? What documents will you deliver?)

- Domain must be properly functioning and should not go offline
- User traffic performance tests must be passed
- All databases must store information correctly and return them quickly
- Classroom layout tables must be stored efficiently and returned properly
- Fetch requests should not take more than a couple of seconds
- Classroom codes that are generated should allow proper access to the correct classrooms
- Login must work properly and passwords must be hashed
- Classroom customization must be interactable and easy to use properly
- Classroom directory must return all of a user’s classrooms and allow them to be accessed
- Documentation must be easily traceable
- Documentation sections must be easily distinguishable and searchable

9. Sources of information

(This section includes documents from software vendors,

outsourcing companies, and most importantly customers.

For example the customer's company has a human resource

document and our system is designed in compliance to

that document. For the class project, you can cite

the first kind of documents.)

- https://docs.python.org/3/ python manual
- https://www.w3schools.com/tags/ HTML manual / useful functions
- https://devdocs.io/javascript/ javascript manual
- https://developer.mozilla.org/en-US/docs/Web/CSS CSS manual
- https://www.domain.com/domains/ to secure domains
- https://cstack.github.io/db_tutorial/ building database
- https://www.lucidchart.com/pages/database-diagram/database-design designing database
- http://anh.cs.luc.edu/handsonPythonTutorial/graphics.html creating graphics
- https://www.tutorialspoint.com/python/python_gui_programming.htm creating UI
- https://wiki.python.org/moin/GuiProgramming creating UI
- https://docs.oracle.com/cd/B19306_01/server.102/b14200/statements_5005.htm Linking Databases

10. Revision history

(Each entry shows the date, short description and responsible person for each revision)