**Waterfall Method**

2.1.Waterfall-model.eps

The steps below follow the model above.

Requirements:

* Database w/ capability to support 100 students, 10 instructors & 1 admin
  + Student – can register, see available courses & their own schedule
  + Instructor – can see available courses & their course roster
  + Admin – can see everything & edit courses/users/schedules
* Database of courses
* Multiple semesters capability
* Print-out of schedule
* Scheduling Preferences
* MINIMAL BUGS (must be tested thoroughly)

System and Software Design:

Creating a somewhat basic GUI would provide the end user of this program a clean interface for them to perform whatever tasks they need to be done. To ensure that this is working properly, one of the first things that needs to be set up is the database containing all the various users. From there you can code in a login screen where they must provide a username and password, that way different students aren’t allowed to register another student for a random class. Depending on what their login credentials are (student/instructor/admin) will determine what the next page will show. If a student/instructor logs in, they will be immediately prompted to enter a semester (Fall/Spring/Summer) and year for them to edit.

For students, after selecting a semester, they will be given 4 choices; Register for course, See available courses, View schedule and Print schedule. Selecting “Register for course” will prompt them to enter a CRN, determine whether they can join it or not and then ask if they would like to enter another one. Selecting “See available courses” will list all available courses, their CRN and how many spots are left in the class. “View schedule” will print out the students schedule to the GUI while “Print schedule” will export the users schedule to an external file for printing.

For instructors they will be able to see available courses just like the student’s class can, an option for “View course roster” and “View course schedule”. When they select their course roster it will display what class(es) they are teaching, the name of the student and their username. After displaying this list it will prompt “Would you like to print this schedule?” and if so it will output to a separate text file. Just like the student class the instructor will also be able to view/print their course schedule in the same way.

For the admin they will be given the same prompt after logging in just like the other classes. The difference is the addition of new options; Add/remove course from database, add/remove user from database, add/remove user from course and search/print rosters/courses. For adding/removing items from a database, there will be 2 options that are almost identical. One for courses and one for users. It will prompt the user for the course/user they want to add/remove and then do so. It will be similar for adding/removing a user from a course except you first choose a user, and then from there it will list the courses they are a part of. Then you can choose which course you would like to add/remove. For printing rosters/courses, they will both be under the “Print” option. Then the admin can type in which course they want to look at and then choose whether to print that roster. If you just want the courses, the admin will type in “courses” to print all the courses and their CRN.

Implementation and Unit Testing:

This step will overlap with the previous step since there will be bug testing while implementing the code described. It’ll follow the basic steps of typing a section of code, running it to see if there’s any bugs, solving any bugs if there are and then repeating or just repeating if there are no bugs.

Integration and System Testing:

This step is a bit related to the previous one, except now the code should be fully fleshed out. We would want to fill the database with a few users in each type along with inputting some courses into the database. From there you can test out the system itself to see if it works properly and then return to the previous step if there are any bugs.

Operation and Maintenance:

By now the system testing should be done and bug free. Now we can fill the databases with all of the users that are required to and continue to provide support if any other bugs appear.

**Incremental Method**

2.2 Incremental-dev.eps

The steps below follow the model above.

Outline Description:

We want a system that has the same requirements as found in the waterfall method.

Phase 1:

1. Build a basic function that will verify classes are working properly
2. Build database that will contain all users

Phase 2:

1. Build User class with functions that are used across all 3 user types
2. Build Student class using inheritance, add extra functions (add/drop course)
3. Add student to database for function testing
4. Test the functionality of users, ensure that entering random strings doesn’t break code, etc

Phase 3:

1. Build Instructor class using inheritance, add extra functions
2. Add instructor to database for testing purposes
3. Test the functionality

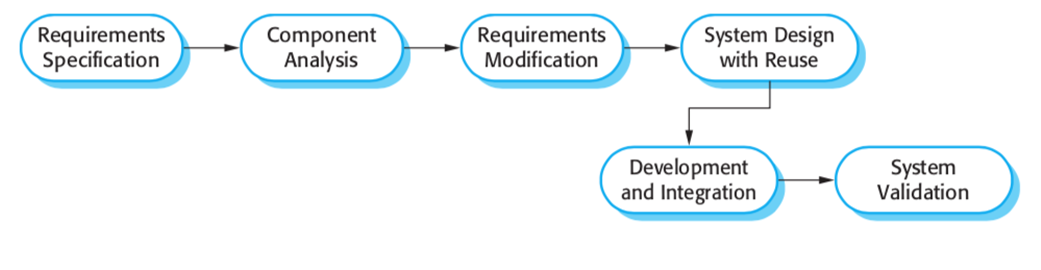
Phase 4:

1. Build admin class using inheritance, add extra functions
2. Add admin to database for testing purposes
3. Testing functionality

Phase 5:

1. Build GUI to make it more presentable and easier to follow
2. Test any sort of bugs for the GUI
3. Re-test all of the databases/users to ensure that they still work properly with the new information.

**Integrate and Configure Method**

****

The steps below follow the model above.

This model will follow the same requirements as the previous two. If we want to integrate some sort of outside code to use in order to save us time it would make sense to analyze exactly what would be the hardest task in this project. Considering that our first assignment was creating the user classes, it’s not very difficult getting that to work. It’s the same thing with the main code that generates a GUI. So, that leaves us with creating the database. I happened to find a good source someone had already done creating a student database using SQL ([freeCodeCamp/learn-sql-by-building-a-student-database-part-1 (github.com)](https://github.com/freeCodeCamp/learn-sql-by-building-a-student-database-part-1)). This code can be easily manipulated to fit our own needs and saves a significant amount of time. So, once the classes have been created this code can be implanted into the program. From there it would just be some bug testing to ensure that everything is working.