Problem Statement

A university needs a software system to manage students, instructors, and an administrator. The system allows users access and manage courses by updating schedules and user information for multiple semesters.

Requirements Engineering

1. Feasibility Study
    - Does the technology exist to build the system? – yes. The system needs classes, objects, a database, and a user interface. (graphical/ text based)
    - Does it fit the budget? - yes

    Conclusion - project is feasible, and we can continue.

2. Requirements elicitation
    - We have examined other systems such as Wentworth Institute of Technology's Leopard Web. From this, some requirements are
        - Enter a user ID to access the system
        - Allow all users of the system to view/search available classes.
        - A user-friendly interface
        - The system must be customizable and flexible to manage multiple semesters

    - Get specifications from the intended users - ask them to submit a document, meetings, send surveys, etc

3. Requirements Specification
    - All users should be able to view the courses that are available for multiple semesters.
        - display a table for the CRNs, Course Names, Times, and Instructors.
        - display the semesters that the courses are taught in.
        - print their own information.

    - The administrator should be able to add courses and users to the system, assign or remove students from courses, and view all information, rosters, and scheduling information.
        - manage and update student and staff information
        - view student, staff, and class information
        - assign courses to students

- The instructors should be able to see available courses, their own assigned schedule, and their course roster(s).
    - display all courses that the Admin created.
    - display a list of the classes that they are teaching.
    - search and display rosters per class per semester.

- The students should be able to search for courses, add or drop courses from their schedule and view their schedule.
    - search and display tables of all courses by semester
    - update course schedule per semester
    - view their current schedule along with others that they create

4. Requirements Validation
    - Allow the system users to check and update the requirements from step 3 above.

Design and Implementation

- Architectural Design - identify the high-level components. Our system: classes and objects, functions, a database, and a user interface.
- Interface Design - this is connecting the high-level components above.
- Component Design - classes and objects (Users - students, instructors, administrator). This is where you design the user interface.
- Database Design - choose the number of tables and columns per table in your database.

Course Table, User Tables with all information from Requirements Specifications

1.Architectural design– high-level components: classes and objects, functions,database, and user interface.

2.Interface design–[how would you connect the above components?].

3.Component design –classes and objects (for example, courses class, user class, student class, instructor class, and admin class. User interface design (text-based).

4.Database design –User table – first name, last name, ID, user description (trainer, vet, or vet tech), animals you work with, and preferred working days.

User Table - first name, last name, user ID, position (student, instructor, admin).

Student Table - first name, last name, user ID, field of study, current courses and instructors per semester.

Instructor Table - first name, last name, user ID, field of study, courses to be taught,

Admin Table - first name, last name, user ID, instructors managed, and students managed.

Course Table - CRN, Course Name, times, instructor teaching course, semester the course is taught in, students in the class.

Software Validation

1. Component Testing (unit testing) - test as you go - all components must be tested.
2. System Testing - testing the system after integrating the components (can also be done as you integrate piece by piece.)
3. Acceptance testing -
   1) allow the user to use the system as they would normally and/or
   2)acquire realistic data from the user to test the system.

Software Evolution

Modify the system based on changing user needs over time or any bugs found.