**Applied Programming Concepts**

**Assignment #3 – Process Models and Version Control**

Hank Pham

06/10/2021

**#Kevin Huang**

* **Waterfall:**

**Spec:**

**#All requirements are carefully described-Kevin Huang**

* Database of users (should be able to hold 100 students, 10 instructors, 1 admin)
* Database of courses (CRN, course name, times, instructor)
* Three types of users:
* student – can register, can see available courses and their own schedule.
* instructor – can see available courses and their own course roster.
* admin – can see everything, can edit courses/users/schedules.
* Should include multiple semesters, print-out of schedule, scheduling preferences.
* Have a GUI on three different levels: Admin, Student, and Instructor.

**Design:**

* Programming Language: Python, SQL
* Classes attributes:
* Student – First name, last name, student ID, graduation year, major/department, email.
* Instructor – First name, last name, employee ID, title, department, email, year hired.
* Administrator – First name, last name, employee ID, title, email, office number
* Course – Title, CRN, department, instructor, time, day(s) of the week, semester, year, credits
* Create classes for student, professor, and admin:
* Inherent first name, last name, ID attributes.
* Each class has different level of permission and different methods.
* Databases:
* Users: admin table, student table, instructor table
* Courses
* Rosters
* GUI:
* Admin: add courses to the system, remove courses from the system, add/remove users, force student in or out of course/rosters, search/print rosters/courses
* Student: search courses, add/drop courses, print schedule
* Instructor: print schedules/rosters, search courses

#**Classes have specific information and the GUI also meets the requirements-Kevin**

**Implementation:**

* System will be developed in Python for Windows 10 users.
* Follow the 80/20 method where completes 80% of system to fufill the basic functionality first before perfecting the last 20% for great user experience.
* Develop in the order:
* Create classes for each type of use.
* Create databases.
* Integrate databases to the classes.
* Create a GUI.
* Integrate functionality into the GUI.

**Testing:**

* Follow the 80/20 process, at each step, the system will be tested thoroughly for basic functionality before testing abuse cases.
* Validate databases.
* Test the GUI from different user’s perspective (admin, instructor, student).

#**What will the operation and maintance be like? Does the system require updates in the future and what if there are issues?-Kevin**

* **Incremental Development:**

**Initial Version:**

* Build and test admin class first. The admin class should have all the functions that will be included in Instructor and Student classes.
* Create simple databases of users, courses, and rosters.
* Database of users (should be able to hold 100 students, 10 instructors, 1 admin)
* Database of courses (CRN, course name, times, instructor)
* admin – can see everything, can edit courses/users/schedules.
* Integrate admin class with databases.
* Test thoroughly.

**Intermediate Version:**

* Build and test student and instructor classes based on the admin class.
* Integrate the databases with the classes.
* student – can register, can see available courses and their own schedule.
* instructor – can see available courses and their own course roster.
* Test all 3 classes fully and thoroughly.

**Final Version:**

* Create a GUI that works with the admin class first before expanding it to other two classes.
* Test each version of the GUI making sure each respective class can only have access to their own functions.

**#Maybe check to see if the database is collecting the data correctly as well?-Kevin**

* **Integrate and Configure:**

**Requirements Specification:**

* Database of users (should be able to hold 100 students, 10 instructors, 1 admin)
* Database of courses (CRN, course name, times, instructor)
* Three types of users:
* student – can register, can see available courses and their own schedule.
* instructor – can see available courses and their own course roster.
* admin – can see everything, can edit courses/users/schedules.
* Should include multiple semesters, print-out of schedule, scheduling preferences.
* Have a GUI on three different levels: Admin, Student, and Instructor.

**Component Analysis:**

* Student Registration System – Software Design Specification by Faculty of Applied Science at Rajarata University of Sri Lanka ([https://web.csulb.edu/~mopkins/cecs493/sdsfinaledit-140325142607-p HYPERLINK "https://web.csulb.edu/~mopkins/cecs493/sdsfinaledit-140325142607-phpapp01.pdf"h HYPERLINK "https://web.csulb.edu/~mopkins/cecs493/sdsfinaledit-140325142607-phpapp01.pdf"papp01.pdf](https://web.csulb.edu/~mopkins/cecs493/sdsfinaledit-140325142607-phpapp01.pdf))
* Course Registration System by mohamedirfansh (<https://github.com/mohamedirfansh/Course-Registration-System>)

#**Great citations on existing programs-Kevin**

**Requirements Modification:**

* Student Registration System is a very comprehensive handbook on building a registration system with multiple sequence diagrams for system design. But the scope must be lower for the project due to time constraints.
* Course Registration System by mohamedirfansh is a basic course registration system that includes many necessary functionalities. However, class instructor needs to be added. The code was written in Java, so it needs to be converted into Python.
* Both systems don’t have a GUI.

**System Design with Reuse:**

* The basic classes Admin, Student, and Instructor can be expanded in the future with added functionality.

#**How will the program be integrated will the programs mentioned above? Be more specific about the additions that are needed.-Kevin**

**System Validation:**

* System needs to be tested thoroughly to fit the needs, goals, and scope of the project.