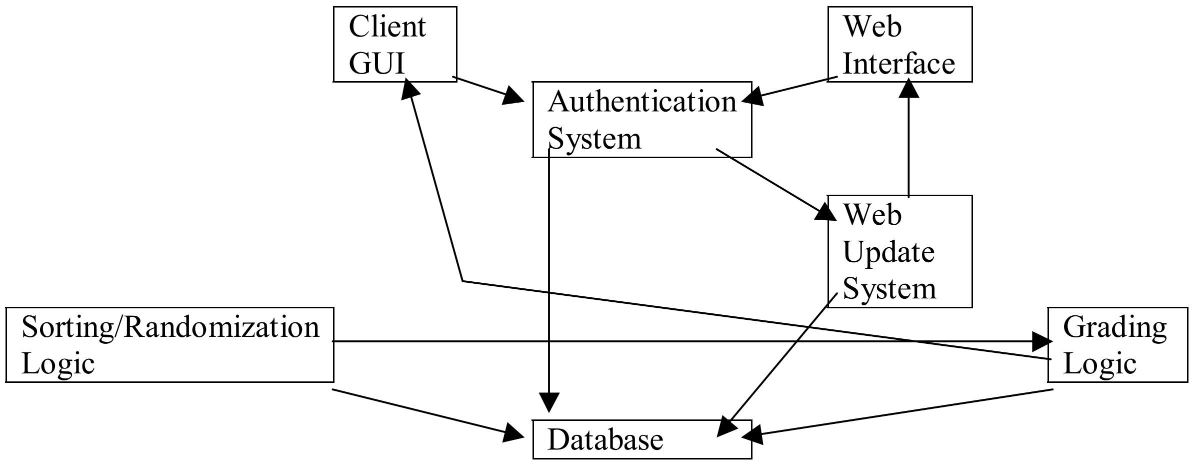
CS190 Specifications Document: Virtual Teacher's Assistant
Author: R. J. Scott McKenzie <rmckenzi>

Description:

Before the creation of student TAs as hired labor, professors had to do much in the way of busywork in order to keep a class running smoothly and efficiently. With the institution of TAs, however, the professor is free to teach while the TAs handle the administrativa of grading smaller assignments, updating the course webpage, et cetera. However, the demands of today's society and various technological advances have allowed us to move forward in removing many of the burdens of work from the TA (or at least make them easier). To that end, this program is an organizational tool which allows for the more mechanical parts of the job to be done easier.

System Model Diagram:



Client GUI: The main visual interface of the program. Contains tabbed windows with sub-windows for efficient organization of data.

Web Interface: An HTML-based secure website designed for the student users to view the information without having access to the professor-side client.

Authentication System: Encryption, decryption, and access restriction in order to determine what functions of the program are modifiable (i.e. if you are a TA, then a Head TA's actions have priority over yours, and the professor has the top priority)

Web Update System: A text-to-HTML parser for adding repetitive types of information (assignments, messages, etc) in a easy-to-use fashion, instead of using raw HTML to add info.

Sorting/Randomization Logic: A subset of the grading tools, required for interpreting rules used for randomly assigning assignments to graders.

Grading Logic: The contents of this area would be mainly methods for accessing the database of grades, adding, removing, and editing grades, as well as interpreting numerical grades according to customizable scale/curve schemas.

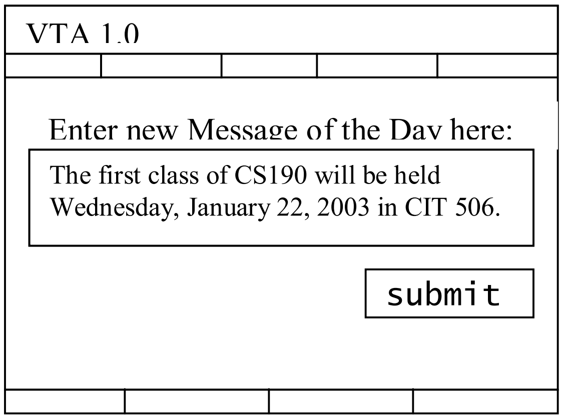
Database: The innards of this program will be in the form of a database that will hold information about users and assignments. The user data will include:

- Login and password
- Assignments and their grades
- Course status (student, TA, professor)

The assignment data will include:

-Assignment worth towards final grade
-Link to information about the assignment on the Web

GUI Representation (in all its hacky clipart glory!):



The first class of CS190 will be held Wednesday, January 22, 2003 in CIT 506.

GUI Description: The planned layout for the interface is to be a tabbed window, tabbed sub-window interface with primary tabs on the top and secondary tabs on the bottom of the screen. The main tabs would be Login, Web, Assign, Grade, and Data. The Login tab would have no sub-tabs; it would simply have one window for authentication. The Web tab would have areas for editing MOTD, Assignments, Course Info, and there would be a "New Tab" function for any new subpages the instructors wished to create. The Assign tab would have Create and Grade tabs. The Grade main tab would have a Sort Assignments, Grade, and Course Rubric sub-tabs. The Data tab would have interfaces to view Assignments, Users, and Grades.

Non-Functional Requirements:
Ideally, testing this program would involve tricking a professor at Brown (preferably one with a course with few students, or a course with many TAs) into using this program with

actual course data, but the timeframe of the assignment may not allow that. Therefore, testing will mainly involve creating a sample course (let's call it CS101) and getting students with and without previous TA experience to try and manipulate aspects of the program. This will also double as a usability study, as the purpose of the program is to help the TAs in their work, not hinder them. If something is found to be a hindrance, it will be changed immediately.

As for reliability, the internal database will meet minimum specification when crashing the program does not corrupt data that is not currently in the process of being edited. Having to lose a week's worth of data due to a power outage would be unacceptable.

All potential causes of error in the program's execution should resolve themselves with very little interruption for the user, especially if the error would simply result in an error message that would have no meaning to the user.

Updated Requirements:

Minimum Specification:
-Allow data entry and storage of grades in a user database.
-Allow course rubric definition by the user according to weights of assignments.

Extras:
-Add HTML Updater
-Add student web interface
-Add assignment sorter
-Add fully customizable course rubrics, with load/save file option

Risky Parts:

The design of this project suffers from lack of personal experience on my part with the world of teaching. Perhaps this can be overcome by teaming up with someone with a similar project idea who also has TA experience.

Also, parts of the project seem disjointed at first glance (assignment sorter? cool, I guess…). One major goal that I have now is to come up with a better defined theme for the tools available in this project.