

# CS526 GCRS - Graduate Course Request System - Spring 2018

Wael Ayadi, Nick Romano

Rutgers University

Piscataway, NJ, USA

Email: wael.ayadi@rutgers.edu, nick.p.romano@gmail.com

**Abstract**— We will create a Graduate Course Request System that will allow for easier and faster delivery and approval of graduate course requests for undergraduate students. It will incorporate our multi-purpose Visual Transcript visualization, which provides an interactive interface for extracting useful information from a student's course history.

## I. PROJECT DESCRIPTION

The Rutgers University Graduate CS Department allows for undergraduate students to register for graduate courses through a special request process. This process, as it is now, involves manually signing a form for every graduate course that the student wishes to register for. This signed form must then be signed by the instructor of the course that is being requested. Afterwards, the form is given to the graduate secretary of the Graduate CS Department. Depending on whether or not the requested course requires a prerequisite override (it often does, considering the student submitting this request is an undergraduate student), the form would then have to be forwarded to the graduate school for approval. This step of the process specifically may take up to a full week to complete! Once the course request is finally approved by the graduate school, the student receives his Special Permission Number (SPN) for the requested course.

Not only is this process tedious to describe, but it is tedious in practice as well. The student often has to chase down instructors for their approvals in person, as it is not realistic for them to constantly check their emails for course requests, as well as do the necessary background checks on the student to see if an approval for their request is appropriate. Further complications occur if an instructor is on sabbatical leave, if the graduate secretary is taking a day off, etc.. To make matters worse, students are bound by the add/drop week deadline for course registration at the beginning of every semester, thus making this registration process a timely matter that cannot always be done before the start of the semester. These issues are further compounded by the fact that the Rutgers Graduate CS Department has been growing in terms of student enrollment. The increased frequency of these requests end up delaying this process even further.

We propose the GCRS - Graduate Course Request System, which is a system that can be linked with Rutgers's Central Authentication System (CAS) to allow for Rutgers faculty to digitally manage and pass along student graduate course

requests. This system will use our Visual Transcript visualization, which will allow easy and fast data visualization of a student's course history and related data. Incorporating Visual Transcript into GCRS will give an interactive interface for faculty to quickly see the information they require when it comes to approving these requests.

### A. Stage1 - The Requirement Gathering Stage.

The goal for this project by the end of the semester is to have four views, one for each user type, where each view functionally shows course requests at its respective stage, along with the Visual Transcript display that the user can interact with.

There will be four types of users in the system: students, professors, secretaries, and graduate school administrators. Each user type also corresponds to a "stage" in the approval process. Stage 1 is when a student creates a course request, which initiates the approval process. Stage 2 is when a request requires approval from a professor, Stage 3 is when the request gets forwarded from a professor and requires approval from a secretary, and Stage 4 is when a request gets forwarded from a secretary and requires approval from a graduate school administrator. Once Stage 4 is complete, the student's request becomes approved, and an SPN is issued to that student. Each of these users will have one view that corresponds to them, where they will be in charge of their corresponding approval stage.

An example scenario would be as follows:

Student A is an undergraduate student who wishes to take CS526: Data Interaction and Visual Analytics, a graduate computer science course. Student A logs into GCRS, where he then creates a course request for CS526 [Stage 1]. The lecturer for CS526 is Professor James Abello. The professor then logs into GCRS, where Student A's course request is listed on his view [Stage 2]. The professor selects Student A's request, which then calls an instance of Visual Transcript for Student A's course history. Visual Transcript shows Student A's course history, as well as the corresponding grades he received for each course, sorted by completion time. After inspecting the Visual Transcript instance, the professor hits the Approve button, which forwards Student A's request to the graduate secretary. The secretary logs into GCRS, where Student A's course request is listed on her view [Stage 3]. Again, Student A's request is selected, which brings up a

Visual Transcript instance. The secretary notices that two other students have also requested CS526, but she can only issue one SPN. She selects CS526, which displays an instance of Visual Transcript that sorts the three students by expected graduation date. Student A will be graduating the soonest out of the three. The secretary selects Student A's requests and approves it. The system detects that Student A requires a prerequisite override for CS526. The approval automatically gets forwarded to a graduate school administrator. An admin logs into GCRS, where Student A's course request is listed on his view [Stage 4]. Again, the course request is selected, which spawns another Visual Transcript instance. The professor instructs the Visual Transcript instance to sort Student A's course history by course difficulty. The administrator inspects the transcript and is satisfied. Then he hits Approve, which automatically sends Student A an SPN, which will allow him to register for CS526.

A tentative timeline for project completion is as follows:

- **Feb 15:** A dummy database for use for this project will be generated and prepared for use.
- **Mar 1:** A prototype Visual Transcript view will be ready.
- **Mar 15:** A fully functional Visual Transcript instance will be functional.
- **Mar 29:** A system view will be completed for a generic user type. The view will contain a list of requests, the Visual Transcript, as well as any appropriate command buttons like "Approve" and "Deny."
- **Apr 12:** All four system views will be completed, as well as a temporary system login for each of the four user types.
- **Apr 19:** The powerpoint presentation will be completed.
- **Apr 26:** The project report will be completed.

#### B. Stage2 - The Design Stage.

This project will use various technologies for both the front-end and the back-end. For the back end, we plan to use an SQL database to store the data, as well as Java servlets for code execution. For the front-end, we plan on using Angular and Javascript for the layout and D3.js for the Visual Transcript instances. We may use jQuery for SQL query manipulation that will serve as input to the algorithm that we formulate that will retrieve the necessary data and spawn a Visual Transcript instance on each view.

As shown in Figure 1, the backend operations will be primarily driven by queries to the MySQL database. The query results will be fed into the D3.js API that will generate the Visual Transcript visuals. When the user selects certain nodes on the Visual Transcript interface, such as a specific request or a specific class, that item will be stored in a doubly linked list that will allow for progression and regression in the Visual Transcript interface. Using that linked list will allow our related items retrieval algorithm to populate the Visual Transcript instance with related information regarding the current selection.

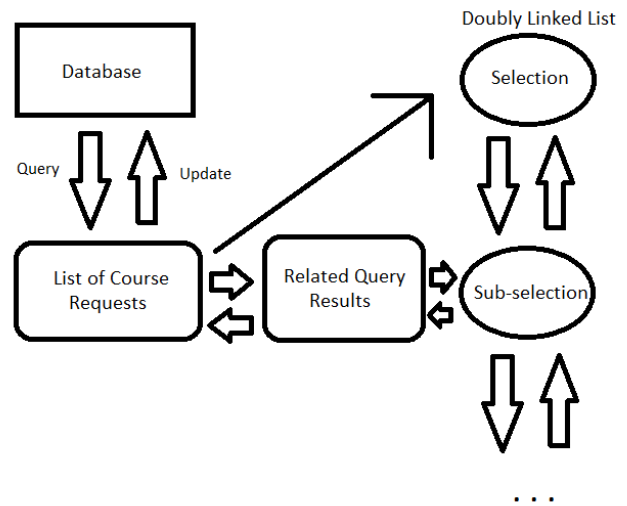


Fig. 1. Internal Data Structures

#### C. Stage3 - The Implementation Stage.

The intended programming languages, programming environments, and technologies that will be used for this project have been listed in Stage 2. The remaining deliverables for this section are a work in progress.

#### D. Stage4 - User Interface.

The user interface will consist of a visualization of all the course requests that correspond to the user's view. The list can be sorted by requests, students, or classes. Once a course request is selected, a Visual Transcript instance will spawn, creating an interactive visualization of the corresponding student's course history, which can also be manipulated further. The user interface will also contain "Approve" and "Deny" buttons, once a specific course request is selected.

Describe a User Interface (UI) to your application along with the related information that will be shown on each interface view (How users will query or navigate the data and view the query or navigation results). The emphasis should be placed on the process a user needs to follow in order to meet a particular information need in a user-friendly manner. The deliverables for this stage include the following items :

- The modes of user interaction with the data (text queries, mouse hovering, and/or mouse clicks ?).
- The error messages that will pop-up when users access and/or updates are denied
- The information messages or results that will pop-up in response to user interface events.
- The error messages in response to data range constraints violations.
- The interface mechanisms that activate different views in order to facilitate data accesses, according to users' needs.
- Each view created must be justified. Any triggers built upon those views should be explained and justified as

well. At least one project view should be created with a justification for its use.

Please insert your deliverables for Stage4 as follows:

- The initial statement to activate your application with the corresponding initial UI screenshot
- Two different sample navigation user paths through the data exemplifying the different modes of interaction and the corresponding screenshots.
- The error messages popping-up when users access and/or updates are denied (along with explanations and examples):
  - The error message:
  - The error message explanation (upon which violation it takes place): Please insert the error message explanation in here.
  - The error message example according to user(s) scenario(s): Please insert the error message example in here.
- The information messages or results that pop-up in response to user interface events.
  - The information message: Please insert the error message in here.
  - The information message explanation and the corresponding event trigger
  - The error message example in response to data range constraints and the corresponding user's scenario Please insert the error message example in here.
- The interface mechanisms that activate different views.
  - The interface mechanism: Please insert the interface mechanism here.

## II. PROJECT HIGHLIGHTS.