1. High level description

Describe the purpose of your software and its main features (i.e., what it can do)

Gradtool is an online admission system written for the University of Utah School of Computing Graduate Program. It uses a login-based web interface that collects application materials from applicants and verifies that all necessary materials have been received. Letters of recommendation can be uploaded to the system by outside professors at the request of the applicant. For applicant reviewers, tools are provided to sort applications by specific criteria and then view and make comments on individual applications. Administrators are able to manage accounts, manually enter applications, and enter news items that will be seen by applicants.

2. System Components

What programming language, environments, and software components (e.g., databases, openGL, etc.) does your system use? What platform(s) does it run on? If someone wants to use your software, are there any special system requirements (hardware and/or software) that the person's computer must have to run your software?

Gradtool is a server-side web application coded in PHP (version 4.12, http://www.php.net) using MySQL (version 3.23.49, http://www.mysql.com) for backend data storage. It was tested on the Apache web server, however should work on any web server software that supports PHP.

Users of the system need only have an internet connection and a recent web browser (it was tested using Internet Explorer 6.0 for Windows, Netscape 4.79 and 6.2 for Windows, Netscape Communicator 4.78 for Solaris, Mozilla 0.9.8 for Solaris, Opera 5.0b1 for Solaris, and Internet Explorer 5.0 for Solaris.)

3. Coolness factor

In your opinion, what is especially novel, interesting, and/or cool about your system?

Cody’s authentication system is something he is proud of and would be applicable to any web system where logins are required.

Tavish’s search query function is something that has me (Cody) in awe of the power of PHP. He spent a lot of time working on it, and it constitutes a major part of our system’s functionality.

Lastly, Tavish and Masao’s file uploading features are truly unique and we hope the password authentication will work out as a viable method for receiving letters of recommendation rather than requiring a signature.
4. System Capabilities

Provide a list of the capabilities that your software can perform.

- **Applicant Account Creation**
  Applicants enter their email address and are sent a randomly generated password for initial access to the system.

- **Lost Password Reset**
  Anyone that forgets their password can be sent a new randomly generated one by the system.

- **Password Change**
  Once logged on, users of the system can change their password.

- **Application Process**
  The application is collected in five parts. Each part has required fields marked in red against which completion is checked. As each part of the application is completed, the applicant’s main page is updated to show this.

- **Letter of Recommendation upload request**
  The last part of the application allows applicants to enter the email addresses of professors from whom they would like to receive letters of recommendation. The system then creates an account for each professor and emails them a random password. They can then log on to the system and see a list of applicants that have requested letters from them, and upload files for each. The applicant is able to see that files have been uploaded for him/her, but cannot see filenames or contents.

- **Application review—Finding applicants**
  Applicant reviewers each have individual accounts (created by the administrator.) When a reviewer logs on, they are presented with the basic search tool that allows for filtering based on any number of criteria. Each search criteria has two dropdown menus with the first option being the sort field (i.e. application year, application status, etc.) and the second the desired value to sort by. Two search criteria appear by default, but any number can be added.

- **Application review—Summary, comments, and file downloading**
  The applicant summary page lists all information collected about an applicant. Each reviewer can add their own comments to the application which can be seen by all other reviewers. This page also lists all letters of recommendation that have been uploaded by outside professors, and allows them to be downloaded. The MIME type of the file is stored when it gets uploaded, so depending on the browser, operating system, and installed software of the reviewer and the type of document, the file may load in the browser window for easy viewing or printing.

- **Administration—Account management**
  Administrators are able to see a list of all accounts sorted by the four different access levels (administrator, reviewer, applicant, and uploading professor.) Accounts can be individually created, edited, and deleted as necessary. Administrators can also manually change the password of any account.

- **Administration—News items**
When an applicant logs in to the system, a list of administrator created news items appears at the top of the page. Tools are provided for the administrator to quickly add, edit, or delete any of these news items.

5. Individual Contributions
What portions of your system did each team member work on?

Tavish – Main programmer
Cody – Assistant programmer, site design, team web site
Masao – Assistant programmer, documentation

Cody built the foundations of Gradtool such as the authentication system and basic account management. He also coded the news functions. Together, Cody and Tavish completed the "look and feel" of Gradtool in addition to the applicant reviewer tools. Tavish coded the search tools and revamped much of the code for portability, and also completed the file uploading system (which built on Masao’s initial version.) Cody and Masao incorporated Mod_SSL and OpenSSL but we did not use these in the end because we decided the server side security features were good enough.

6. Accomplishments
Are you happy with your final project? Did you get everything accomplished that you wanted to?

Although there are a few things that we did not have time to complete (cron tools primarily), we are very proud of our project. Only one of us came into this with a working knowledge of PHP, MySQL, and HTML so we all feel a great sense of accomplishment in having implemented as much as we did. We all know for certain that this knowledge will be helpful in our future careers.

7. Lessons Learned: Working as a Team
Discuss any lessons that you’ve learned from your CS4500 project about working as a team.

Cody: As said below, I learned that planning and delegation of work are the single most important parts of working in a team on a large project. There were a few points in our project when none of us really knew what our jobs were and it led to stagnation.

Tavish: Working in a group I learned you need to quickly understand the people in your group. Learn their strengths and weaknesses, and most importantly know your strengths and weaknesses: know your limitations so as to not take on "more than you can chew."

I learned that it's not the project, initial idea or planning that are the most difficult parts, but the implementation. This is where I found we got most bogged down -- by thinking through the entire problem in our heads, foolishly estimating it to be less hard than it turned out to be and then being crunched during the coding phase.
Overall I enjoyed this experience. I found that I could quickly evolve my skills to fill the gaps that needed to be filled and the other two guys were easy to work with. I feel that we lucked out with our group as we were able to mesh so well.

**Masao:** We really liked to line up three computers in the lab, discussing and coding together. We figured out working environment was important and CADE lab is the best place to work. It was difficult working as a team especially during the Olympic break. Our group meetings were very efficient. We tried to keep them short and list out the problems we were having and planning for next accomplishments.

8. **Lessons Learned: Building a Large Software System**
*Discuss any lessons that you've learned from your CS4500 project about building a large software system.*

Planning, Planning, Planning! We didn’t do enough of it and what we did wasn’t specific enough, and it came back to bite us when we had to re-code many elements of the system. In our defense, the main reason for this was we didn’t know what to expect due to our limited knowledge of PHP and MySQL, but more still could have been done.

One thing that we did do from the start that was very helpful, however, was using CVS for our code repository (Tavish ran the CVS server on his home computer.) It was a great help to be able to stay up to date with what others were doing and was very suited to our sometimes irregular coding schedules.

9. **Lessons Learned: Time Machine**
*If a time machine could transport you back to January, what would you do differently in designing, implementing, or managing your project?*

See #8.