New York University
Computer Science and Engineering
CS 6083, Fall 2016

November 22, 2016

Project #2 (due December 15)

In the second project, you have to create a web-based user interface for the database designed in the first project. In particular, users should be able to register, create a profile, log in, create, read, rate, and comment on recipes, join cooking groups, and RSVP to group cooking meets, etc., as described.

Note that you have more freedom in this second project to design your own system. You still have to follow the basic guidelines, but you can choose the actual look and feel of the site, and offer other features that you find useful. In general, design an overall nice and functional system. If you are doing the project as a group of two, note that both students have to attend the demo and know ALL details of the design. So work together with your partner, not separately. Also, slightly more will be expected if you are working in a team. Start by revising your design from the first project as needed. In general, part of the credit for this project will be given for revising and improving the design you did in the first project.

After a user logs in, she should probably come to a personalized start page where she might see a feed of items that are relevant to her, e.g., a list of her group memberships, a list of recipes she recently looked at, or maybe new events, or RSVPs for events by other group members. You can decide what to put there. Users should be able to browse the site, say, by clicking on tags or following links to related recipes. There should also be some way to search using keywords that are matched against recipe titles, descriptions, and tags. Note that the database should log certain user activities on the site; in particular, you should log information whenever a user looks at a recipe, searches using keywords, or clicks on a tag. This allows the system to display recipes the user has recently looked at, or to display recently clicked tags or recent searches, and in general to learn what kind of recipes the user is interested in.

Users should be able to perform all operations via a standard web browser. This should be implemented by writing a program that is called by a web server, connects to your database, then calls appropriate stored procedures that you have defined in the database (or maybe send queries), and finally returns the results as a web page. You can implement the interface in several different ways. You may use frameworks such as PHP, Java, Ruby on Rails, or VB to connect to your backend database. Contact the GAs for technical questions. The main restriction is that the backend should be a relational database using the schema you designed in the first part, with suitable improvements as needed.

Your interface must take appropriate measures to guard against SQL injection and cross-site scripting attacks. To prevent SQL injection you can use stored procedures and prepared statements (if your programming language supports them). If your language does not support prepared statements, your code should check and sanitize inputs from users before concatenating them into query strings. To guard against cross-site scripting, outputs to be returned to user's browsers should be checked/sanitized to avoid scripts. Some languages provide functions, such as PHP's htmlspecialchars, to help with this.

Every group is expected to demo their project to one of the GAs at the end of the semester. If you use your own installation, make sure you can access this during the demo. One popular choice is to use a local web server, database, and browser on your laptop, which you then bring to the demo. (In this case, your project can just run locally on your laptop). Also, one thing to consider is how to keep state for a user session and how to assign URLs to content – it might be desirable if users could bookmark a page within your site, say for one message or discussion, or one search that was performed. Grading will be done on the entire project based on what features are supported, how attractive and convenient the system is for users, your project description and documentation (important!), and the appropriateness of your design in terms of overall architecture and use of the database system. Make sure to input some interesting data so you can give a good demo.

Describe and document your design. Log some sessions with your system. Bring your description and logs to the demo (softcopy is OK). You should also be able to show and explain your source code during the demo. The documentation should consist of 15–20 pages of carefully written text describing and justifying your design and the decisions you made during the implementation, and describing how a user should use your system. Note that your documentation and other materials should cover both Projects 1 and 2, so you should modify and extend your materials from the first project appropriately. There will be opportunity to get extra credit by implementing extra features, but extra credit is limited to about 10-15%. There may also be extra credit of up to 5% for doing an early demo, before the deadline.