New York University November 27, 2019
Computer Science and Engineering
CS 6083, Fall 2019

Project #2 (due around 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, and reply to messages, leave and apply to join blocks, and choose friends and neighbors, 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. There will be some extra points available for a nice and smooth design. 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. 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.

A note about the interface you are expected to build for this project. When users log in, they should probably come to a starting page where they can see a feed of recent messages and site activities that are accessible to them, Users should also be able to browse the site, and there should be some way to search using keywords and locations/maps, for example retrieving all messages that refer to a particular location and/or contain certain keywords. Also, there should be a ``map view'' where people can see a map of the block or neighborhood together with the home addresses of other users in the block and the locations of recent messages (where messages can be tagged with a particular location as described in Project #1). You should use Google Maps or similar APIs to achieve this.

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 TAs for technical questions. The main restriction is that the backend should be a relational database using the schema designed in the first part, with 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 (documentation) and the logs in hardcopy to the demo. You should also be able to show 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 for extra credit by implementing cool extra features, but extra credit is limited to about 5-10% (and the TAs decide what is cool). There may also be extra credit of up to 3-5% for doing an early demo, before the deadline.