1 Overview

In this course project you will design an online question and answer (Q&A) site similar to StackOverflow. Your site should provide an interface that allows users to browse questions and answers, to ask new questions and comment on questions, and to vote answers up or down. Questions should be tagged by meaningful keywords, and it should be possible to browse questions by user, by tag, or by their status (e.g. "unanswered"). Registered users should have a profile page and should acquire badges based on the questions or answers they have provided.

You should use StackOverflow (and perhaps similar sites) as inspiration in the development of your site. The StackOverflow FAQ [http://stackoverflow.com/faq] has some details on how the site functions. You may also want to read more about it on Wikipedia [http://en.wikipedia.org/wiki/Stack_Overflow].

The Q&A site you develop needs to (i) satisfy minimum functionality requirements, (ii) provide good performance for common user operations, and (iii) implement selected advanced features in addition to the basic functionalities. The grading of the project combines all three requirements using the scheme shown below:

65% Correct implementation of minimum functionality

10% Performance of required features

15% Advanced feature(s)

10% Intermediate milestone grades

You may consult with other students about your site development, however your implementation must be your own. Your code should be your own – you should not use any external libraries or open-source code not part of standard django or python.

2 Minimum functionality

Your application must support the following basic features.
2.1 Registration
Anonymous users may browse the site, but you may assume that all questions and answers are posted by registered users and that only registered users can vote on questions/answers. A registration page should allow new users to create a profile or allow existing users to login with a username and password. Note that users may need to edit their profile more than once after creating it the first time.

2.2 User Profile
Each user will have a profile page describing basic information (name, username, last login to the site, hometown, etc.) and the badges earned by the user. This page should also list the questions asked by a user, with the ability to sort questions a few ways (see below).

2.3 Questions
When a registered user asks a question, he or she becomes the owner of the question and may edit the question later. The owner also chooses one or more tags for the question. Any registered users can add comments to questions or post answers to questions.

2.4 Voting
Both questions and answers can be voted up or down by registered users. The author of a question should not be able to vote on their own question (and likewise for answers). A question becomes "answered" if there is at least one answer that has been voted up.

2.5 Browsing Questions
Users should be able to view the questions on the site by browsing the list of questions sorted by: newest (questions most recently asked), votes (questions with the most votes), or active (questions with recent activity). It should also be possible to filter questions by tag or by status "answered" or "unanswered" (while still sorting by the above properties).

2.6 User Badges
Develop a simple system of badges awarded to users for their contributions to the site. (It need not be as complex as the stack Overflow badge system.) Badges should be awarded automatically whenever they are earned, without a process for requesting badges. See the page above and implement the badges for "Critic", "Supporter", "Student", and "Tumbleweed" (but ignore the low views condition). There should be a page displaying summary statistics about all badge types and the number of awarded badges of each type.

2.7 Search
Your application should allow users to search the collection of questions by keyword, and should return both exact matches and substring matches (using LIKE).
3 Performance

Your application is expected to run with reasonable performance. While there are no specific
requirements in terms of response time, most operations on the website, including user regis-
tration, user login, searches, and updates, should be completed with no obvious delay.

If you observe significant delay in operations in your application, it is important that you
review your design and implementation to identify the causes of the slow performance. As a
general guideline, you may want to consider the following things:

• Django automatically creates indexes for you, which may be sufficient. You may want
to consider creating additional indexes to improve the performance. The class material
relevant to this task is physical design and tuning. The relevant SQL command is CREATE
INDEX index_name ON table_name (column_name(s))

• Did you implement the functionalities of your application in the right places of your pro-
gram? For example, a join between two tables should be performed in the database back-
end. Performing the join in your Python code is a mistake – even though you may still
provide the correct answers, the performance of your application will suffer. Does the
loading of a single page in your application require many queries? You may want to in-
vestigate caching in django to improve this.

• Can you refine your models to improve the performance?

4 Advanced Features

Your application is expected to have at least one advanced feature beyond the basic function-
alities described above. The following list presents several ideas regarding the project enhance-
ment. You are welcome – in fact, encouraged – to propose and develop new features not listed
above.

1. Add a feature to record the number of times a question is viewed (by any user). In Stack-
Overflow, this is referred to as "views". Integrate this into your design by: (i) permitting
questions to be sorted by number of views, and (ii) using the view count to award badges.

2. Advanced search – Extend the search functionality to include search over not just ques-
tion text but also answer text. The result of a search will then need to return links to
questions or answers. Attempt to rank the output appropriately by favoring questions
containing the search text over answers.

3. Activity Log – Add to the user profile page the ability to view an activity log describing all
the actions taken by the user, ordered by time. This should include users’ changes to their
profile, questions asked, commented-on, or answered, etc.

4. Security – Implement one or more of the following security features: secure use of cookies,
resistance to SQL injection attacks, secure storage of passwords, secure urls.
5. Scale – Generate synthetic data that can be loaded into your site to test its performance on a realistic base of data. You should issue a bulk update operation to populate the database tables with users and questions/answers. You should end up with about 1,000 users and 10,000 questions, and then make sure that response time for operations on your site is still good. The synthetic values you insert do not have to be meaningful (e.g. use ’user1000’ for a username, etc.).

6. Even greater scale – Explore the implications of scaling your database by a factor of 10 over what is described above.

5 Deliverables

Your site should be running on the edlab machines for evaluation by the Instructor and TA, although you may do development on your own computer. You are required to turn-in the following (please see the course website for details and due dates).

1. Intermediate milestones - In conjunction with weekly homework assignments, you will be asked to implement portions of the functionality of your site throughout the semester. These intermediate milestones will be graded and will contribute a small amount to your final project grade.

2. Final Assessment & Report - The final submission of the project consists of a brief (about one-page) written report on your implementation, along with the Python code for all of your models, both submitted to SPARK by 11:59pm Dec 18th. The report should include the following:

   Link  The link to your functioning site.

   Minimum functionality  State whether your site supports all of the minimum functionality specified above. If there are any noteworthy aspects of your implementation, please mention them. For example, you may wish to describe your technique for representing votes, or how badges are awarded and stored. If you diverged from the minimum requirements by extending your site, please mention those extensions. If any features are missing, explain the current state of your implementation and any reasons why you were not able to meet the requirements.

   Advanced features  Explain which of the advanced feature or features you implemented along with any noteworthy aspects of your implementation.

   Unsolicited features, known bugs, comments  If there are other special features of your site, known bugs, or other comments you think are relevant to grading of your project, please mention them.