# Quiz Website

Your objective is to create a website where people can show off their intellectual prowess by taking quizzes on a wide range of subjects. However, not only can they take quizzes, they can compare their quiz scores to friends, challenge friends to beat their score, or write quizzes that other people can take. This project should be turned in by **9pm** the evening of Friday March 11th.

In addition to turning in the actual project, you must meet the following deadlines (each item will be explained in more detail later in the handout):

- Team design meeting with a member of the teaching staff by Friday February 25th.
- Team demonstration meeting with a member of teaching staff by Friday March 4th.
- Actual project submitted, Friday March 11th by 9pm.
- Grading meeting with TA, the afternoon of Saturday March 12th.
- E-mail team evals by midnight the evening of Monday, March 14th.

In this handout, I will distinguish between required features, recommended features, and extensions. I will also mark features for three-person, four-person, and five-person teams. If a feature is not marked, assume that it is required for all teams.

We expect all teams to work on extensions. Teams completing only the required features should expect to receive a C grade. Three and Four-Person teams completing required and recommended features should expect a B grade. I don't have any recommended features listed for the five-person teams, so they'll need to pick and choose from the extensions listed or come up with their own in order to get an A or B grade. Small-sized teams may do features marked as required or recommended by larger-sized teams as extensions. You may also come up with your own extensions.

## Quizzes

Let's start off by taking a closer look at the quizzes themselves. Each quiz is composed of a number of different questions.

### Question Types

All teams must support the following question types:

**Question-Response**—This is a standard text question with an appropriate text response. For example: Who was President during the Bay of Pigs fiasco?

**Fill in the Blank**—This is similar to standard Question-Response, except a blank can go anywhere within a question. For example: "One of President Lincoln's most famous speeches was the _____ Address."

**Multiple Choice**—Allow the user to select from one of a number of possible provided answers. Please present multiple-choice questions using radio buttons—this should

not be treated as a Question-Response question where the user enters an "A", "B", or "C" into a blank textfield.

**Picture-Response Questions**—In a picture response question, the system will display an image, and the user will provide a text response to the image. Here are some examples of picture-response questions. The system displays an image of a bird, the user responds with the name of the bird species; the system displays an image of a US President, the user responds with the name of the president; the system displays a chemical structure of a molecule, the user responds with the name of the molecule.

To keep things simple, you may use absolute URLs to external images as the source of your images, instead of allowing the user to upload images to your server when creating a picture-response questions. For example a quiz on buildings at Stanford would serve up:

http://events.stanford.edu/events/252/25201/Memchu_small.jpg

rather than serving up a copy of the image stored on your test server.

Your system should be designed to allow easy introduction of new question types. Here are some straight-forward question types you can consider adding as extensions. Even if you don't use these extensions, adding them should not require reworking of any of your existing code (although it will, of course, require you to write additional code).

**Multi-Answer Questions**—This is similar to the standard question-response, except there needs to be more than one text field for responses. Allow the quiz creator to determine if the responses need to be in a particular order or not. For example, list the 50 states in the US (order not relevant) or list the 10 most populous countries in the world in order from largest to smallest.

**Multiple Choice with Multiple Answers**—This is similar to a standard multiple choice question, except the user can select more than one response. For example: please mark each statement below which is true (1) Stanford was established in 1891, (2) Stanford has the best computer science department in the world, (3) Stanford will be going to a bowl game this year.

**Matching**—An elegant professionally done matching system would probably require client-side programming possibly in conjunction with the HTML5 canvas element. You could create a rather awkward version by allowing users to choose matches via a set of pulldown menus.

More exotic question extensions are possible including:

**Auto-Generated Questions**—Here the computer generates questions on its own. This could be used, for example, to create a mathematics exam for 3rd graders, where the questions were not pre-written, but instead generated on the fly based on the student's performance. These types of questions will introduce issues with how a user would create the exam, which you'll have to address.

**Graded Questions**—The computer may not be able to determine a correct answer to all questions. For this type of question, the system will store the user's responses and provide an interface for a human to come in later and provide a grade. This could be used, for example, to allow short answer questions or even essay questions. This extension will require adding a variety of mechanisms including notification to users when a quiz is graded and providing an interface for graders.

**Timed Questions**—As we'll see in a moment, quizzes are timed, however questions are not. You can add a timer to individual questions. This extension is probably best done with client-side programming, although I can think of at least one inelegant way to solve it using just HTML tags.

### Answers

Question/Response, Fill in the Blank, and Picture-Response questions may have one or more legal answers. For example, a user might answer a question of "Who was the US's first President?" with either "Washington" or "George Washington". Multiple-Answer questions with unordered answers may have more legal answers than answer slots – e.g., name five movies which won the "Best Picture" Academy Award. Multiple-Answer questions with ordered answers will have only as many answers as answer slots – e.g., name the five largest cities in the United States.

As an extension you may try something fancier with the Multiple-Answer questions. For example, given the question "Name five dorm complexes" you may want to allow the user to either answer "FloMo" or "Florence Moore" but not both.

### Quiz Properties and Quiz Options

Each quiz should include an overall description of the purpose of the quiz.

Quizzes have the following required options:

**Random Questions**—Allow the creator to set the quiz to either randomize the order of the questions or to always present them in the same order.

**One Page vs. Multiple Pages**—Allow the quiz writer to determine if all the questions should appear on a single webpage, with a single submit button, or if the quiz should display a single question allow the user to submit the answer, then display another question. The one-question-per-page approach will work well for a flash-card style quiz, where the website flashes up an image or photograph and asks for a response, followed by a new page with a new image. Single-page quizzes will be good for most other quiz types.

**Immediate Correction**—For multiple page quizzes, this setting determines whether the user will receive immediate feedback on an answer, or if the quiz will only be graded once all the questions have been seen and responded to. The immediate correction option will work in conjunction with picture-response questions to create a flash-card type quiz. The computer will bring up a flash card (i.e., a picture) the user will

respond with the answer and the computer will immediately provide feedback on whether the answer was correct or not.

**Practice Mode**—The quiz author can choose whether or not the quiz can be taken in practice mode. This possible feature is described in the extension section.

### Scoring

Quizzes will be scored on the basis of how many questions a user got right. They should also keep track of how long a user took to complete the quiz. For multi-answer questions, treat each blank as a separate answer for grading purposes. For example if the multi-answer question was "List the five largest cities in the US" it would count as 5 separate answers for grading purposes, and a user could receive anywhere from 1-5 points for the question depending on their answers.

All websites should display the user's score and completion time when they complete their quiz. As a recommended feature for three-person teams and a required feature for teams of four or more, each quiz should keep a history of how well users did when they took it. Three-person teams not implementing this feature should ignore all further references to "history" or past quiz performance in this document.

Quiz pages should display a list of top scorers. Rank top scores first by the number of questions which were correct, and in the case of ties by the amount of time taken. In other words, if one user finished all the questions correctly in ten minutes and a second finished in 5 minutes, but got only 90%, the first user will be ranked first. However, if two users got 80% right, but one took two minutes and the other took fifteen minutes for the exam, the user who took only two minutes will be ranked higher.

## Users

The system will keep track of users. Users should have login names and passwords.

Use a system similar to the HW4 Cracker program to keep the passwords encrypted. When the user creates an account for the first time, hash their password and store that in the table, instead of storing their password in clear text. When they log back in hash the password they enter and compare it to the hash stored when the account was created. If they match let them in the account.

For added security you can add a value called a *salt* to the password before hashing it. Check a security textbook or online resource for more details.

Feel free to cannibalize one of your team member's assignment 5 login system and assignment 4 cracker program as the basis for your user system. You may leave passwords restricted to the HW4 Cracker character set, or support more comprehensive passwords. Remember you are hashing passwords, not cracking them, so processing requirements will be minimal compared to what was needed in HW4.

### Friends

The system allows users to become friends with each other. Provide at minimum two ways for users to find each other and become friends:

- Any time a user is listed, for example in a listing of top quiz scorers or as the creator of a quiz, this should include a link to the user's page. This page should include a way to add the user as a friend.

- Provide a method for users to lookup users, so if a real-life friend sends their quiz-site user name outside of the quiz-system (say by sending a real, non-quiz-site e-mail), someone can use that name to look up the person on the system and add them to the friends list.

As with the Facebook system adding a friend is a two step process. When a user adds a friend, that will generate a "Friend Request" message (see next section). The friendship won't become official until the "Friend Request" is confirmed.

Provide a way for people to remove friends.

### Mail Messages

The quiz website has its own internal mail system. At minimum you should support the following message types:

**Friend Request**—Someone has asked to be friends with the user. Provide a method for the user to say yes or no, and update the system as appropriate on a yes response.

**Challenge**—A friend has challenged the user to take a quiz. This type of message should automatically provide a link to the quiz, and should include the challenging user's best score on the quiz.

**Note**—This message type is just a simple text note which a user can use to send whatever text they want to another user.

### History

The system should keep track of a user's past quiz performance and display it on a history summary webpage. It should also display an abbreviated version of their history on the homepage.

### Achievements

Achievements are a recommended four-person feature and a required five-person feature. When a user completes a certain set of requirements, award them with an achievement. The achievements should be displayed on their homepage, and you may display them in other locations as well as desired. At minimum provide the following achievements:

**Amateur Author**—The user created a quiz.

**Prolific Author**—The user created five quizzes.

**Prodigious Author**—The user created ten quizzes.

**Quiz Machine**—The user took ten quizzes.

**I am the Greatest**—The user had the highest score on a quiz. Note, once earned this achievement does not go away if someone else later bests the high score.

**Practice Makes Perfect**—The user took a quiz in practice mode.

Feel free to change the achievement names to something more exciting and to come up with colorful icons for each achievement. One nice approach would be to display the icon and then have a tooltip, describing how the achievement was earned, appear when the mouse moves on top of the achievement icon. Add additional achievements as desired.

## Administration

Provide a web interface to allow administration of your website. This feature is recommended for four-person teams and required for five-person teams.

At minimum administrators should be able to:

- Create announcements which will show up on the homepage.
- Remove user accounts.
- Remove quizzes.
- Clear all history information for a particular quiz.
- Promote user accounts to administration accounts.
- See site statistics. These should include number of users and number of quizzes taken.

## Pages

Here are some pages that you should include in your website. This is by no means a complete list of every webpage your website will include. But I wanted to go into more depth on the minimum requirements for some of the pages. You may skip items listed if they are dependent on a feature your team is not supporting—for example you won't list administration announcements, if you are not supporting the administration feature.

### Home Page

Once the user logs in, they should see the following items:

- An Announcements section highlighting any items put up by the website administrators.
- A list of popular quizzes.
- A list of recently created quizzes.
- A list of their own recent quiz taking activities.
- A list of their recent quiz creating activities, if any. You may eliminate this list if the user has never created a quiz.
- An indication of all achievements they've earned. This can be an abbreviated list with a more complete list elsewhere, if you believe that will provide you with a better looking or more functional website.
- An indication if they've received any messages, with some information on the most recently received messages (such as whether they are challenges or friend's requests).
- A list of friend's recent activities including quizzes taken or created and achievements earned. This summary should include links to the friend's user page and the quiz pages.

### Quiz Summary Page

Each quiz should have a summary page which includes

- The text description of the quiz.
- The creator of the quiz (hot linked to the creator's user page).
- A list of the user's past performance on this specific quiz. Consider allowing the user to order this by date, by percent correct, and by amount of time the quiz took.
- A list of the highest performers of all time.
- A list of top performers in the last day.[1]
- A list showing the performance of recent test takers (both good and bad).
- Summary statistics on how well all users have performed on the quiz.
- A way to initiate taking the quiz.

[1] You may change the time interval here for testing purposes. For example, it could be the top performers in the past fifteen minutes. You can leave this shorter time period in for when you present and turn in the project.

- A way to start the quiz in practice mode, if available.
- A way to start editing the quiz, if the user is the quiz owner.

### Quiz Results Page

This page appears after a user has taken a quiz. It shows how well the user performed on the quiz including their score and time. You may also want to list all the user's answers along with the correct answers here.

This page could also include tables showing comparisons to the user's past performance on the quiz and comparisons with top performers or with the user's friends.

## Overall Website Look

We expect three-person teams to have very rough page designs. Web pages should be functional, but do not have to look good—for example text fields do not have to line up and it's okay for the pages to have a generally ratty, disorganized look and feel as long as they are functional. Four-person teams should have reasonably neat and organized looking webpages. Five-person teams should have semi-professional looking webpages.

If a four or five person team does include anyone knowledgeable on HTML and CSS on the team, they may do additional extensions in place of creating a nice looking website.

## Error Checking

All teams are required to have minimal error checking.

To support minimal error checking make sure that if the user starts at the login page, they can never follow a link on the website or submit a form which results in receiving a server error message. They should not see a 404 webpage or any other error webpage automatically generated by the webserver code itself.

You are not responsible for what happens if the user enters in a malformed URL or manually modifies a URL including GET form information to submit something nonsensical.

You may assume that the user never enters nonsensical data into forms. All required form inputs will be filled in and will be of the proper type and format (e.g., you will never receive a word when you expect a number).

## User Sessions and Threading

You should support multiple users simultaneously visiting your website. Therefore you need to handle user-specific data using sessions. However, you may assume that the users never simultaneously require access to the same data structures; therefore you do not need to provide locking mechanisms at the thread level.

## Extensions

Here are some possible extensions. Feel free to come up with extensions not on this list.

### Tags and Categories

Allow the quiz creator to categorize the quiz. A quiz, for example, might be marked as history, geography, or science.

You may want to distinguish between categories and tags, where each quiz can only be placed in a single category, but may have multiple tags. In addition subject categories may be limited to a pre-defined list created by the website administration, whereas a user can attach any tag desired to a quiz.

Quizzes can be listed by category on a webpage listing all the quizzes (or perhaps just the most popular quizzes if you've got a lot of quizzes on the websites). Quizzes can be searched for by tag.

### Quiz Editing

You do not need to provide a method for users to modify quizzes after they have been created. You can add a post-creation editing ability as an extension.

### Rating and Review System

Allow users to rate quizzes they have taken. Consider also allowing users to write reviews of quizzes. Use the ratings to create a list of most popular quizzes.

### Reporting System

Allow users to mark quizzes as inappropriate. Create an administration interface to review quizzes marked to determine if they should stay on the website. Consider adding the ability to ban users for a limited amount of time for inappropriate behavior.

### Improved Error Handling

Gracefully handle situations in which the user enters incorrect or nonsensical data into your forms.

### Non-Registered Access

In a standard implementation of this website, a user visiting the website would be presented with a login page. They would not be able to see any of the website before either logging in or creating an account.

As an alternative, you could allow non-registered users to see any administration announcements, lists of popular quizzes, and even explore individual quiz pages, but not take any quizzes until they have created an account and logged in.

### Cookies

Use cookies which allow users to get automatically logged back in to the website when they visit.

### Privacy Settings

Create privacy settings which determine whether or not a user's quiz performance shows up for everyone or just friends. Possible privacy settings could include listing a user's name, but not allowing access to their "user page" or not listing their name to people who aren't friends, but instead listing their quiz performance as "anonymous".

### XML Loading

We will release an XML format for quizzes along with some sample quizzes by next Tuesday. Consider creating a mechanism to automatically load quizzes written with this format. While teams should, of course, not share code, we will allow them to share quizzes with one another using XML.

### Naming

Come up with a catchy name for this assignment. The best I've come up with is Quiz-O-Rama, which frankly isn't great. If your name is good, and if people like this assignment, your name could one day be as famous as Bunny World.

### Practice Mode

Practice mode is a special mode in which the user practices questions in the quiz, but their score is not recorded. The computer provides additional assistance in practice mode by repeating questions that the user has difficulty with and removing questions that the user knows well.

In practice mode, the system will present questions from the quiz. The order of questions will either be random or fixed depending on that quiz's original setup options. Once the quiz is completed (i.e., once all the questions have been presented), the quiz will repeat. However, if the user gets the answer to a particular question correct three times during the session, that question will be removed from the session's question rotation.

Terminate practice mode either when the user explicitly decides to leave a practice session or if the user gets all questions correct three times.

If you really want to get fancy, consider allowing the user greater control over practice mode. For example, the user might have a list of past runs of the questions and how they did on each question. They might be able to manually select a subset of the questions to use. They might have greater control over when a question should be removed from the practice session—for example telling the computer to remove a question the first time they get it right, instead of waiting for the third time they answer correctly.

## Restrictions

You may not use a server-side framework—your server-side code should be written using vanilla Servlets and JSPs. If you decide to do client-side processing, you may use JQuery as a client-side framework. You'll need to get written permission from us in order to use any other frameworks.

All user and quiz data should be stored using a MySQL database. Text files are limited to those described explicitly above, plus the web.xml file for setting up configuration information. For extensions in which you want to use a text file, please check with us.

## Summary of Team-Size Differences

Here are the differences between the team sizes. Remember assuming you properly implement the features listed, a team doing all required features can expect a C, a team doing recommended and required features will receive a B. All teams should do extensions if they want to receive an A. There is no upper-limit on points on the project, and last quarter we had one team earn over 150% credit on the project.

### 3-Person

- webpage design may look rough as long as it's functional

Recommended

- history

### 4-Person

- webpage design should look neat and organized

Required

- history

Recommended

- achievements
- administration

### 5-Person

- webpages should look semi-professional

Required

- history
- achievements
- administration

Recommended

- Must do extensions.

## Meetings and Deliverables

### Design Meeting

All teams should meet with Dr. Young or one of the TAs to discuss their design and plans sometime between Monday February 21 and Friday the 25th. Before this meeting takes place, I expect you all to have given careful consideration to how you plan to approach the problem and how you might assign tasks to individual team members. We will setup a schedule for team signups and announce it this weekend.

Teams which consist of 50% or more non-local SCPD students are exempt from this requirement. I am open to suggestions for how to incorporate non-local SCPD students in these meetings.

### Demonstration Meeting

In the past we've had some teams fail to deliver successful projects. One of the common themes between these teams was the failure to integrate early. In order to prevent these disasters, you are required to meet with a member of the teaching staff during office hours during the week of February 28th and show running code. The demo should show code which includes contributions from at least two team members (to demonstrate some code integration has taken place).

### Code Submission

Zip up all your code and submit it via the standard submission script. In addition to submitting your code, write a file which lists all your team members and describes all the features which you completed. If you have skipped any features which were required or recommended for your team size, you must provide a list of what you missed. This file can be a Word Document or an HTML file. This file should be named README.doc or README.html and should included in your zip file.

Only one team member should submit the actual code, the other team members should submit a simple text file listing their team members, and notifying the TAs which team member account was used to submit the code.

Your project is due Friday March 11th by **9pm**. We will use the same draconian penalty that we used on HW5 on the final project. Projects will be penalized on a per-minute basis. Projects turned in just 3 hours late will receive a zero.

### Grading Meeting

All teams will meet with TAs for grading on the afternoon of Saturday the March 12th. We'll provide scheduling details as the date gets closer.

### Team Evaluation E-Mail

Send a short e-mail to patrick.young@stanford.edu describing how the project went and evaluating your performance and that of your fellow team members. The mail should list each team member with a very short description of what they did and give an estimate of what percentage of the project you think they did – for example: John did 20%, Jane did 40%, and Jesse did 40%. For the majority of teams, everyone will have contributed more or less equally, and the analysis will not affect anything. However, these evaluations may modify individual team member's grades if there is a strong and clear disparity in the amount of work done.