ACM Application

Navin Bhaskar
Rochester Institute of Technology
Rochester, New York
E-Mail: nxb8951@cs.rit.edu

March 5, 2004

Document Version : Final

Version	Date	Description
Final	3/5/2004	Final Version.
Draft	3/1/2004	1) Use cases simplified. 2) Task of authentication and authorization delegated to the application server. 3) Clarification mechanism and concurrency issues elaborated.
Draft	1/14/2004	1) Teams cannot choose judge for clarifications. 2) Screen shots modified for point 1. 3) Inputs from ACM contest of Oct 2003 incorporated. 4) Sections for Logging, File Submission data flow, Scoreboard refreshing included and/or elaborated
Draft	10/2/2003	Draft proposal

Contents

List of Figures

Abstract

Association for Computer Machinery (ACM) conducts an international collegiate programming contest held on an annual basis. Teams compete to solve multiple questions within the allotted six hour duration of the contest. A panel of judges grades the solutions online. The team that answers the maximum questions correctly wins the contest.

This application will assist in the process of selection of winners of the programming contest. It will make the process of submission of solutions, and grading of the answers convenient. The ACM application is a Java based and web-enabled application and therefore platform independent.

1 Required Features

1. Maintain a scoreboard.
2. All the activities (commands, arguments and results) are to be logged.
3. In case of a server failure, there should be a way to recover from the same and display the scorecard just before the event.
4. There should be a way to freeze the scoreboard. During this period of inactivity, the scoreboard should display the running time at the server.
5. Clarifications will be distributed among the judges on the basis of question numbers. Clarifications provided by the judges to a team will be automatically posted to all teams.
6. Ability to analyze the logs in case of a server-crash and deduce when it was submitted and graded.
7. Reasonable performance.

2 Additional Features

1. Reusability of this application in similar contests is desirable.
2. Web page with individual contact information made available exclusively to the judges.
3. Web pages containing documentation to be made available to the teams.
4. Option to view past contests for use by judges and administrators.

3 Actors and Use Cases

"A Use case is a description of a set of sequences of actions, including variants that a system performs to yield an observable result of value to an actor." [1]

3.1 Actors

3.1.1 Team

Each team is comprised of one or more participants and identified by an unique (user) name. Individual team's can request clarifications from judges. The judges will provide responses to these clarifications, which are later posted to all the teams. More than one team will participate in the programming contest.

3.1.2 Judge

A judge is responsible for the following:-

1. Grading team submissions.
2. Providing clarifications to teams.

3.1.3 Administrator

An administrator is responsible for the following activities:-

1. (Re) starting the server.
2. Creating users.
3. Setting problems for the programming contest.
4. Freezing the scoreboard.
5. Recovering from a server crash.
6. Stopping the server.

3.2 Common Use Cases

3.2.1 Pre-conditions common to all use cases

1. All the components of the application namely, the ACM server, Answer queue and the database are up and running.

3.2.2 ACMLogin

Actors involved: All actors have to go through this use case.

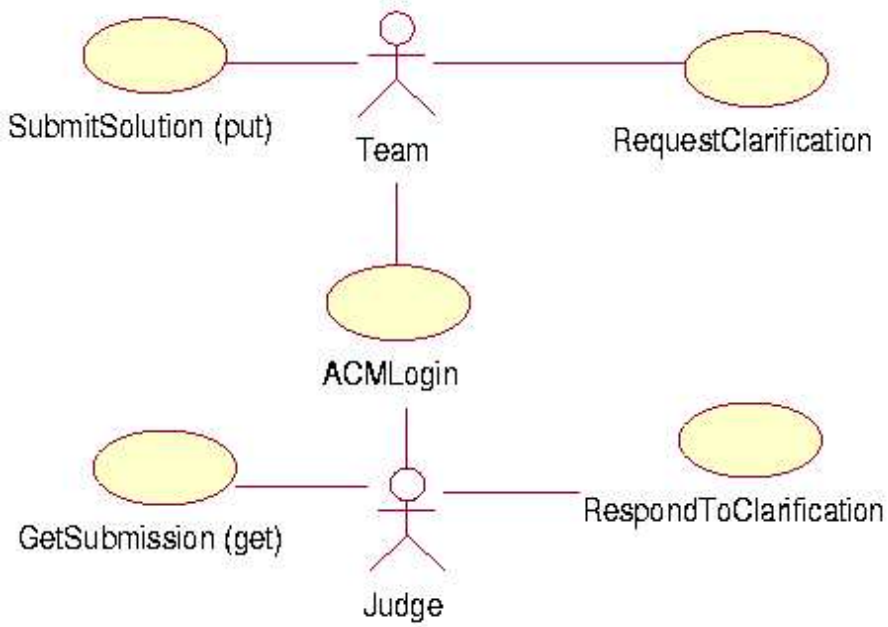
Pre-conditions:

1. Administrator has performed the *CreateUser* use case for the user, therefore an unique username and password is available.

Main flow of events: The use case starts when the user provides the username and secret password in the login dialog. If the username and password is verifiable and user possesses the relevant roles, a welcome screen is displayed. Role-specific menu options are visible on the left-hand frame of the web page.

Exceptional flow of events: The user is given as many opportunities as required to provide the correct login information. For security reasons, no specific error message is displayed during the denial.

3.3 Use Cases for a Team



ss

Figure 1: Use Cases for Teams and Judges

3.3.1 SubmitSolution (Put)

Actor involved: Team

Pre-conditions:

1. Team is ready with the solution to a problem. It could also be another attempt at the same problem.
2. Team chooses the question number to upload from the drop-down list in the web page.

Main flow of events: The use case starts when the team wishes to submit the source file containing the solution to the chosen problem and the corresponding build file. Team confirms the action by choosing 'Submit' option. The files are uploaded to the ACM server using HTTP form submit. The ACM Server updates the Answer queue with the latest submission information, stores the uploaded files in an archive file format and notifies all scoreboards about the submission.

3.3.2 RequestClarification

Actor involved: Team.

Pre-conditions:

1. The programming contest is in progress and the team requires a clarification to one or more of the questions in the contest.

Main flow of events: The use case starts when the team enters the text of the required clarification in the text box provided in the web page and presses the submit button. The clarification is sent to the judge using mail as the mode of transport.

Exceptional flow of events: The team is prompted with an error message displaying the reason due to which the clarification could not be submitted.

3.4 Use Cases for a Judge

3.4.1 GetSubmission (Get)

Actor involved: Judge.

Pre-conditions:

1. A button has appeared in the judges' scoreboard applet.

2. Judge is responsible for grading this problem.

Main flow of events: The use case starts when the user clicks on the button. A file dialog appears and the judge is asked to provide the directory to download the archive file. The judge has to extract manually the file under an empty directory. The judge compiles the source files using the build file submitted by the team and compares the result of program execution with the correct results. The judge is redirected to the grade solutions page where the grade is entered based on the output of the program. If the output is incorrect the judge, chooses the closest reason. All team scoreboards are automatically updated.

3.4.2 RespondToClarification

Pre-conditions:

1. The team requiring the clarification has successfully completed the *RequestClarification* use case beforehand.

Main flow of events: The judge can use any mail client to reply to team clarifications and send it to all teams (using a suitable mail alias). This mechanism allows the team to have a reply address and view all clarifications (including the one's requested by other teams).

3.5 Use Cases for an Administrator

3.5.1 FreezeScoreboard

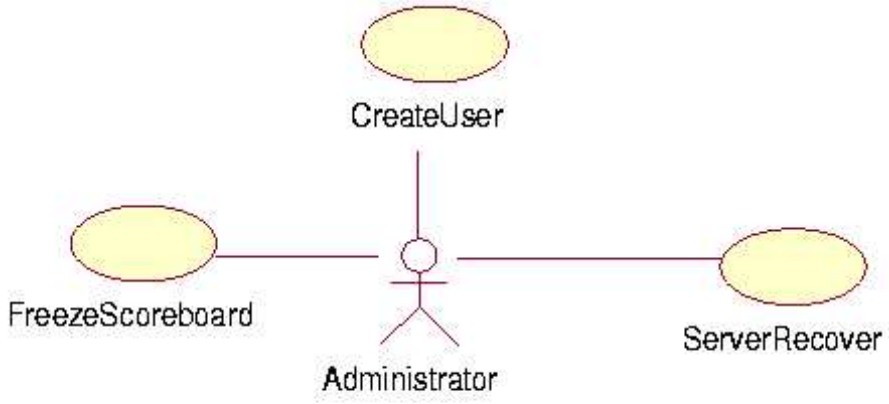
Actor involved: Administrator.

Pre-conditions:

1. The programming contest is in progress.

2. The user has the relevant administrator role to perform the operation.

Main flow of events: The use case starts when the administrator logs into the system and chooses the 'Freeze Scoreboard' option. This will result in the scoreboard to be frozen for until the time, all the judges have come to a conclusion about the winners of the contest. To indicate such a period of inactivity, the time at the server will be displayed on the individual scoreboards. Note that only team scoreboards are frozen, judges can continue with the grading process.



ss

Figure 2: Use Cases for an administrator

3.5.2 ServerRecover

Actor involved: Administrator.

Pre-conditions:

1. The user has the required administrator roles (is authorized) to perform recovery operations.

Main flow of events: The use case starts when there has been a server crash and the administrator and is attempting to restart the whole application. The administrator starts the ACM server providing the command line arguments to inform the server that it is attempting to recover from a server crash and not start new contest.

Post-conditions: All clients will have to login back to the system.

3.5.3 CreateUser

Actor involved: Administrator.

Main flow of events: The use case starts when the administrator requires to add an user of the application. The administrator needs to provide the user name, password and full name of the user to be added. The administrator needs to assign additionally one role to the newly created user. Roles

are discussed under section 6.1.

Post-conditions: The newly created user can access the application.

4 Proposed Architecture

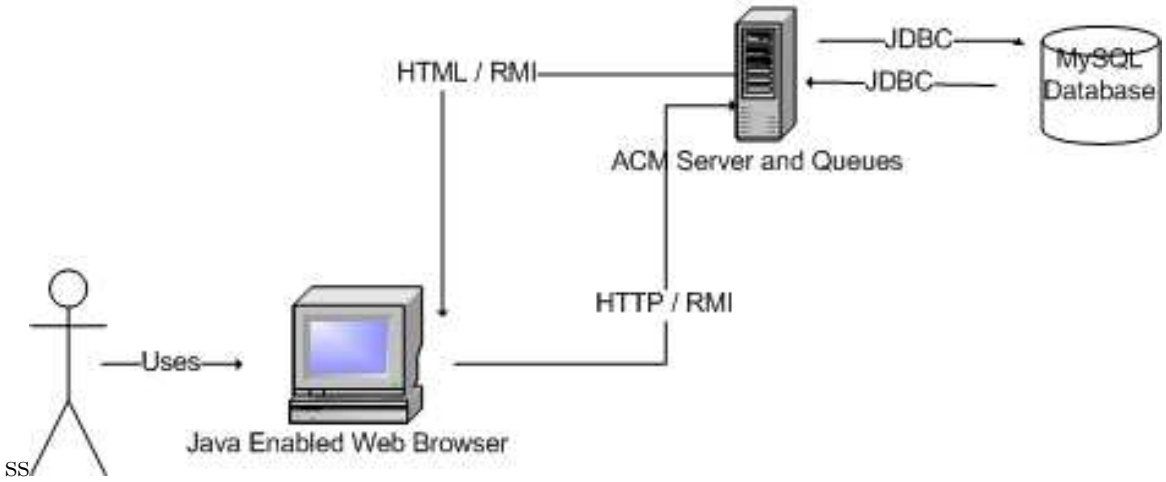


Figure 3: Architecture Diagram

The ACM application is based on the 3-tier model with the front tier being any Java enabled web browser, middle tier collectively being the ACM server, the 'Answer' queue (Java Servlet) and the final (data) tier being the relational database to log the contest activity and persist queue information. Such a database should be capable of allowing data access using JDBC protocol. The choice of a relational database makes the task of storage and retrieval of data convenient. Additionally, this choice also makes the task of creation of reports in the future easy.

Each team can only use one client at a time because of resource constraints. In case of the web-interface client, the data from the client is sent to ACM server when the client submits the web form using HTTP post mechanism. The submitted information is finally maintained in the database. The

teams have a way to view the scoreboard [2] with up-to-date standings, implemented using Java Applet technology. Once any changes in the team scores occur after judges have graded the answers, all the client scoreboards are automatically refreshed. Standings in the client-scoreboardś are automatically updated by the ACM server using Java RMI technology.

ACM application requires that the user be authenticated and authorized to access the application. Simple password matching does authentication, whereas authorization is verified by checking the roles assigned to the user.There will exactly be a single ACM Server servicing all client requests.

Teams upload the files which are archived and enqueued at the Answer queue by the ACM Server. Judges are notified of submissions by the scoreboard. Judges download the archive, compile, execute and compare it manually with the correct output before assigning grades. There will also be a facility to release submissions by judges if it was downloaded by mistake. Teams can seek clarifications from judges during the contest. Clarifications submitted by the team using HTTP post mechanism is transported to the judges and back using mail protocols like SMTP and POP3/IMAP. Secure versions of these protocols could also be supported. Though this implementation will be deployed on Apache Jakarta Project's Tomcat Application Server, it can be ported on any J2EE compliant application server.

4.1 Application Components

The components of the ACM application are listed below:-

4.1.1 User Interface

Any Sun's Java enabled web browser could be used as the user interface for the ACM application. Cascading Style Sheet (CSS) technology will be utilized to ensure consistent style. Use of JavaScript will be avoided because of web browser portability issues.

4.1.2 Application Server

Apart from returning dynamically generated HTML pages to the user, the application server also provides important services like naming and directory look-up, database connection pooling and mail services.One such page would contain an applet that would display the most up-to-date team standings. Additionally it also performs the task of a web server.

4.1.3 Answer Queue

The Answer queue is a Java Servlet implementing queue funtionality. It will enqueue team solutions on a First In First Out (FIFO) basis.

4.1.4 ACM Server

ACM server is a Java Servlet application. The ACM server (or servlet to be precise) is responsible for refreshing the scoreboard applet with the latest standings. Java Applet security permits an applet to "open a network connection to the host that provided the .class file" [3]. Communication between the applet and the servlets happens using messages and object serialization. The protocol for such a communication is only known to the parties involved namely the applet and the servlet.

4.1.5 Database

The database used in this application permanently stores all the contest information.The database is a very important component of the application given the fact that logging activity and queues are implemented using underlying database tables. Such a database should allow application access using JDBC protocol. The character-based fields have been restricted to a maximum size of 4000 characters so that data could be ported to other popular databases like Oracle in the future.

5 Concurrency issues

As per Sun's Java Servlet specifications, a Servlet container may invoke the service method concurrently for each client request. The onus of taking care of concurrent access is on the servlet itself.

It is essential that there is only one ACM Servlet servicing client requests. Java's now deprecated Single Thread Model guarantees that only one request thread is in the service method. However, it is upto the Servlet container to obtain a servlet reference from a pool of instantiated servlets. For such a reason, Single Thread Model is ruled out. Therefore, the ACM Server and the Answer Queue is implemented as a distributable Generic Servlet. Synchronizing the entire service method will delay all client requests. To avoid this overhead, the task of each request will be delegated to a Singleton class having synchronized methods that handle individual requests. [4]

6 Design

6.1 Roles required for application authorization

ACM application requires that the user be authenticated and authorized simultaneously. Authorization is verified with the help of roles. Roles allow users to gain access to protected resources like web pages, files etc. Three distinct roles have been identified. ACM application will utilize existing application server infrastructure to perform authorization.

6.1.1 Judge

A person with this role can participate as a judge in the contest. A judges' primary job is to grade team submissions. The responsibility of providing clarifications to the teams also rests with the judge.

6.1.2 Team

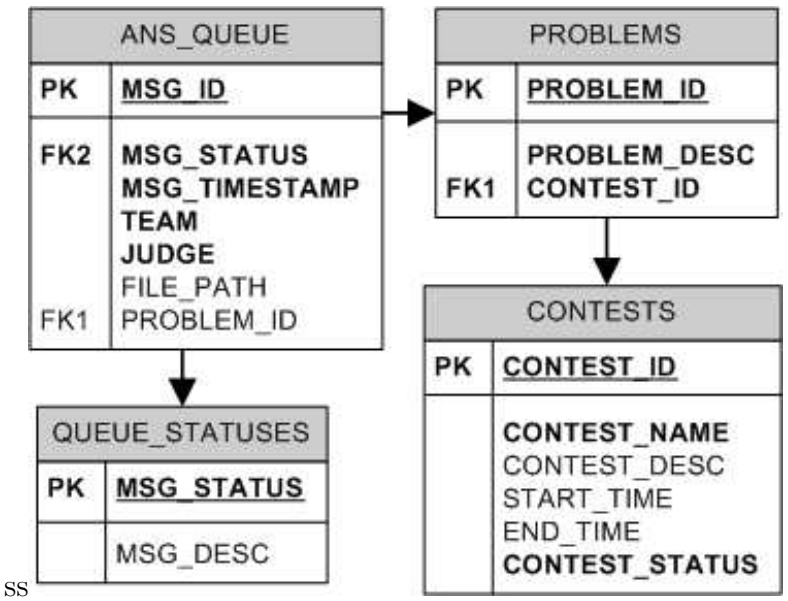
A user having this role can participate in the programming contest. A team is made up of one or more participants.

6.1.3 Administrator

A person with this role can perform all the responsibilities of a system administrator like freezing the scoreboard, starting the server, recovering from a server-crash, creating and authorizing user access etc.

6.2 Entity Relationship diagrams

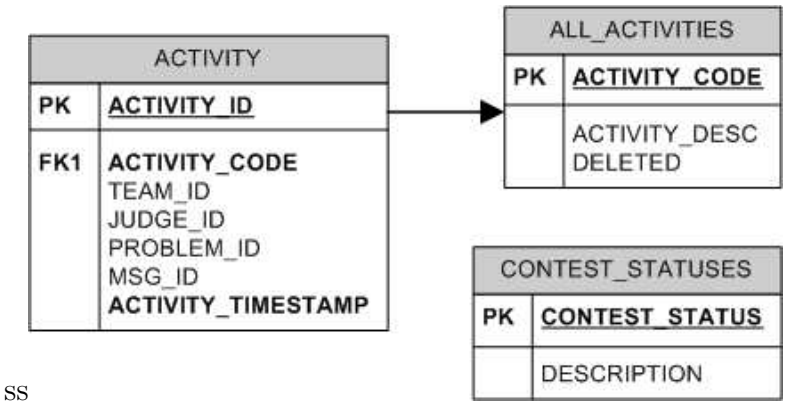
Logging information and queue information is maintained in underlying database tables. The ACM application is made up of a single queue namely the 'Answer' queue. The PROBLEMS table maintains a list of all questions that can be asked in a competition. Problems are maintained per competition; therefore the same problem will be identified by their unique and therefore different PROBLEM_ID. The CONTESTS table maintains the name of each competition. The ANS_QUEUE on the other hand maintains information about each submission a particular team has made to the application. The details of the submissions like timestamp information and status are also maintained. Any message in this queue can have the following distinct status submitted, grading and complete. The information about the judge who graded the teams solution will also be maintained.



ss

Figure 4: Queues

The ALL_ACTIVITIES table maintains information about all the tasks that the ACM application records like adding a problem, requesting a clarification, signing up etc. Each of these activities maps to a unique ACTIVITY_CODE. This information will be referenced by the ACTIVITY table during logging and if required, during recovery purposes.



ss

Figure 5: Logging

The ACTIVITY table is central to the ACM application. All the actions performed by the teams, judges and the administrators will be maintained in

this table. Depending on the activity being logged, user information of the judge and/or team will also be maintained. The scoreboard applet will use information from this table to render the scoreboard. Since all the details are maintained in the database specifically the ACTIVITY table, the only thing to be recreated during a server crash will be the scoreboard itself. The ACM server application needs to be started with an additional command line flag indicating a recovery and the name of the contest.

6.3 Logging and Failure Recovery mechanism

As mentioned above, all activities like request for clarification, submission of solution, response to a clarification etc. will be maintained in the ACTIVITY table of the database. In case of a recovery after a server failure, the system needs to just go thru the relevant records in the ACTIVITY table based on ACTIVITY_CODE's to reconstruct the whole scoreboard. Since the data is maintained in the respective tables no special recovery operations need to be done apart from re-starting the ACM server application itself.

6.4 Clarification Request and Response mechanism

One important requirement for this application is that teams can seek clarifications from the judges during the contest. Judges are assigned to specific problems. The following steps occur:-

1. Teams fills in the provided text-area and submits the web form.
2. Unknown and transparent to the team, the form data are sent to the judge using SMTP mail protocol. Mail aliases could be employed for the purpose.
3. Judge receives the mail using POP3 or more advanced IMAP protocol. Secure versions of above protocols like POP3S and IMAPS could also be used if insecure connections are disallowed.
4. The judge responds to the clarification and forwards the message to all the teams, again using mail aliases. This allows the team to have a reply address to ask further clarifications to the judge.

6.5 Updation and freezing of scoreboards

The ACM server using Java RMI will do automatic updation of all client scoreboard's. Recall that all web-based clients have an applet-based scoreboard. Applet technology permits opening of sockets with the host that sent

the applet. This will be utilized to update all the standings automatically as soon as there is change in grades of any one of the teams. The important point to be noted is that scoreboard updates will be an (ACM) server initiated event. When the administrator freezes the scoreboard, it will continue to display the current time to distinguish from a server failure.

6.6 File submission data flow

The following distinct steps occur during file submission:-

1. Team selects the problem number for the solution to be uploaded from the drop-down menu. Using the file dialog, the team provides the source file and corresponding build file to be uploaded and chooses the 'Submit' option.

2. On the server side where the files were uploaded, the files are arranged under the hierarchy described in section 6.6.1. The submitted files are compressed into an archive along with the answer file (against which the teams solution will be matched manually by the judge grading that particular solution).

3. The team's name, problem no are recorded in the ACTIVITY table using the correct ACTIVITY_CODE.

4. The 'Answer' queue will be updated to include this latest submission from the team.

6.6.1 File hierarchy

```
+ submissions
  + Competition Name
                + Team Name 1
                        + Problem Number 1
                                + 1
                                    - Source File
                                    - Build File
                                + 2
                                    - Source File
                                    - Build File
                + Team Name 2
                .
                .
```

etc.

6.7 Grading Submission Flow

The following steps have to be performed by a judge during grading:-

1. When any team submits solution to a given problem in the contest, a button appears in the scoreboard against the problem. This feature of the scoreboard is only available for the judges.

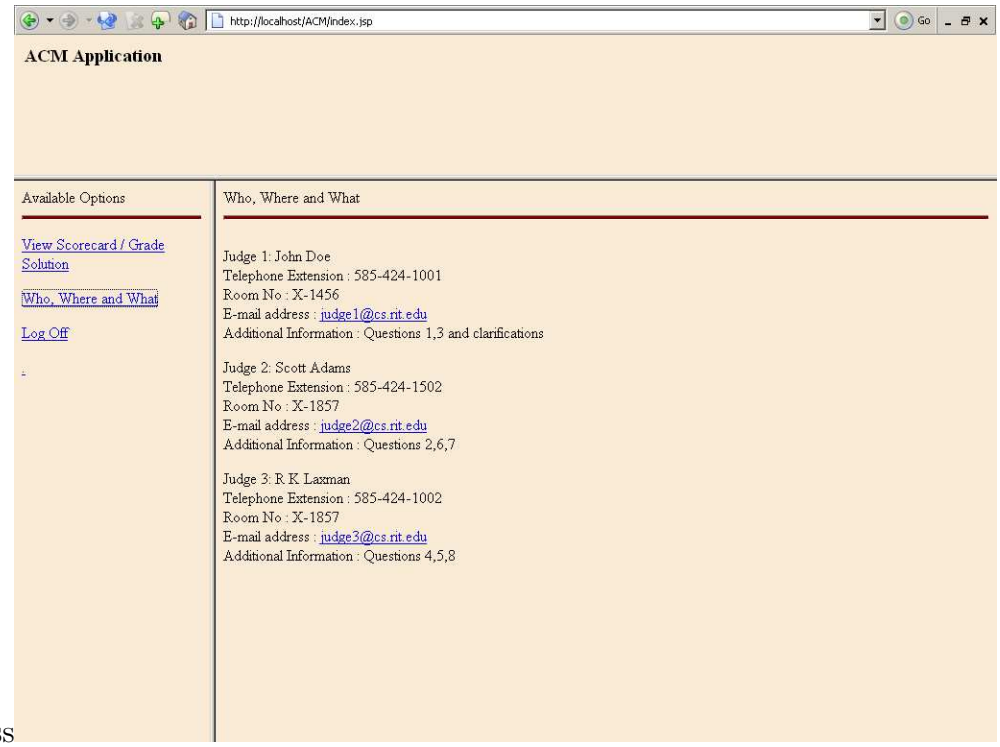
2. When the judge presses the above button, a file dialog appears and the directory where the archive file is to be extracted has to be provided. The judge has to manually delete the existing contents before extracting the contents of the above archive file. The judge is automatically redirected to the 'Assign Grades' screen.

3. The teams program is compiled using the build file and the outputs matched against the answer file (both extracted from the same archive). Depending on the manual output match, the judge assigns a grade to the team and visible on team scoreboard applet.

4. Judges will have an option to release a problem (downloaded team submission) without grading it.

6.8 Screen Shots

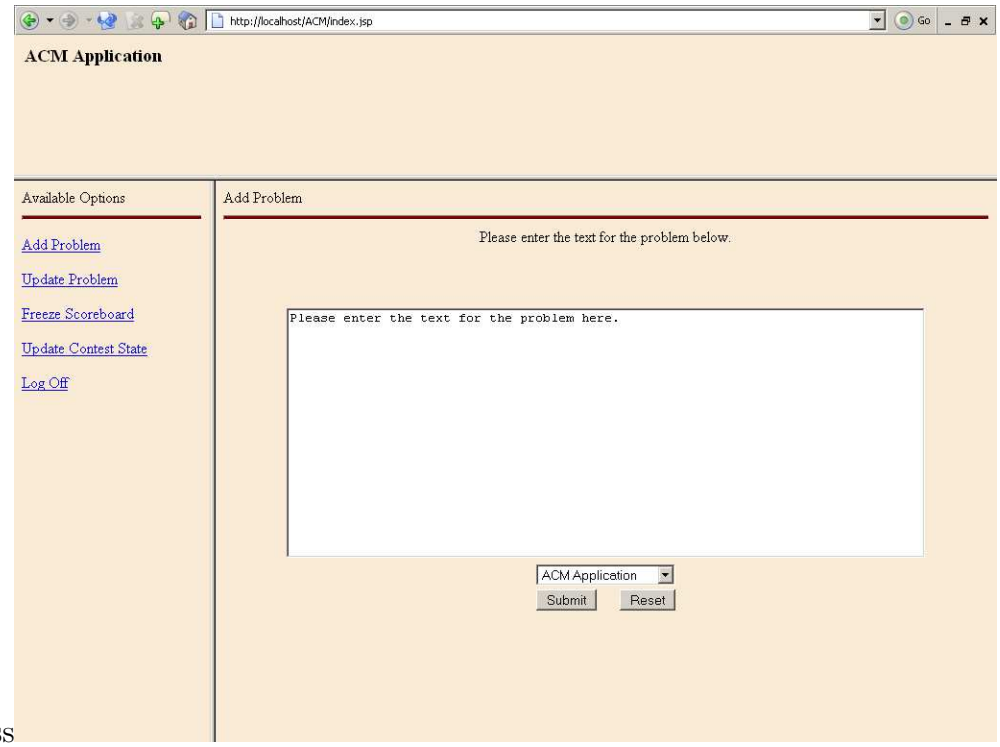
As mentioned before, the ACM application uses authentication and authorization mechanism provided by the application server to protect all protected resources including web pages. During login, users will be automatically challenged with a login dialog requiring a verifiable user name and password. This removes the need for a separate login page.



SS

Figure 6: Who, Where and What

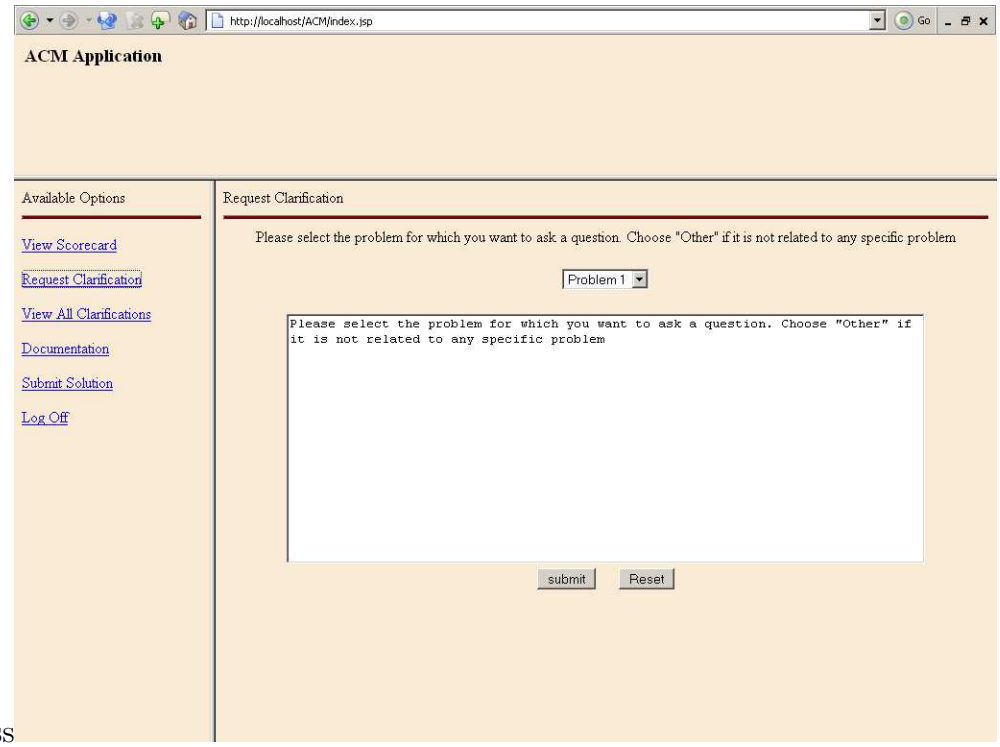
The 'Who, Where, What' page (Figure 6) is only accessible to judges. It enables a judge to get contact information about other judges in the competition. The *Additional Information* section of this page shows the distribution of clarifications among judges.



SS

Figure 7: Add Problem

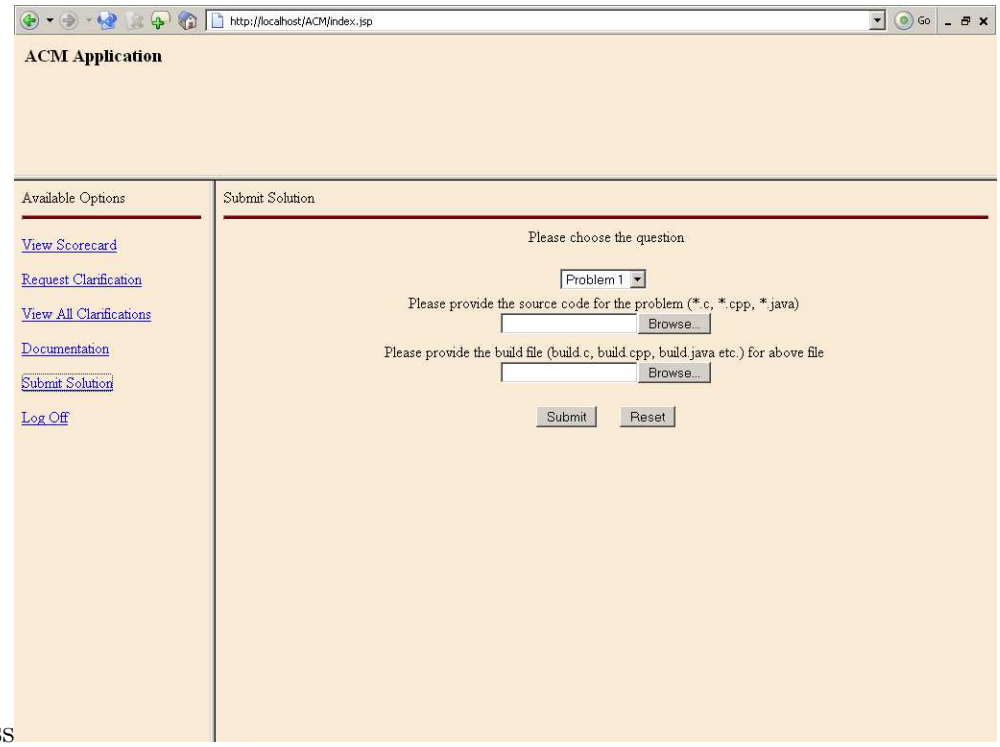
The ‘Add Problem’ page (Figure 7) allows an administrator to add problems for the contest. Problems are classified and maintained on a per-contest basis in the database. A text area in the page allows the administrator to add the problem text. The ’Update Problem’ link allows a judge to modify the problem after being added.



SS

Figure 8: Request Clarification

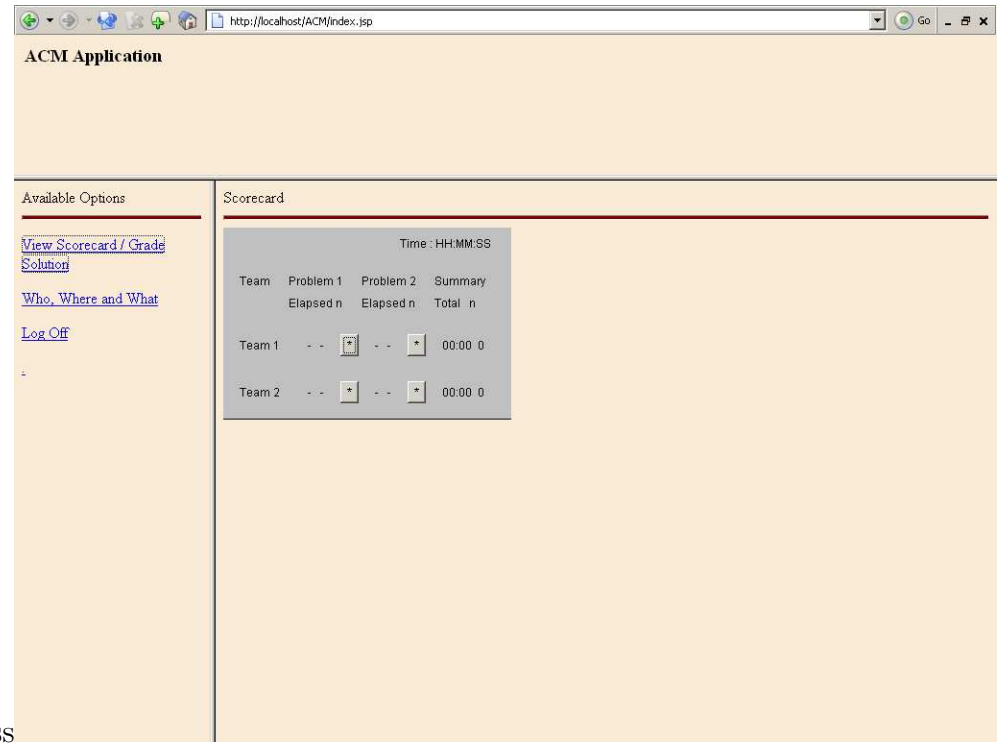
The 'Request Clarification' page (Figure 8) allows participating teams to seek clarifications from judges. The text submitted to the application server is sent to the judge using commonly used mail protocols. Section 6.4 explains the transport mechanism in detail. Teams also have an option to view past clarifications in an orderly fashion in a separate page using the 'View All Clarifications' option.



SS

Figure 9: Submit Solution

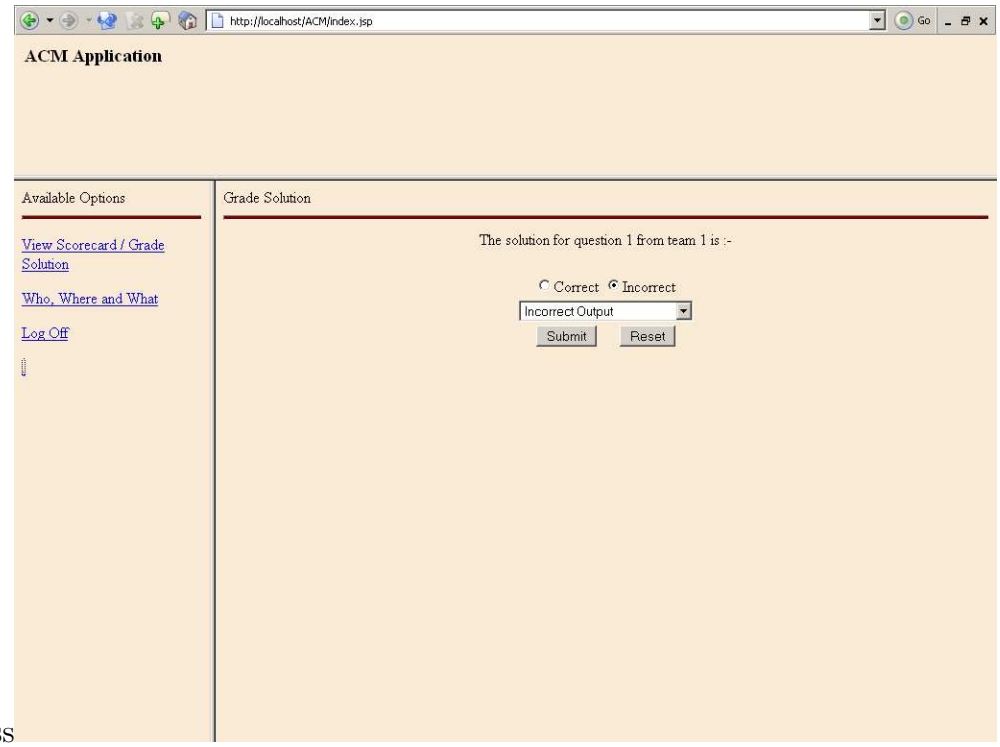
The 'Submit Solution' (Figure 9) page allows any participating team to submit solutions to a problem in the contest with the server. The team needs to upload the source file and the corresponding build file for the selected question. Files are uploaded during form submission. These files are available to the judge during the grading process. The complete flow is explained in detail under section 6.6.



SS

Figure 10: Judge Scorecard

The 'View Scorecard/Grade Solution' option allows (Figure 10) a judge to view team standings and grade solutions. The scoreboard is a Java Applet capable of displaying current standing by periodically communicating with the ACM server (servlet) application running on the web-application server. If the team has recently submitted the solution to a given problem, a button will appear in the scoreboard for that particular team against the problem.



SS

Figure 11: Grade Solution

When the judge clicks on the button, a copy of the team's submission is downloaded to the judge's machine in an archive file format from the server. The hierarchy and contents of this archive file have been explained in section 6.6.1. Note that this option is only available to the judges. Teams will view a similar scoreboard without the above feature.

The judge will have to extract manually the contents of the archive into an empty directory, compile it using the build file provided by the team and match the results with the output of correct answer (also included in the archive). Once the archive file is downloaded, the judge is redirected to the 'Grade Solution' page where grades can be finally assigned to the team.

References

[1] Grady Booch,James Rumbaugh,Ivar Jacobson, *The Unified Modeling Language User Guide*. U.S.A: Addison Wesley, 1st ed., 1998.

[2] Mark Roth, *Scoreboard Application*. R.I.T: Provided by Prof. Paul Tymann.

[3] Sun Microsystems, "Applet Security." Website. `http://java.sun.com/sfaq/`.

[4] Sun Microsystems, "Java Servlet Specification." Website, 2003. `http://java.sun.com/`.