Team Maman

Design Documentation

Diagrams and Requirements

Aaron Andrews, Andrew Torgeson, Micheal Burton, Miles English, Nathan Sargeant, and Kami WIlson
4/20/2012

The Admin and student both create accounts and log into the system.

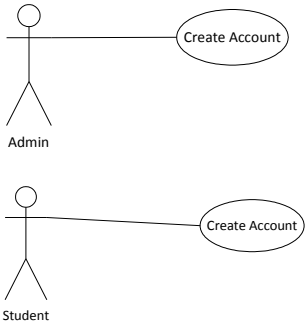
From there the Admin sets up the competition by setting the time, problems and mode. From there the admin starts the competition with the play button. After the competition begins the admin can view various stats such as student progress, team progress, and so on. If at any time during the competition the admin notices that a problem is giving the students trouble, then he/she can send out a hint which the students immediately see. After the competition is over the Admin can save all the stats from the competition if he/she so chooses.

Once the competition starts the students can enter tests cases, view hints, view code coverage, and go to the next or last problem.

Name: Create Account
Participating Actor: User (Student and Admin)
Entry Condition:
- User is on log in screen
- User doesn't already have an account

Exit Condition:
- An account is created for the User

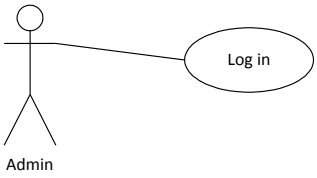
Event Flow:
- User clicks on Create Account Button
- User enters in desired username and password
- User enters if Student or Admin
- User is notified if account creation was successful or not



Name: Admin Log in
Participating Actor: Admin
Entry Condition:
- Admin is on log in screen
- Admin is registered on system

Exit Condition:
- Admin is given access to admin powers

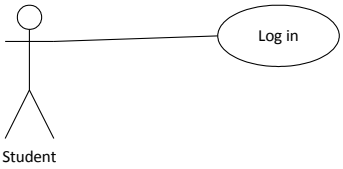
Event Flow:
- Admin enters username and password
- Server checks registration then gives or denies access to admin powers



Name: Student Log in
Participating Actor: Student
Entry Condition:
- Student is on log in screen
- Student is registered on systems

Exit Condition:
- Student is given access to admin powers

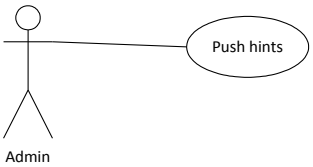
Event Flow:
- Student enters username and password
- Server checks registration then gives or denies access to admin powers



Name: Create and Push Hint
Participating Actor: Admin
Entry Condition:
- There is a competition running
- Admin is on competition overview screen

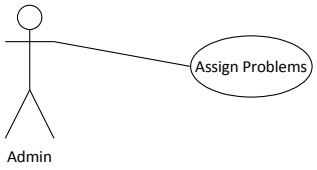
Exit Condition:
- A hint is created and broadcast to each student

Event Flow:

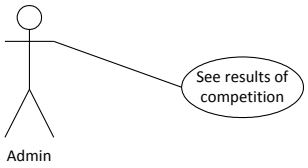


Admin clicks the create hint button
Admin may choose between automated hint or custom hint
Hint is broadcast to everyone
Admin is notified of success or failure

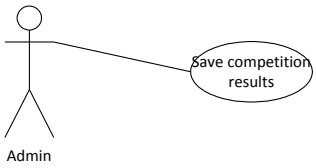
Name: Assign Problem
Participating Actor: Admin
Entry Condition:
Admin is on competition setup screen
Exit Condition:
The selected problem will be used in the competition
Event Flow:
Admin clicks the Select Number of Problems from dropdown box
Admin chooses 1-5 problems for the competition
Admin clicks on Setup Competition



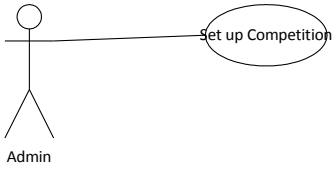
Name: See Results of Competition
Participating Actor: Admin
Entry Condition:
A competition has just ended
Exit Condition:
Results of competition are displayed
Event Flow:
Admin is notified that the competition has ended
Admin clicks Show Results button
Final statistics for each team are displayed in order (highest-lowest) on screen



Name: Save Competition Results
Participating Actor: Admin
Entry Condition:
Admin is on results screen
Exit Condition:
Results are exported to a file
Event Flow:
Admin clicks on Export button
Admin is asked where the data should be saved
Data is saved at specified location



Name: Setup Competition
Participating Actor: Admin
Entry Condition:
Admin is logged in and on setup screen
Exit Condition:
A competition is setup and ready to go



Event Flow:

Admin enters the time limit for the competition
Admin enters time before ready check
Admin assigns problems
Admin adds teams to competition
Admin clicks Finish button
Admin is taken to the waiting screen

Name: Time Before Ready Check
Participating Actor: Admin
Entry Condition:

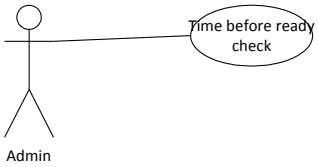
Admin is on competition setup screen

Exit Condition:

Time before ready checks is set

Event Flow:

Admin enters a time for the ready checks



Name: View Progress
Participating Actor: Admin
Entry Condition:

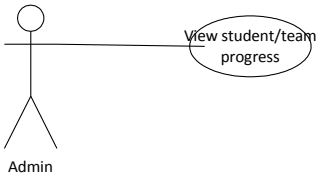
Admin is on competition screen

Exit Condition:

Admin clicks on something else

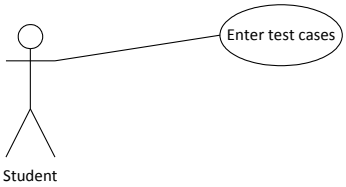
Event Flow:

Admin clicks on team window
Team Window shows all teams and their progress (number of bugs)



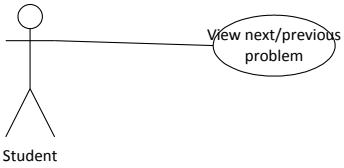
Name: Enter Test Cases
Participating Actor: Student
Enter Condition: Student is in the competition
Exit Condition: Student receives results
Event Flow:

Students enters input/expected output
Test case is checked by system
Student receives result: Found or not found

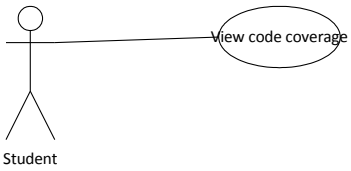


Name: Next/Prev Problem
Participating Actor: Student (Captain)
Entry Condition: Student is in the competition
Exit Condition: Team is navigated to next/previous problem
Event Flow:

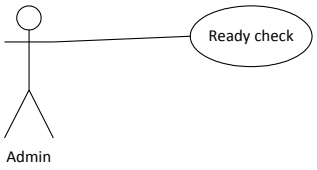
Student press Next/Previous button
Team is redirected to next/previous problem



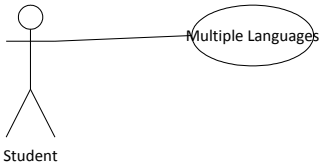
Name: View Code Coverage
Participating Actor: Student
Entry Condition: Student is testing bugs
Exit Condition: Code is highlighted
Event Flow:
Student enters test case
All code run by program is highlighted



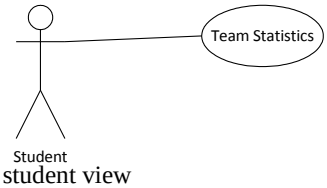
Name: Ready Check
Participating Actor: Admin
Entry Condition: Admin is on waiting screen
Exit Condition: Ready check is sent to students
Event Flow:
Admin presses Ready Check button
Ready check is sent to students
Students confirm or deny readiness
Results are displayed on waiting screen



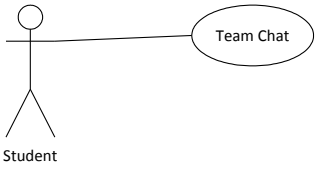
Name: Multiple Languages
Participation Actor: Student
Entry Condition: Student is in competition
Exit Condition: Student receives results
Event Flow:
Admin selects C++ as language for problems
Student clicks on different problems
Student views problem in C++



Name: Team Statistics
Participation Actor: Student
Entry Condition: Competition is has started
Exit Condition: Student exits out of website
Event Flow:
Students from different teams find bugs
The top three teams will be displayed in the header of the student view
If the teams have found the same amount of bugs then the team which found the bugs the fastest will have a higher score



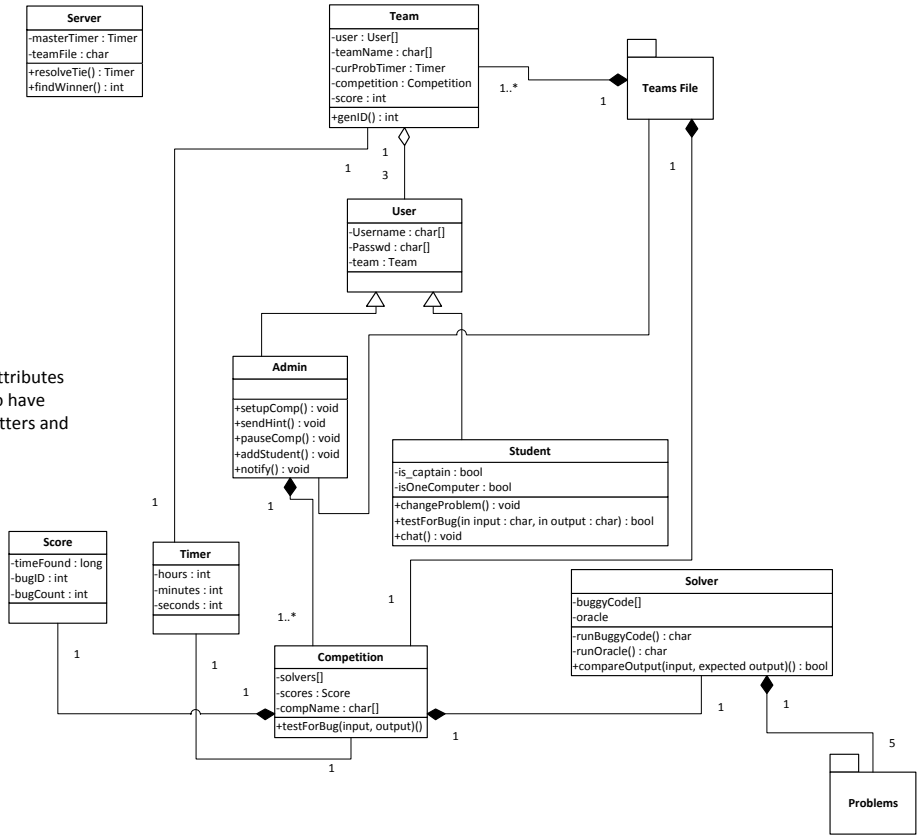
Name: Team Chat
Participating Actor: Student
Entry Condition: Student enters competition
Exit Condition: The competition ends
Event Flow:
Student enters text into the textbox chat and hits enter
Team members are able to view and chat with each other



Classes

~~Team~~
~~Admin~~
~~User~~
~~Student~~
~~Score~~
~~Solver~~
~~Timer~~
~~Competition~~
~~Server~~

All class with attributes are assumed to have appropriate getters and setters.



- There will be 3 users per team and the admins will be on a sudo team of admins provided that there are multiple admins.
- The student and admin inherit from user
- Timer is used by team and competition the timer that competition uses is synced with the server timer which is set by the admin and it tells the student how long they have to complete the competition.
- Competition has a score object and a solver object and each team has a competition object.
- Each team in the team file will receive the admins description of the competition.
- Each team will keep track of their own score via the competition object which will in turn be sent to the administrator for viewing.
- For each competition as a whole there will be one competition file.

Requirements

I. Introduction and Context

The purpose of this system is to create a simple and easy to use software testing/bug finding.

Many schools want to test students' bug finding skills, but lack the ability to stage a competition effectively. The system will simplify the way that schools can test student's abilities by making it all automated. The system will allow schools ranging from middle/high school to graduate school levels to set-up and execute competitions to test student's abilities at finding bugs. The system will simplify this process.

II. Users and Their Goals

The users of this system will be the competition administrators and the students participating in said competition. Admins will only require minimal computer literacy. Students will need little computer knowledge to use the software, but considering it is a bug finding competition, the students should know enough for their respective skill level. The goal is to allow schools a way to test students bug finding skills.

III. Functional Requirements

3.1 Admins can log in with usernames and passwords.
3.2 Admins can create, modify, and delete teams.
3.3 Admins can view overall, team, and individual progress throughout the competition.
3.4 Admins can create and broadcast hints to all competitors.
3.5 Admins can assign up to five problems for a competition.
3.6 Admins can view different competitions at any time.
3.7 Admins can set up a competition.
    3.7.1 Set up includes time, teams, problems, modes, start, stop.
3.8 Admins can see the results at the conclusion of a competition.
3.9 Admins can save competition results to view later.
3.10 Admins can change the problem language (Java or C++) for competitions.
3.11 Students can register an account, creating a username and password.
3.12 Students can log in using the registered usernames and passwords.
3.13 Teams of students can use up to 3 separate machines for the competition.
3.14 Students can see ~~see~~ the time until the competition starts on a "get-ready" screen.
3.15 Students can participate in a competition mode with requirements and code enabled.
3.16 Students can participate in a competition mode with requirements, code, and code statement coverage enabled.
3.17 Students can participate in either of the two competition modes and receive hints from the admins.
3.18 Students can view the progress (number of bugs found) the whole team has found as well as the progress of the lead team.
3.19 Students can enter test cases (input and expected output) and see the real output.
3.20 Students will be notified when they find a bug.
3.21 Students can see the time left in the competition.
3.22 Students can test in competitions in either Java or C++ depending on Admin settings
3.23 Teams can navigate between the competition problems at will.

3.24 Teams can view their progress throughout the competition. The top three teams will be displayed on the top of the screen according to bugs found and speed of finding them.

3.25 Students view an animation each time a bug is found

Formatted: Indent: Left: 0.41", No bullets or numbering

Formatted: No bullets or numbering

IV. Non-functional Requirements

4.1 The team set-up data will be stored in a single file.
4.2 The results of the competition will be stored in a single file.
4.3 The system will require an internet connection.
4.4 The system will use a graphical user interface.

V. Extra Requirements

5.1 PHP 15%
5.2 Team Chat
5.3 Extra languages in problems (C++, C#, PHP)
5.4 Create Teams in Bulk (read from file)
5.5 Profanity Filter
5.6 Save Statistics about Competition
5.7 Special Options Menu (push hints automatically, etc)
5.8 Help Menu
5.9 Tutorial During Wait Screen
5.10 Ready Check
5.11 Multiple Admins
5.12 Anti-Cheat System
5.13 Disqualify Team
5.14 Lock/Unlock Team
5.15 Student Location Information
5.16 GUI
5.17 “Hardcore” mode
5.18 Disabled Person Support (color-blind mode)
5.19 Easy on Eye colors
5.20 Advertising
5.21 Animations/Sounds
5.22 Encryption of Data