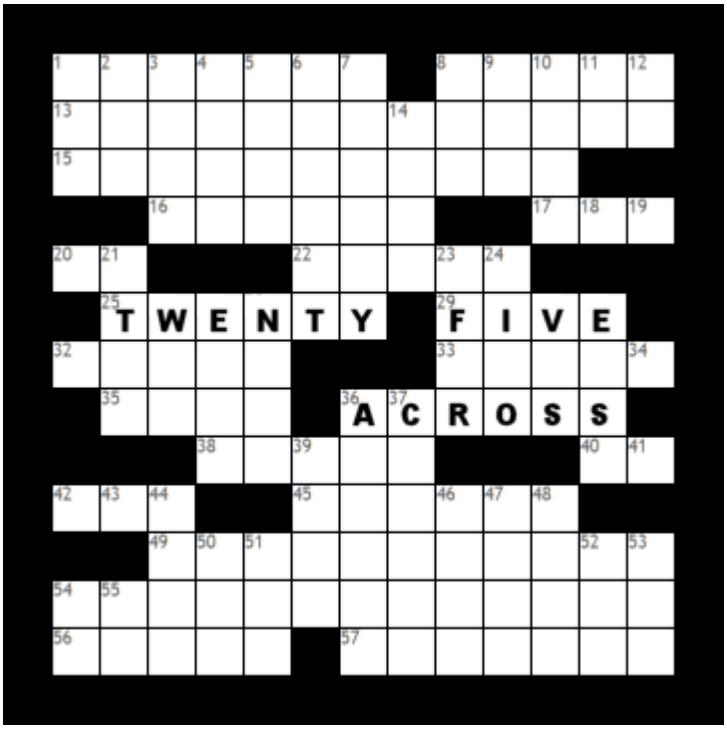
Twenty-Five Across



Project Concept

10/5/2010

Muhammad Ali Akbar (maa2206)
Edlira Kumbarce (ek2248)
Chisom Maduike (cnm2113)
Peyton Sherwood (ps2597)

COMS W4156: Advanced Software Engineering
Prof. Gail Kaiser

Basic concept

Twenty-Five Across is a collaborative crossword puzzle solving game. Two players work together to solve a crossword puzzle; each user can see the other's modifications to the board in real-time. The number of words completed by each player is tallied. While such a "score" is kept, the main objective for both players is to finish the puzzle. In this way, Twenty-Five Across is as much a collaborative effort as a competitive one. Other users may observe any ongoing game in a read-only viewing mode. Game information persists through both client and server restarts, so players may leave (intentionally or accidentally, e.g., an accidental disconnection from the server) and return at any time during the game. Besides solving the puzzle to end the game, players may also resign.

Login/Registration Screen

New users have the option to register in the system by clicking on the **Register** link. A small form will be displayed asking the user to fill out basic information and then click on the **Submit** button. Once a user has been registered, he/she is redirected to the login screen.
Existing users can enter their username and password and click on the **Log In** button. If the login information supplied is incorrect or blank when the **Log In** button is clicked, a message is displayed indicating the problem and the user is prompted to re-enter the login information.

Home Page

Upon successful login, the user is directed to a **Home Page**, where a list of ongoing games is displayed. The **Home Page** also contains options for the user to start a new game or join an ongoing game (provided the game "table" is not fully occupied, i.e. there are not two users already playing the game). The user may join an ongoing game either as an active player or as a spectator. Options to view game statistics are also available from the **Home Page**. If the user is an administrator, links pointing to **User Management** and **Game Management** sections are also displayed. An option to **Exit** the game room is also available.

Game Play Screen

The **Game Screen** displays the board of the game currently in session, the list of clues, the number of players solving the game, and the number of users viewing the game.

Puzzle Board

All users connected to the same game will be able to see the puzzle board. Only the active players can make changes on the puzzle board.

Surrender

Any player can surrender the game by clicking the surrender button. Victory will be awarded to the player still playing the game.

Clues

Clicking on any number on the puzzle board will show the clue for the corresponding word. If there are words in both across and down direction, first click will show the clue for word in across direction, and the next click will show the clue for word in down direction.

Finish

When a player is satisfied with the choices and do not want to make any more changes, he/she can click the Finish button. When both users have clicked Finish, the scores will be calculated and displayed.

Administrator Screens

Users with **Administrator** roles have access to additional screens from where they can manage the system's users and games. These screens are hidden from regular users.

User Management

The **User Management** page contains the complete list of the system's users, along with their respective role. The administrator has options to create, edit, or disable user accounts.

Add New User

A blank form appears with fields such a Username, First Name, Last Name, Role, Password, etc. Once the administrator has filled in the required information on the form, clicking on **Submit** will add a new user on the system.

Edit User

Next to each existing user there is an option to Edit that user. Clicking on it displays a form similar to the one seen when adding a new user, except the fields are now filled in with the selected user's information. The administrator edits the fields and clicks on **Submit** to save the new information for the selected user.

Delete User

Next to each user, there will be a Delete User option. The administrator may decide to delete a user from the system. This will erase the user's account from the system.

Game Management

The **Game Management** page will allow the administrator to manipulate various properties of games. The administrator will be able to kick any user (player or observer)

out of the game's room or ban users, disallowing them to play games. He will also be able to forcibly end games.

Ban User

Next to each user, there will be a Ban User option. The administrator may decide to ban a user from playing any more games due to foul play. The user will not be able to participate in any further games, even though his/her user account still exists in the system.

Un-ban User

Next to each banned user, there will be an Un-ban User option. The administrator may decide to allow a previously banned user to participate in games again.

Component Model Framework

We have decided to use Enterprise Java Beans (EJB), probably version 3.1 on a Windows platform. EJB 3.1 comes bundled with Java Enterprise Edition 6 (i.e., version 1.6), available at: http://www.oracle.com/technetwork/java/javaee/downloads/index-jsp-140710.html

We briefly considered using Spring, an alternative framework with many similar features to EJB. Spring was developed largely as a response to perceived shortcomings in earlier versions of EJB. It has become very popular in the recent years and offers RPC, transaction management, a data access framework (including support for JDBC) and messaging, among a host of other things of the sort we've been talking about in class. Beyond this, though, Spring allows for more flexibility, particularly in the area of transaction management, because it allows the programmer to integrate a number of different transaction management APIs, including JDBC and Hibernate, in addition to the JTA API that EJB allows. Primarily, we considered Spring because of its current popularity in the community -- we thought it would be nice for each of us to have experience with something that is *en vogue*.

There are a few drawbacks to using Spring, however. EJB is a documented standard, whereas Spring is only a vendor-specific implementation. EJB is also likely to be better documented, and therefore it may be easier for us to solve problems as we come across them. As far as popularity goes, even Spring's advantage here is in question; after the EJB 3.1 specification was released, EJB regained support, increasing its competitiveness with Spring. Additionally, the course staff has experience with EJB, whereas they don't have very much experience at all with Spring; likewise, EJB concepts will be covered in class.

In the end, we decided that our ability to consult the course staff and a more extensive set of documentation when we get stuck outweighed the value of having experience with a new, popular framework. We all agreed that using EJB, particularly the latest version, 3.1, which has been out for over a year, would be the best course of action.

Some tutorial information on EJB 3.1 can be found here:
http://netbeans.org/kb/docs/javaee/javaee-entapp-ejb.html (EJB 3.1 tutorial on NetBeans)

Controversies

Fortunately, we have had no major controversies thus far. As a group, we did struggle with the decision of which framework to use (see above). However, we were able to come to a consensus about which was best to use. We look forward to working together and anticipate great teamwork tackling the other challenges that we will no doubt encounter!