Twenty-Five Across

First Iteration Plan

10/19/2010

Muhammad Ali Akbar (maa2206@columbia.edu)
Edlira Kumbarce (ek2248@columbia.edu)
Chisom Maduike (cnm2113@columbia.edu)
Peyton Sherwood (ps2597@columbia.edu)

COMS W4156: Advanced Software Engineering
Prof. Gail Kaiser

Requirements

1. Non-user-facing infrastructure

1.1 Client and Server

The system shall be implemented as a server process communicating with one or more client processes. The server accepts connections from client processes, authenticates users to play games, keeps track of each ongoing crossword puzzle game and its players/viewers, accepts player actions from clients, and sends back updated information to the relevant clients attached to the same game. The client process is run by each human user to display a graphical user interface through which the user can play crossword games.

1.2 Database Store

The database system resides only on the server. It is used to store information about users, such as authentication information; available puzzles, including their design and clues and answers; ongoing and recently finished games, including the users playing the game, state information about which boxes are filled and by whom, and how many words have been solved by each user.

1.3 Remote Procedure Calls

There are a number of methods which must be called remotely by the client and processed on the server. Please refer to the **Work Breakdown** section for a detailed list of all the methods, their inputs (supplied by the client) and their outputs (transmitted from the server back to the client).

1.4 Messaging System

The system must support a number of "messages" that can be sent by either the client or server to notify the other of certain events. Please refer to the Work Breakdown section for a list with explanations for all the messages.

2. Use Cases

2.1 New User Registration

Priority: High

Clicking the **Register** link on the Login/Registration page allows one to register as a new user. They are redirected to a page where they must provide basic information including username and password. After submitting the registration info, the new user is redirected to the Login/Registration page.

2.2 User Login

Priority: High

In the Login/Registration page, a registered user enters their username and password within the provided fields as well as clicking the **Log In** button to log into the system. Upon successful entry, the user is redirected to the Game Room. If a log in attempt was unsuccessful, the system will prompt the user to re-enter their log-in information.

2.3 Join a Game as a Participant

Priority: High

The Game Room displays all games currently in session. There can only be two players per game. A user has the option of joining an ongoing game by selecting an available game and clicking on the **Join** link. The user is then redirected to the Game Play screen.

2.4 Join a Game as a Spectator

Priority: Medium

A user can view a game without participating by selecting any of the available games in the game room and clicking on the **Join as Spectator** link. If the game is not set to private by the creator of the game, then the user will be redirected to the Game Play screen.

2.5 View Game Statistics

Priority: Low

Within the Game Room screen, a user can click on the **View Game Statistics** link to view overall progress of the user from previous games. This information will be displayed in a separate screen after the **View Game Statistics** button is clicked.

2.6 Exit Game Room

Priority: High

A user can exit a game room by clicking on the **Exit** option. This terminates the user's current session within the program and redirects them to the login page.

2.7 View User Management Screen

Priority: High

In the Game Room screen, an administrative user will have the option of clicking on a link to view the User Management screen. Here the administrator can view all users of the program as well as taking care of several tasks related to user management.

2.8 View Game Management Screen

Priority: High

Admin users will have an option to view a **Game Management** link on the Game Room screen which will redirect them to the game management page when clicked. Here they can handle tasks related to the management of all games.

2.9 Add New User

Priority: Low

In the User Management screen, an admin can add a new user by clicking on Add New User. This process is similar to a user registering for the first time. After filling out the necessary fields and clicking Submit, a new user is registered in the system. The **createUser()** method is called within the client program for this action to occur.

2.10 Edit User

Priority: Low

An admin can edit information regarding a user within the User Management screen. To do so, they must Click on the **Edit User** link next to a user from the user list. A screen similar to the Add New User is subsequently displayed, except the fields are initially populated with the user information the admin wishes to edit. The **updateUser()** method is called within the client program for this action to occur.

2.11 Delete User

Priority: Medium

Within the User Management screen, there is also an option to **Delete** a user. An admin wishing to perform this action must click on the Delete link next to the particular user in the user list. Consequently, after a prompt to verify the action, the user is deleted. The **deleteUser()** method is called within the client program for this action to occur.

2.12 Ban User

Priority: High

The Game Management screen displays all Games in session and allows the admin to make modifications as they deem fit. An admin can ban a user from a game which also extends to playing other games. This action is carried out by clicking on the **Ban** option next to a user. The user will be registered but unable to play any games. The **banUser()** method is called within the client program for this action to occur.

2.13 Un-Ban User

Priority: High

This is the opposite of Ban User. This will allow a previously banned user to be eligible for game play. The **unbanUser()** method is called within the client program for this action to occur.

2.14 Select a Word

Priority: High

Within the Game Play screen, a participating user can navigate through the board by clicking on the first box that makes up a word to have the word highlighted. In instances where the first letter belongs to a horizontal ("Across") as well as vertical ("Down") word, clicking once will highlight the horizontal clue, and clicking again will highlight the vertical clue.
The **msgLocationClicked**, **msgClueClicked**, **msgHighlight** messaging methods are invoked for this action to occur.

2.15 View Clue List

Priority: High

The entire clue list is displayed next to the game board.

2.16 Enter a Guess For a Clue

Priority: High

In the Game Play screen, a participating player can click on a word on the board followed by entering the guessed solution to the clue. Subsequently hitting enter will register the entry onto the board. A player's entries can be overridden by another player. The **playWord()** method is called within the client program for this action to occur. In addition, the **msgRedraw** messaging method is invoked.

2.17 Surrender

Priority: High

A participating player can forfeit the game. In cases of more than one player, this action will cause the other player to win the game. The **surrender()** method is called within the client program for this action to occur.

2.18 Finish

Priority: High

A player signals they have finished by clicking on the **Finish** link. When both players (in cases of multi-player) are finished, the scores are tallied and displayed. The **finish()** method is called within the client program for this action to occur. The **msgGameOver** method is invoked once all players of a game have clicked the **Finish** link.

Work Breakdown

1. Preparatory work

1.1. Set up code repository.

We need to create a repository where all of our development code can reside and get everyone authentication credentials. We have chosen Google Code Project hosting, using an SVN repository, for our project. A folder structure needs to be created for source code, documentation files, etc.

1.2. Choose development environment and tools.

We need to determine which operating system, language, IDE, web/application server, and database package we'll use. All development will be done on machines running a Windows operating system. Since almost all team members are familiar with Eclipse, it is decided that Eclipse Java EE for Web Developers will be used as an IDE. EJB 3.1 is to be used as a component model framework. The database will reside on a MySQL server. The web interface of the system will be served by GlassFish.

1.3. Set up development environment.

All team members need to install or set up Eclipse on their machines, install GlassFish and/or set it up as a server within Eclipse, and install or set up the MySQL server. A local copy of the project is then checked out from the repository.

1.4. Set up a basic system.

A very simple system is set up with a toy database backing a toy server (EJB and servlet) to make sure that basic remote methods can be used by the servlet and their results can be viewed in a web browser.

1.5. Sketch out the class infrastructure.

Prior to the implementation phase, a list of Java classes necessary for the system is compiled and then turned into a diagram depicting hierarchy structure and relationships between classes.

1.6. Sketch out the database schema.

Design database schema with tables needed to store system data, marking primary keys as appropriate.

2. Implementation

2.1. CLIENT

2.1.1. Hack the open-source Crossword Sage

Modify the source code of Crossword Sage into a usable GUI for our project's purposes.

- Add ability to add words to the puzzle (per user).
- Gameplay actions: enter word, check puzzle.
- Color coding & number of words completed per user stats.

2.1.2. Design and code the Login and Registration Screen

Design and code HTML for web page that will display the page where users can log in and register new accounts.

2.1.3. Design and code the List of Games Screen

Design and code the HTML to render the List of Games screen.

2.1.4. Design and code the Game Screen

Design and code HTML for web page that will display a game, potentially in an applet, together with the game's players, clue list, scores, etc.

2.1.5. Design and code the User Management Screen

Design and code HTML for web page that will be used to manage user accounts.

2.1.6. Design and code the Game Management Screen

Design and code HTML for web page that will be used to manage games and game statistics.

2.2. SERVER

2.2.1. Create the database and its tables

Construct and execute SQL statements that create the database to be used in the system, together with its tables.

2.2.2. Implement the Login process handler

Code the functionality that handles the user's log in process, including the authentication of the user's credentials.

2.2.3. Implement the user Registration process handler

Add the ability for users to create new accounts (register) in the system.

2.2.4. Implement persistence layer

Create entity beans corresponding to the records to be stored in the database. Create session beans to handle interactions with the back-end database.

2.2.5. Write RPC methods

Game-related methods:

playWord(): Fill word-space with user's supplied word

Inputs:

- The literal word played by the user
- Location of target word-space (e.g., 3 down)
- A flag to "force" the entry (e.g., if another word is already there or some letters don't match)

Output:

- `true` - successful entry
- `false` - another word already exists there!
- `false` - word too big/too small
- `false` - word letters don't match up
- `false` - other reason

getState(): Client requests puzzle state from server

Inputs:

(This method is executed on the desired puzzle object to update.)

Outputs:

- Puzzle construction data (size of puzzle, location of words)
- Which words/squares are filled in by players, with which letters, and which players filled them in

- Scores of players
- Which players are playing the game and which players and viewers are present

listGames(): Client requests list of games from server

Inputs:

(None)

Outputs:

- List of puzzle games including, for each puzzle:
  - Player(s) currently enrolled
  - State of game: not started, in progress, finished
  - Number of viewers

surrender(): Client indicates that user surrenders game

Input:

(This method is executed on the desired puzzle object.)

Output:

(None)

finish(): Client indicates that user is finished entering words

Inputs:

(This method is executed on the desired puzzle object.)

Output:

If the other player has already called finish() at this point, the solutions and scores for both players are returned to the client. Otherwise, the server indicates that it is waiting for the other player to finish before the game is over.

Game-related methods:

createUser(): Creates a new user

Inputs:

- User's name
- User's login name
- User's initial password
- Any other information tracked by the server

Outputs:

- `true` – creation was successful
- `false` – duplicate user login
- `false` – another server-side problem

updateUser(): Modifies properties for a user

Inputs:

- User's name
- User's login name
- User's initial password
- Any other information tracked by the server

Outputs:

(None)

deleteUser(): Deletes a user

Inputs:

(This method is executed on the desired user object.)

Outputs:

- `true` – deletion succeeded
- `false` – deletion failed

banUser(): Bans a user from playing any games

Inputs:

(This method is executed on the desired user object.)

Outputs:

- `true` – banning succeeded
- `false` – banning failed

unbanUser(): Un-bans a user from playing games

Inputs:

(This method is executed on the desired user object.)

Outputs:

- `true` – restore succeeded
- `false` – restore failed

2.2.6. Implement messaging between server and client(s)

msgLocationClicked: User clicked on spot in the puzzle.

The message will include information about which user, which puzzle, and which location was clicked. This information might be used by the server to tell other clients to highlight the location to indicate that another user is entering a word there.

msgClueClicked: User clicked on a clue.

The message will include information about which user, which puzzle, and which clue was clicked. Again, the server might use this to display certain information about the user's activities to other users.

msgHighlight: Server instructs client to highlight word-space with a user color.

The message will include the user currently editing the word-space, the puzzle, and the location of the word-space. This might be used to indicate that another user is editing a word-space.

msgRedraw: Server instructs client to redraw puzzle.

The message will specify which puzzle to redraw. This message might be sent once a user plays a word and all other users' board displays must be updated.

msgGameOver: Server instructs client that game is over.

The message will include a reason such as puzzle is finished or user ended puzzle prematurely or other termination reasons.

msgRosterUpdate: Server reports the users currently in a puzzle.

This may be used when a user enters or exits the puzzle room.

Schedule

Please note that this is a tentative schedule and may change at any time.

Task	Assigned To	Due Date	Progress
Set up code repository	Peyton	Oct 12	Done
Choose development environment and tools	All	Oct 12	Done
Set up development environment	All	Oct 13	Done
Set up a basic system	Muhammad	Oct 20	In progress
Sketch out the class infrastructure	Edlira	Oct 20	In progress
Sketch out the database schema	Chisom	Oct 20	In progress
Hack the open-source Crossword Sage	Chisom, Peyton	Oct 23	
Design and code the Login and Registration Screen HTML	Edlira	Oct 21	
Design and code the List of Games Screen HTML	Edlira	Oct 22	
Design and code the Game Screen HTML	Muhammad	Oct 21	
Design and code the User Management Screen HTML	Muhammad	Oct 22	
Design and code the Game Management Screen HTML	Chisom	Oct 25	
Create the database and its tables	Edlira	Oct 23	
Implement the Login process handler	Edlira	Oct 24	
Implement the user Registration process handler	Muhammad	Oct 24	
Implement persistence layer	Chisom, Peyton	Oct 26	
Write RPC methods	Edlira, Muhammad	Oct 26	
Implement messaging between server and client(s)	All	Oct 27	
Test client and server	All	Oct 28	
Fix bugs	All	Oct 29	
Test	All	Oct 30	
Fix bugs and finalize	All	Nov 1	
Clean up code	All	Nov 2	

Component Model Framework

Nothing has changed.

Controversies

No controversies.