Master's Project Report

AjaxWordPro: A Multi User Word Processor

Kausalya Madhusudhanan

Advisor: Dr. Raphael Finkel

Department Of Computer Science
University Of Kentucky
March 10, 2008

Acknowledgements

I would like to express my sincere and deep gratitude to my advisor, Dr. Raphael Finkel, Dean of the Graduate Studies, University of Kentucky. His wide knowledge and his logical way of thinking have been of great value for me. His stimulating suggestions and encouragement helped me at every stage of my project. His understanding, encouraging and personal guidance have provided a good basis for this master's project. His extensive discussions around my work and interesting explorations in operations have been very helpful for the completion of my project.

I would like to thank my parents, Mr. Madhusudhanan and Mrs. Jayanthi Madhusudhanan. They have always supported and encouraged me to do my best in all matters of life.

I would like to thank all `FCKeditor` team members for providing a high quality JavaScript editor, which is integrated in my project.

I would like to thank Paul Johnston and his team members for implementing the RSA data security MD5 message digest algorithm, which is integrated in my project.

Kausalya Madhusudhanan
March 10th, 2008.

Contents

1	Background	4
2	Requirements	4
3	Implementation decision	5
4	Implementation details	12
5	Screen shots	19
6	What I learned	29
7	What was hard to implement?	30
8	References	30

1. Background

This project reports on the design and implementation of `AjaxWordPro`, a multi-user web-based word processor using the Asynchronous JavaScript and XML (Ajax) technique. This technique creates web pages that are more responsive than traditional web applications by exchanging small amounts of data with the server behind the scenes, so that the entire web page does not have to be reloaded.

Free text corresponds to unformatted text. Free text seems to be a powerful way of communication, because our day-to-day activity of sending emails, blogging and instant messaging involves free text. A text editor is a program used for editing plain text files. Documents created by a word processor generally contain file-format-specific "control characters" that enable text formatting. Commonly used single-user interactive text-editor applications include Notepad and Emacs, and word-processor applications include Microsoft Office Word and OpenOffice Writer.

People also need groupware applications to perform collaborative or group tasks [1]. People need to collaborate with each other to work on the same document to get things done quickly and efficiently. A collaborative multi-user word processor allows several people to edit a document at the same time using different computers [1].

2. Requirements

The requirements of our multi-user web-based word processor are as follows:

- Provide the look and feel of a single-user word processor.
- Provide basic formatting tools to edit the documents by incorporating `FCKeditor`, an open-source WYSIWYG text editor that can be used in web pages.
- Work from a web browser on any operating system and on any device, irrespective of the user's location and machine.
- Allow multiple users to connect to a central server, create, modify and store text documents on the server and graphically edit the contents of the document using a web browser.
- Display the presence of other users accessing the same document.
- Allow the user to view the content changes of a document synchronously.
- Allow the user to view the lock changes of a document asynchronously.
- Provide implicit release of a lock, if the user is inactive for more than one hour.
- Distribute the contents of a file to all its current users, thereby allowing them to modify the contents, without manually separating the documents into smaller parts.
- Provide secure transmission of username and password.

- Provide secure password storage on the server database.

The additional features of `AjaxWordPro` are as follows:

- It allows the user to modify the contents of the file by holding a lock explicitly on the editing region; the user may decide when to make the updates available for other users.
- It suggests plausible replacements for words that are likely to be misspelled.
- It can perform a full-text word search.

3. Implementation Decisions

3.1 Asynchronous JavaScript and XML (Ajax)

`AjaxWordPro` uses Ajax to build dynamic web pages on the client. Ajax incorporates [5]:

- Standards-based presentation using CSS;
- Dynamic display and interaction using the Document Object Model (DOM);
- Asynchronous data retrieval using XMLHttpRequest.

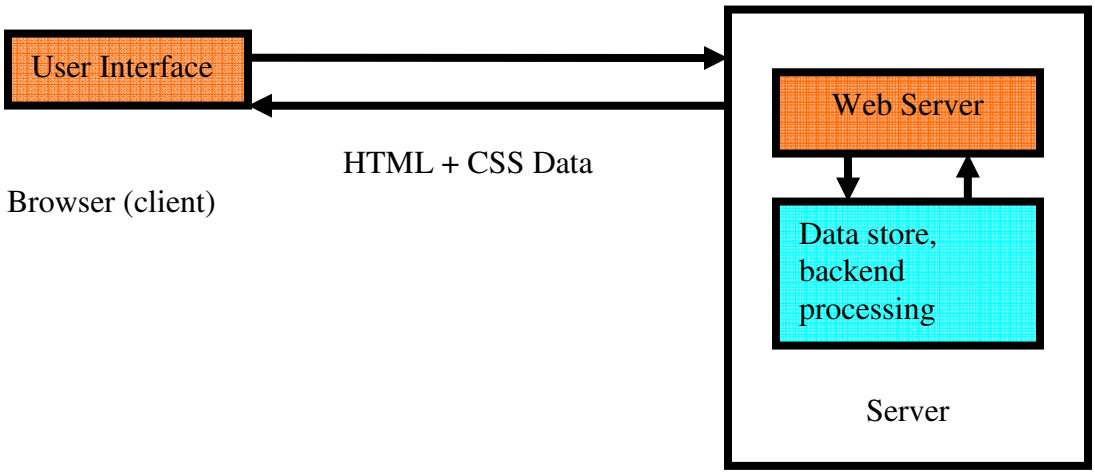
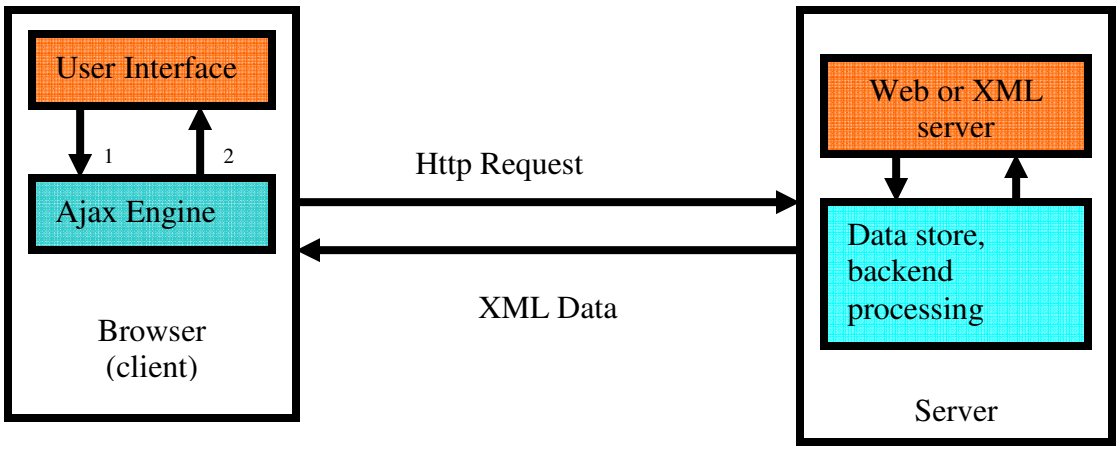
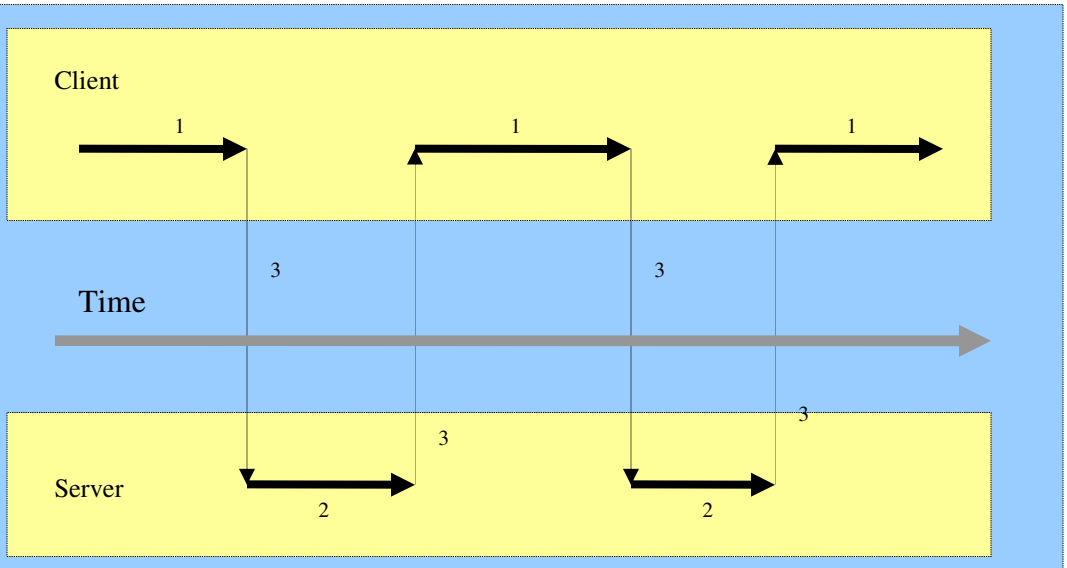


Figure 1[5]: Classic web application model



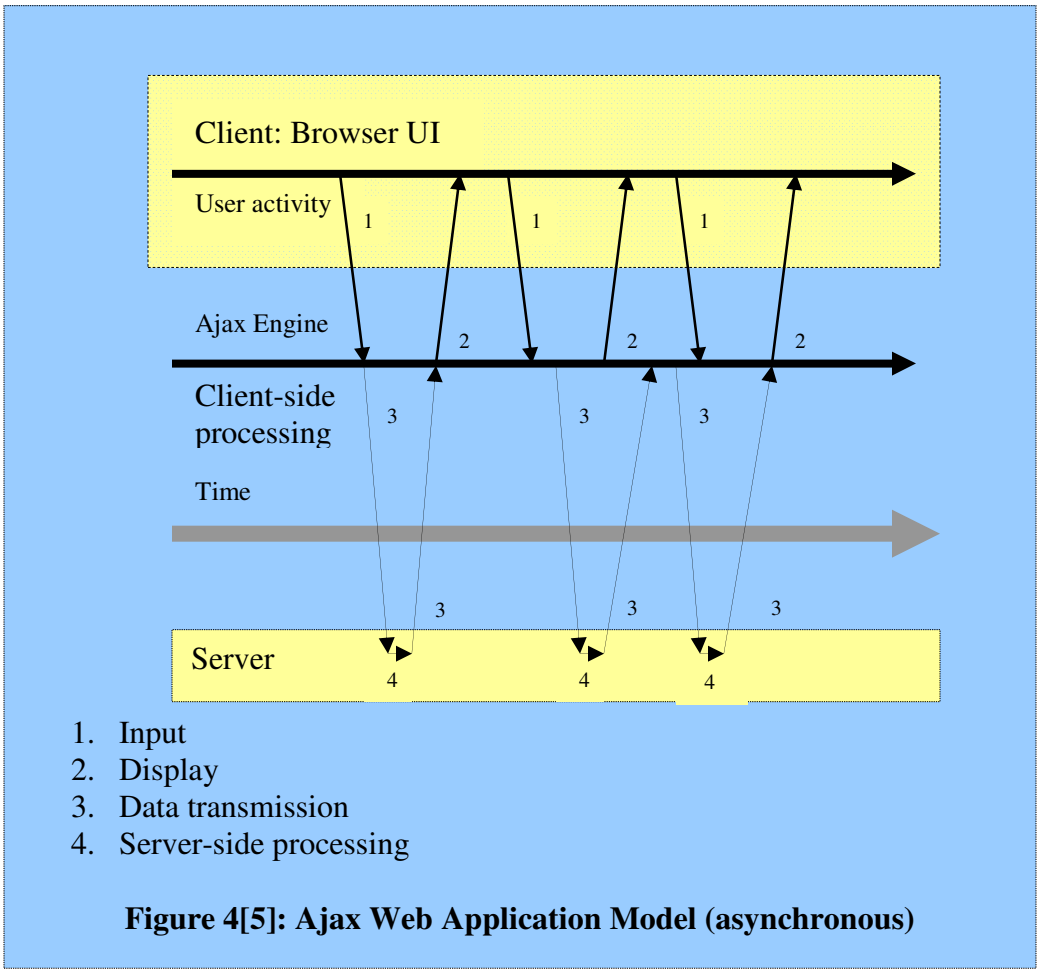
1. JavaScript call
2. HTML + CSS data

Figure 2 [5]: Ajax Web application model



2. User activity
3. System processing
4. Data transmission

Figure 3[5]: Classic web application model (synchronous)



1. Input
2. Display
3. Data transmission
4. Server-side processing

Figure 4[5]: Ajax Web Application Model (asynchronous)

In the traditional web-application model, once an interface is loaded, user interaction comes to a halt every time the application needs something from the server. An Ajax web application eliminates the start-stop-start-stop nature of interaction by introducing an intermediary called an Ajax engine between the user and the server. This engine is responsible for both rendering the interface the user sees and communicating with the server on the user's behalf. The Ajax engine allows the user's interaction with the application to happen asynchronously, independent of communication with the server. Therefore, the user is not blocked until the browser gets the requested response from the server. By exchanging small amounts of data, the web page becomes highly responsive and interactive without reloading [11].

3.2 Scripting Languages

3.2.1 Client-Side Scripting Language: JavaScript

`AjaxWordPro` uses JavaScript as a client-side scripting language. JavaScript provides Ajax function calls. JavaScript sends data to and accepts data from the server using the `XMLHttpRequest` object [11]. The HTML DOM defines a standard set of objects for

HTML, which includes a window object, a document object, a navigator object and a history object. All HTML elements, along with their containing text and attributes, can be accessed through the DOM [14]. A JavaScript program may use the DOM properties and methods to change the contents of a web page on the fly, which enables a web programmer to create interactive web applications.

Examples

1. Setting an attribute of window object can modify the status of the web browser:

```
Window.status = "Here is a new status message";
```

2. Setting an attribute of document object can change the text, URL and target attribute of a link:

```
Document.getElementById('e_id').innerHTML="new_link";
Document.getElementById('e_id').href=
"http://cs.uky.edu";
Document.getElementById('e_id').target="_blank";
```

Where `e_id` is the identifier of a link element

3.2.2 Server-Side Scripting Language: PHP

`AjaxWordPro` employs PHP on the server. PHP stands for Hypertext Preprocessor [12]. It is an object-oriented scripting language that runs on the web server. PHP generated HTML pages, but allows the server to perform computation to insert content into those pages. It has several database functions that enable programs to communicate with databases for storing and retrieving information.

3.3 Database

`AjaxWordPro` stores the information in a database on the server. The data includes the contents of the documents, user information and file information. This project uses the `MySql` database management system [12].

3.4 FCKeditor

`AjaxWordPro` presents editable documents to users via FCKeditor. FCKeditor is an open-source, What You See Is What You Get (WYSIWYG) HTML editor [4]. FCKeditor

is compatible with most internet browsers, such as IE 5.5+, Mozilla 1.3+, Netscape 7+ and Opera 9.5+. It is easy to install and customize for web developers and for web users. Its main features are as follows [7]:

- Multi-browser capability.
- Automatic browser detection and customization.
- Plugin support.
- CSS support for integration with the website.
- Text formatting: alignment, bullets, fonts.
- Cut, Paste, Paste as Plain Text, Undo and Redo.
- Right-click context-menu support.

3.5 Overall design

3.5.1 Concurrency control

`AjaxWordPro` uses exclusive (write) locks for concurrency control. Locks grant write privilege on areas of shared documents [3]. Users need to hold a lock over the data before editing the data.

Although locks allow starvation due to the fact that the server chooses the user to hold the lock in an arbitrary way, the server does not prevent any other user from accessing other parts of the shared document [9]. `AjaxWordPro` notifies the user when other users access the same document, so the users can coordinate their work to avoid starvation.

We choose to lock the document at the granularity of individual paragraphs. When a user introduces a new paragraph, it is considered as part of the previous paragraph for locking purposes. There is an implicit empty paragraph at the start and end of each document.

3.5.2 Server Operations

`AjaxWordPro`'s server, written in PHP, has these responsibilities.

- Authenticate the user.
- Retrieve the requested file from the database.
- Save the contents of a open file into the database.
- Keep track of locks acquired by the users and enforce a locking policy.
- Grant locks that do not conflict with other locks currently held.
- Release locks automatically that are held for more than one hour without activity on the associated data.

3.5.3 Client Operations

The AjaxWordPro client, written in JavaScript and residing in the user's browser, communicates with the server to:

- Establish a connection to the server.
- On user request, send a post submission to the server to open a document, fetch the document contents, save a document, close a document, delete a document, lock a paragraph, unlock a paragraph, and perform a spell check on the contents of the document.
- On a periodic basis, fetch the lock status of the document from the server and display that information to the user.

3.5.4 Client-server communication

When the user performs an action that requires communication with the server, the client creates an instance of the `XMLHttpRequest` type, in which it packages a request that it submits to the server for processing. The client sets up a callback function. The server generates a response and sends it to the client. When the client receives a response from the server, the client executes the callback function, which consults the response and reflects the changes back to the user.

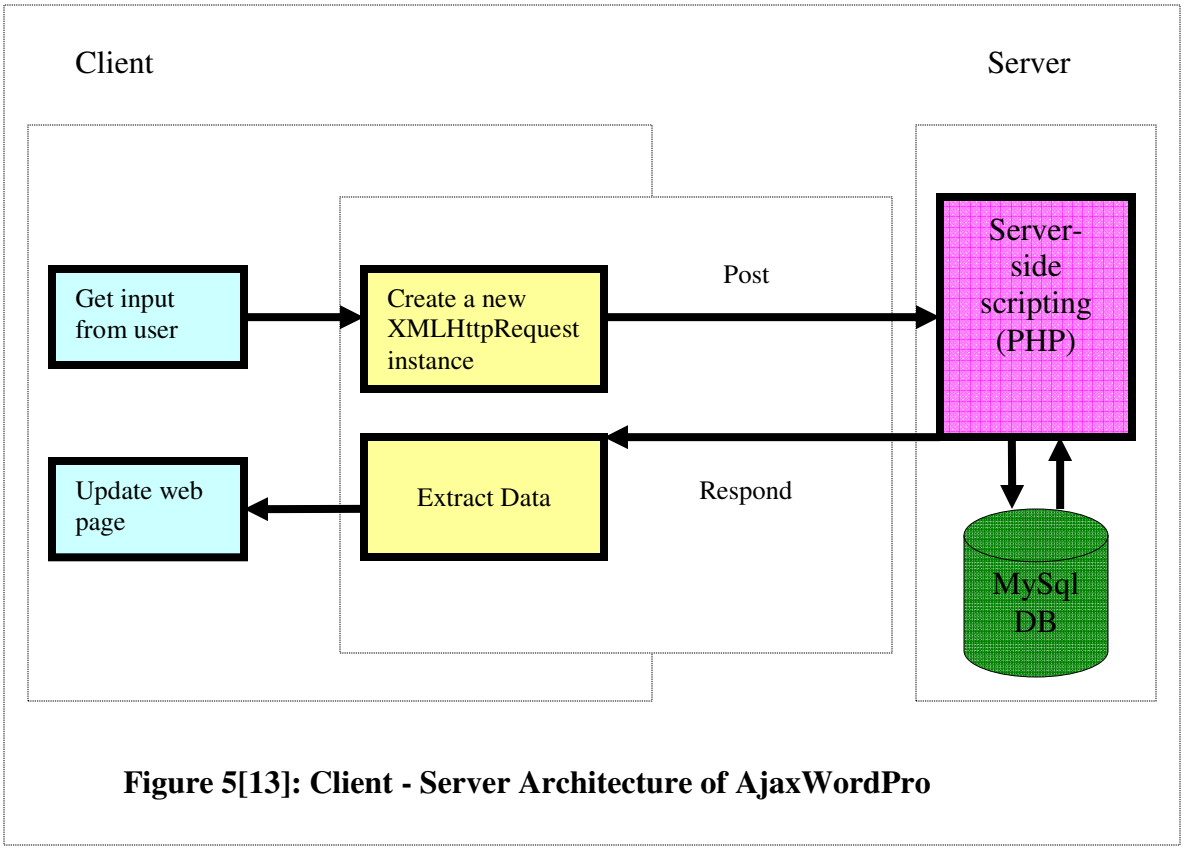


Figure 5[13]: Client - Server Architecture of AjaxWordPro

3.6 Accessing a shared document

- The client sends the user's identity to the server for authentication. Once the user's identity is verified, the user can perform actions to access the shared document.
- When the user opens a document, the client requests the document from the server. The server responds with the content of the document.
- The server also sends information about the locked areas in the document. This information specifies the username and font color of other users editing this document. The client colors the locked paragraphs with different colors to distinguish one user from another. The client's locked paragraph is always green, an inconsistent paragraph (to be discussed later) is always red, and a paragraph that is not locked by any user is black.
- Every client needs to acquire a lock before modifying the document. Whenever a client starts editing the document, the server checks for a lock on the paragraph the client is trying to modify. If the paragraph is not locked, the server allows the client to edit its contents and implicitly grants the lock. If it is locked, the server consults the timestamp of the client holding the lock. People tend to take a break not more than one hour. So, if the client acquires the lock for more than one hour, there are chances that the user computer is down or the browser application is broken making the user inactive for more than one hour. If the user is active for more than one hour with a locked paragraph, then the chances of saving the

contents at least once during the time is very high, which in turn updates the timestamp of the lock preventing the server from automatically releasing the lock. If that client has held the lock for more than one hour, the server grants the lock to the new client. If not, the server denies the lock, and the client disallows editing on that paragraph.

- The user can explicitly requests a lock on the whole document. If the document is not shared by any other user, the server grants the lock for the whole document. If not, the request is denied.
- The client requests lock status every 5 seconds and displays it to the user by coloring paragraphs.
- The user can explicitly fetch the current contents of the document.
- The user can explicitly request a lock on a paragraph.
- The user can explicitly release a lock on a paragraph.

3.7 Secure Socket Layer

`AjaxWordPro` uses the Secure Sockets Layer (SSL) protocol to transmit sensitive information to the server [15]. SSL in the web server provides support for the `https` protocol. When the client requests a secure page, the web server sends its public key with an SSL certificate. SSL Certificate uses SHA-1 with RSA algorithm for encryption and server authentication. The browser checks whether the certificate is still valid and is related to the web server. The browser then uses the key to encrypt a random symmetric encryption key S and sends it to the server. The server decrypts S using its private key. From that point forward, all communication between the client and the server is encrypted with S [18]. Thus, SSL provides mechanisms for encrypting the transmitted data and authenticating the server for security purposes [16].

4. Implementation details

4.1 Database Tables

4.1.1 User Information

The server maintains information about the users in the `user_info` table in the database. Every user is associated with a user identifier, username, password and color of the text used for locking. For security purposes, the server computes the digest of the password using the MD5 [19] algorithm and stores the digest in the database. Since the server stores the digest of the password, a hacker cannot easily retrieve the password from the digest present in the database. Interactive mechanisms for registering users are beyond the scope of this project. Therefore, user information is entered manually in the database.

Field Name	Datatype	Description
Username	Varchar(50)	Name of the user.
Password	Varchar(32)	Digested password: MD5 digest in hex.
UserId	Int(11)	User Identifier. Primary key.
Font_color	Varchar(8)	Color of text to display a lock. The format of the color is '#RRGGBB', where R- Red, G- Green and B- Blue.

Table 1: `user_info` table structure

4.1.2 File Information

The server stores the file information in the `file_info` table of the database. Each file is associated with a unique identifier, file name and contents. Files are named in a flat (non-hierarchical) name space.

Field Name	Datatype	Description
File_name	Varchar(25)	Name of the file
File_data	Blob	Contents of the file
File_id	Int(11)	File identifier: primary key

Table 2: `file_info` table structure

4.1.3 User-File Information

The server maintains a `user_file_info` table to keep track of which user can access which file by storing the user identifier, file identifier and an attribute that defines whether the file is open or not.

Field Name	Datatype	Description
User_Id	Int(11)	User Identifier

File_Id	Int(11)	File identifier
In_use	Int(1)	0: The file is not in use for this user. 1: The file is in use for this user

Table 3: `user_file_info` table structure

4.1.4 Lock Information

The server maintains a `lock_info` table to keep track of locks. Each entry in the `lock_info` table is associated with the user identifier, file identifier, paragraph identifier of the paragraph and timestamp of the lock. The server predefines a few paragraph identifiers that indicate a starting paragraph, an ending paragraph, a locked file, and a file with no locks at present.

A user is allowed to hold only one lock on a document. Whenever the client requests a locked resource, the server checks the timestamp of the user who last acquired the lock. If the difference between the current time and the timestamp of the lock is greater than one hour, then the server grants the lock to the new user. Therefore, it is necessary for a user to save the contents of the editing region periodically so as to update the timestamp, thereby preventing the server from releasing the lock.

Field Name	Datatype	Description
User_Id	Int(11)	User Identifier
File_Id	Int(11)	File identifier
Para_Id	Int(20)	Paragraph identifier of the paragraph
Timestamp	Timestamp	Time at which data was last locked or saved.

Table 4: `lock_info` table structure

4.1.5 Session Information

The server maintains the session information of the user. The server ensures that only one session exists for every user. Each session is associated with the user identifier and the timestamp at which the session was created.

Field Name	Datatype	Description
User_Id	Int(11)	User Identifier
Timestamp	Timestamp	Time at which the session was created.

Table 5: `session_info` table structure

4.1.6 History Information

The server maintains a list of the latest users of all paragraphs of the file in the `history` table. The purpose of this table is to provide information about inconsistent paragraphs to the user. The table structure contains information related to user identifier, file identifier, paragraph identifier and the time at which the paragraph was last saved.

Field Name	Datatype	Description
User_Id	Int(11)	User identifier
File_Id	Int(11)	File identifier
Para_Id	Int(20)	Paragraph identifier of the paragraph
Timestamp	Timestamp	Time at which data was last saved.

Table 6: `history` table structure

4.2 Locking a Paragraph

The locking granularity of the concurrency control mechanism is a paragraph. Finer granularity locking is less constraining to the clients but entails greater overhead [10]. Whenever the user starts editing a paragraph, the server locks the paragraph. In order to decrease the likelihood of the user trying to access a locked paragraph, the client displays locked paragraphs in colors associated with the users who have locked them. Whenever the user tries to modify the contents of an unlocked paragraph, the client sends the MD5 hash of its copy of the paragraph contents to the server to verify that the contents have not changed since the last time the client fetched that paragraph. The server checks whether the paragraph is consistent with the information in the `file_info` table of the database. If so, the server grants the lock; otherwise the server notifies the client, which displays the paragraph in red to show that the local copy is inconsistent with the true copy. In that case, the user may choose to fetch a fresh copy of the file.

The user releases the lock by shifting the cursor from the locked paragraph to a new paragraph. The client then sends the new contents of the modified paragraph so the server can update the `file_data` field of the `file_info` table. When the server releases the

lock, the server modifies an entry in the history table to record the latest user of the modified paragraph.

Users can either explicitly or implicitly release locks. If the user has decided not to edit a particular paragraph, the client can voluntarily release the lock, after which the server deletes the tuple from the `user_file_info` table corresponding to this user. By modifying the content of a paragraph, the user implicitly requests the lock. On shifting from one paragraph to other, the user implicitly releases any lock. Thus, deadlock is never possible, since the user releases any lock before acquiring a new one.

4.3 Opening a document

As the user types in the name of the file to be opened, the client asynchronously acquires from the server and displays those file names whose prefix matches the typed-in word. From the list of filenames, the user can choose a file to open. Then the server fetches the information from the `file_info` table and sets the `in_use` attribute of `user_file_info` table to one for the client. If the user was already modifying some other file, the client prompts the user to save the contents of that file before proceeding with the new one. The server creates an entry in the `lock_info` table with the paragraph identifier set to that special value indicating that the user has opened the file but has not yet acquired any locks. The client retrieves the lock status information from the server and displays it to the user by coloring paragraphs as appropriate.

4.4 Creating a new file

The client sends the server a request to create a new file with the name specified by the user. If a file with that name already exists, the server reports an error. If not, the server creates a new file for the user and adds an entry in the `file_info` table. The server adds an entry in the `user_file_info` table of the database and sets the `in_use` field of the entry to 1.

4.5 Sharing a file

The client sends a request to open a file with the name specified by the user. If the file name matches an existing file, then the file is implicitly shared. The server creates an entry in the `lock_info` table with the paragraph identifier set to that special value indicating that the user has opened the file for reading. The server adds an entry in the `user_file_info` table of the database and sets the `in_use` field of the entry to 1. If the file name does not exist, the server reports an error. If the file is in use by some other users, then the server sends the contents of the file along with the user name and the font color related to other users accessing the file. The client displays each locked paragraph with the font color of the user who has locked that paragraph.

4.6 Saving a file

The client sends the contents of the modified paragraph to the server. The server updates the `file_info` table of the database. The server updates the timestamp field of `lock_info` table of database for that lock.

4.7 Closing a file

The client sends the request to close a file to the server. If the user holds the lock for any paragraph in the file, the client gives a warning about the unsaved contents of the file. The server modifies the entry in the `user_file_info` table of the database. The server sets the `in_use` field of the table entry to 0. The server removes the entry in the `lock_info` table of the database.

4.8 Deleting a file

The client sends a request to the server to delete a file. The server checks the `user_file_info` table for other users for that file. If the client is the only one accessing the file, then the server deletes the entry from `user_file_info` table and deletes the file from the `file_info` table. If not, the server deletes only the entry from the `user_file_info` table.

4.9 Searching for a word in a file

The user may interactively search for a word. The client prompts the user to enter the search word. The client searches the full text of the file for the specified word and highlights the word. The client allows the user to cancel the highlight on search results. The server is not involved in word searches.

4.10 Spell check

When the user requests a spell check, the client forwards the request to the server. The server uses Aspell [17] to spell-check the file and returns a list of misspelled words to the client. The client performs a search for all the misspelled words, which it displays to the user in a highlighted format. Users can right-click on the misspelled word to get suggestions for that word. The client acquires the list of alternative spellings from the server only when the user right-clicks. The client allows the user either to ignore the misspelled word or select one of the suggested words. Any correction applies immediately to all instances of that word.

4.11 Editing a file

A user may edit a file only after acquiring a lock. The client can acquire a lock either on the entire file or on those paragraphs that are not locked by other users. Whenever the user starts editing a paragraph, the client sends the identifier of the paragraph to the server to check whether it is locked or not. If it is not locked, the server grants the lock to the client and the client changes the font color of the paragraph to green to let the user know that the client has acquired a lock on this paragraph. The server updates the paragraph-identifier field of the `lock_info` table with the new paragraph-identifier value.

4.12 Fetching the content update of a file

On user request, the client sends a request to the server to fetch the current contents of the file. Before sending the request, the client ensures that it has sent any local changes to the server and released its lock.

4.13 Plugins

The client adds five plugins in FCKeditor to handle the events performed by the user on the document. They are as follows.

- Key handler: This plugin is responsible for handling key-press events. The client identifies the key strokes of the user on the paragraph, and depending upon the current lock status of the paragraph, the client requests the lock to the server. If the lock is not available, the client prevents the propagation of the key event.
- Lock handler: This plugin handles the “on selection change” event. Whenever the paragraph receives its focus, the client triggers this event. This handler is responsible for releasing the current lock.
- Paragraph delete: This plugin is responsible for merging two paragraphs when the user types either the backspace or the delete key.
- Paragraph enter: This plugin is responsible for splitting a paragraph when the user types the enter key.
- Suggest: This plugin is responsible for retrieving the suggestions for the misspelled word and displaying them in the context menu of the word. When the user selects the suggested word, the plugin replaces the misspelled word with the suggested word.

4.14 Undo and redo operation

- The client allows the user to undo and redo the operations performed within a locked paragraph. When the user releases the lock, the client sends the paragraph contents to the server to update the file contents in the database, and the user cannot undo or redo any operations performed in a released paragraph.

Future work

- AjaxWordPro can be modified to allow the users to lock a part of a paragraph, so that multiple users can work on the same paragraph.
- It can be modified to allow the users to perform the cut, paste, backspace and delete operations across paragraphs.
- It can be modified to allow a user to forcibly release the locks of other users working on the same document.
- It can be enhanced to support some text formatting options such as font editing, bullets, alignment.
- It can be modified to have some interactive mechanisms to register new users.
- It can be enhanced with features for exporting and importing documents.
- It can be modified to have a tree-structured filename space rather than a flat name space.

5. Screen shots

- **Enter username and password and click submit button**

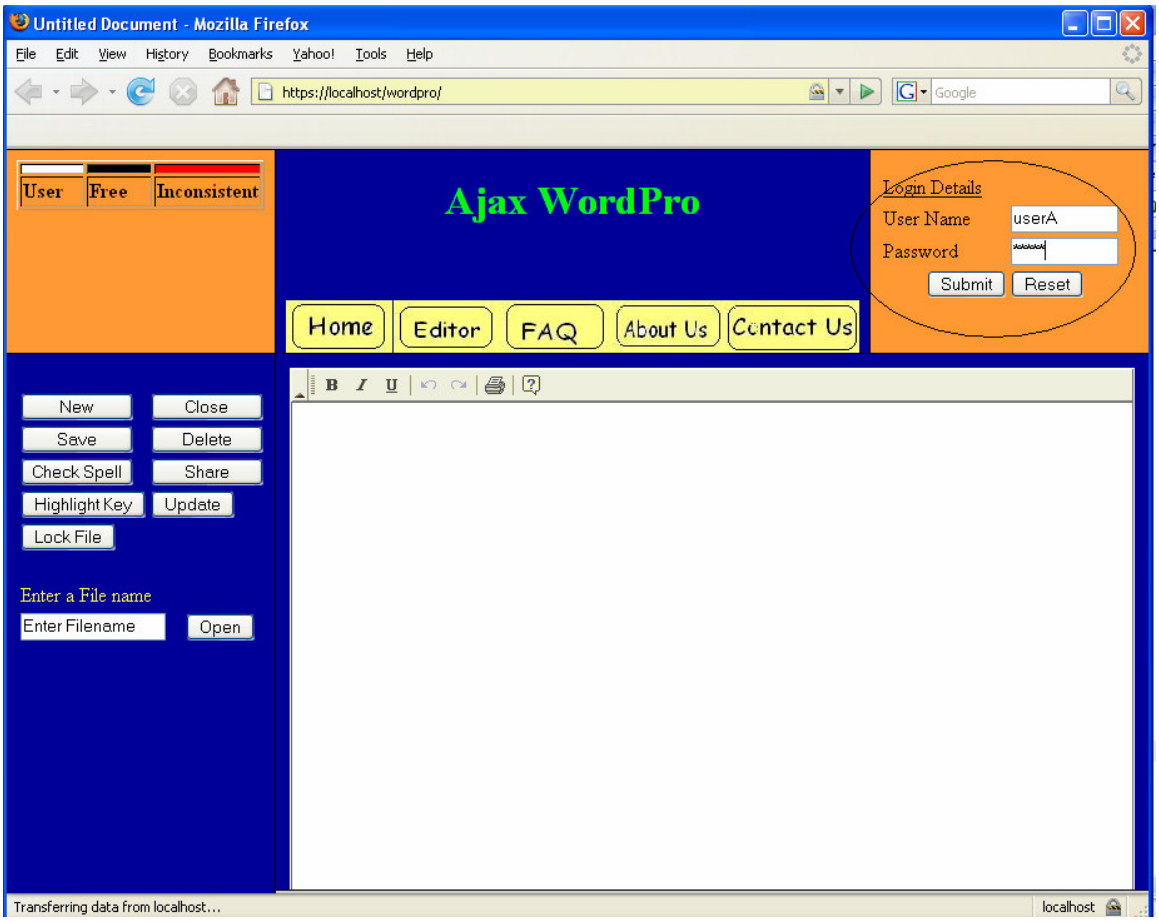


Figure 6: User Authentication

- **Authenticate a user.**

  After the server verifies the user's identity, the server allows the user to perform operations. The status text box displays the current operation performed by the

user (Figure 7). A user can close the session by clicking the logout link at the top of the window.

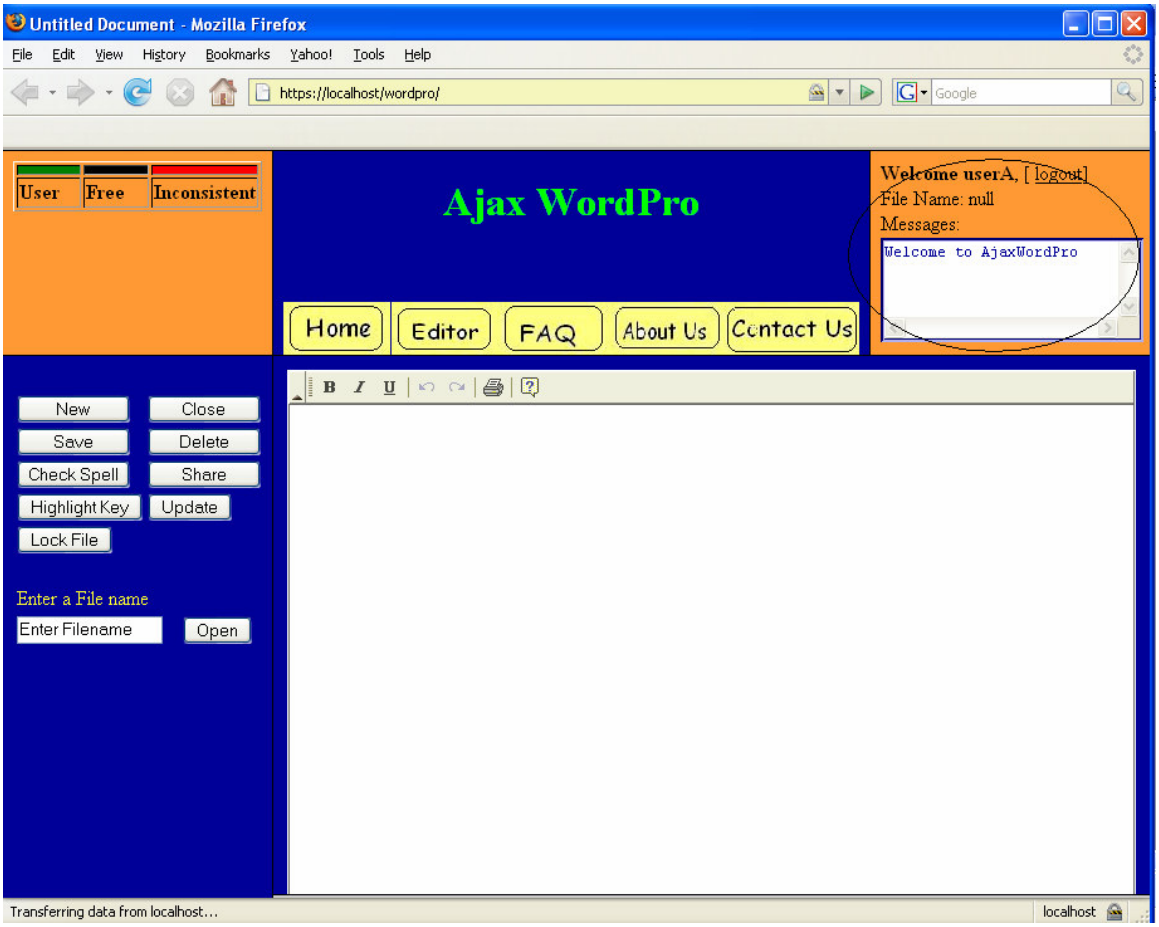


Figure 7: User is authenticated

- **Open an existing file**

  The user types in the name of the file. As the user types in the name, the user sees a list of file names having the matching prefix of the file name (Figure 8). The

user clicks on the required file or types in the file name and clicks the open button to view the contents of the file. The user sees what paragraphs are locked and by whom by referring to the table displayed at the top left corner of the screen (Figure 9).

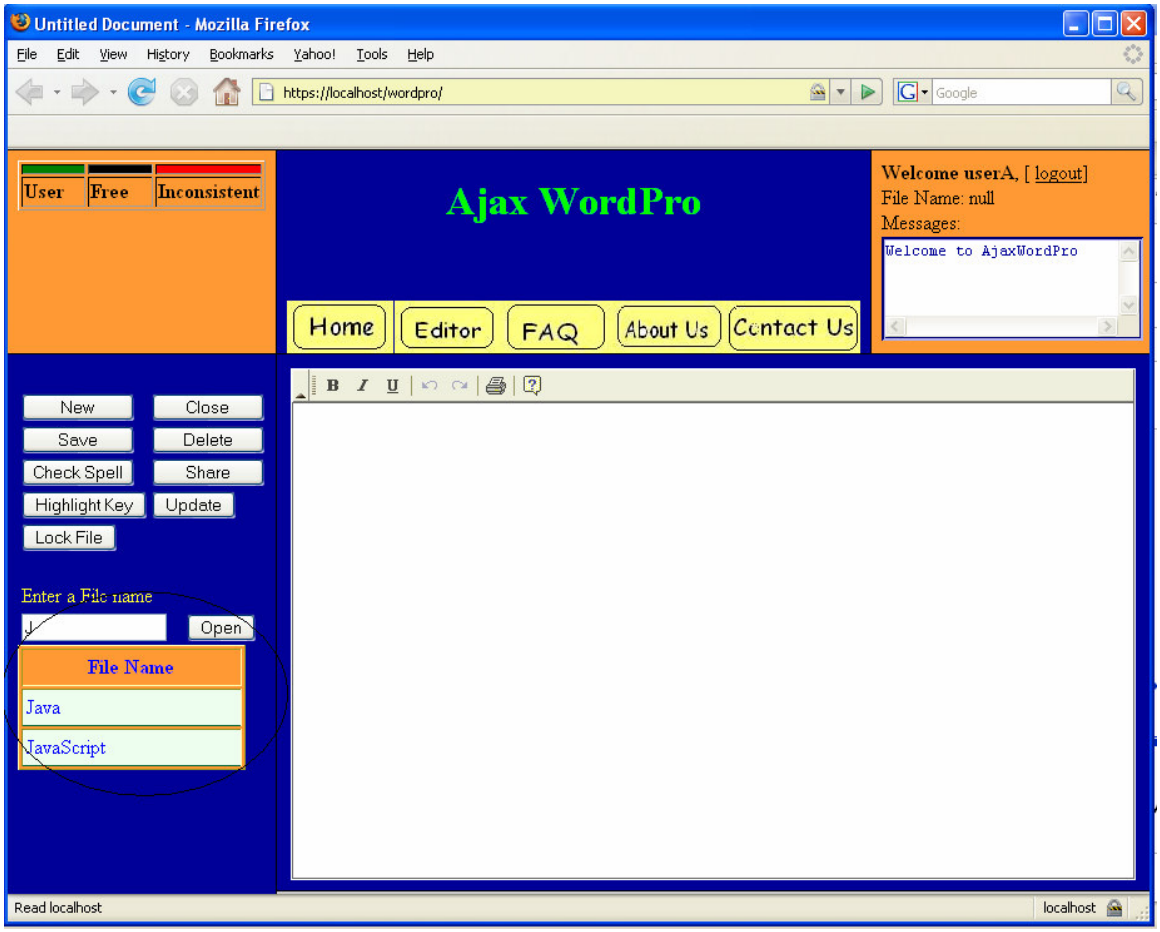


Figure 8: Auto-suggestion of file names

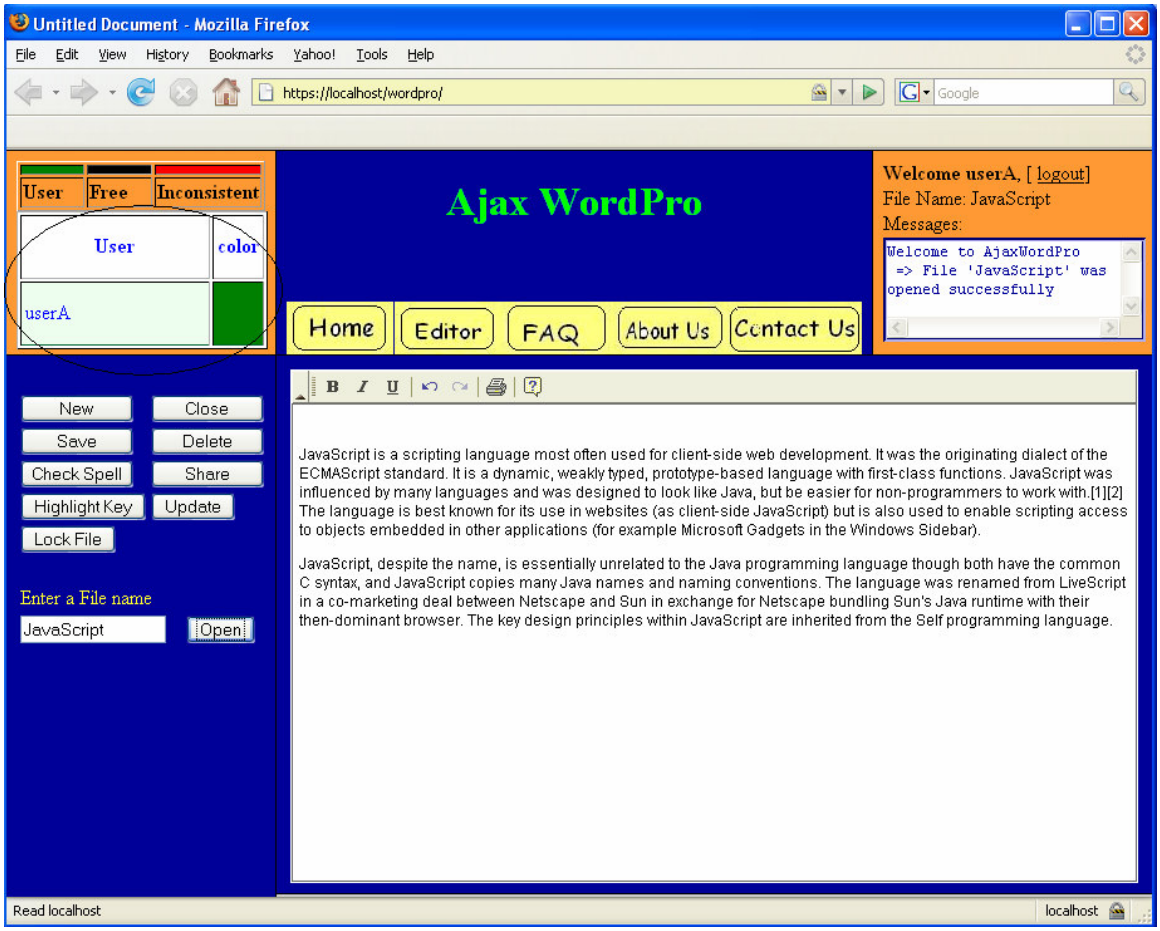


Figure 9: File is opened by the user

- **Lock a paragraph**

The user starts editing the paragraph. The server grants the lock to the user for editing purposes. The font color of the paragraph is green (Figures 10 and 11).

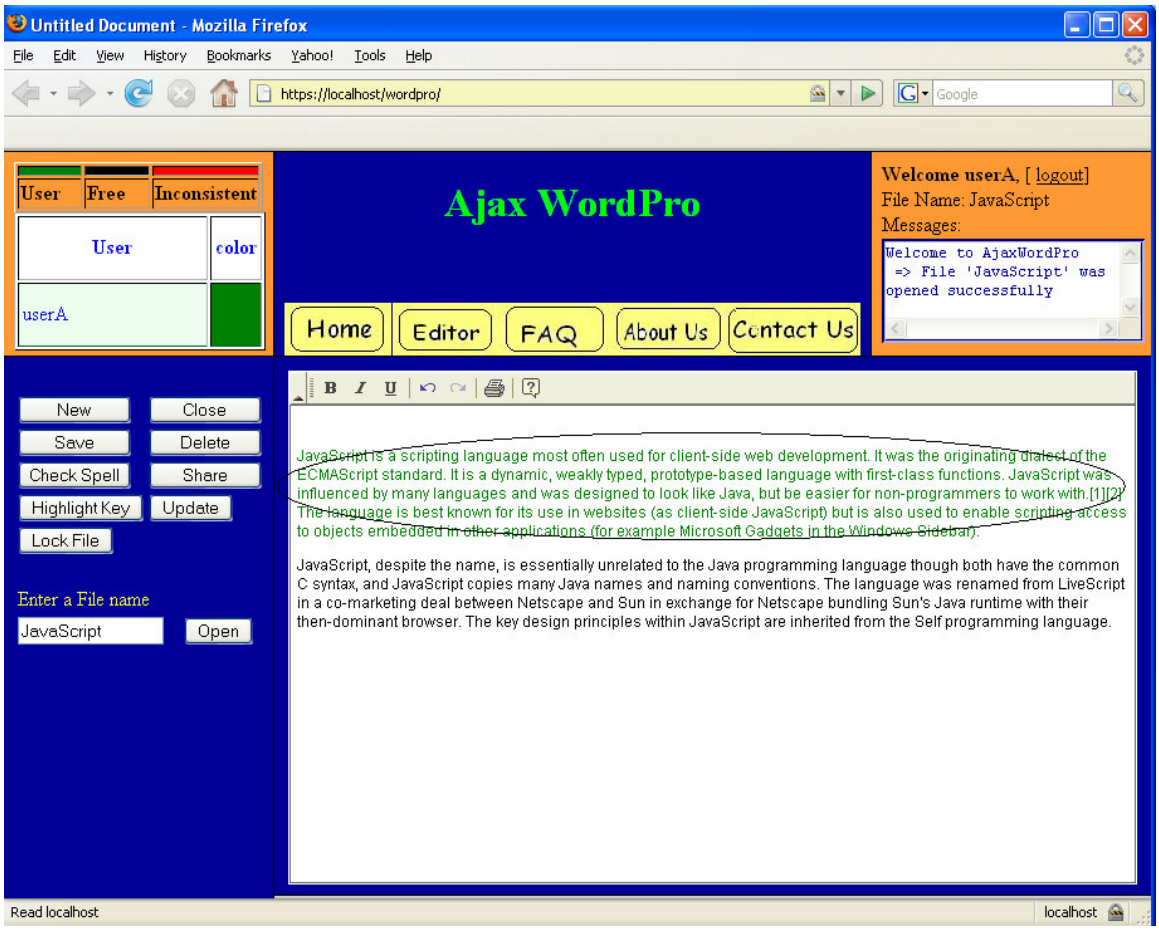


Figure 10: Locked acquired by the user

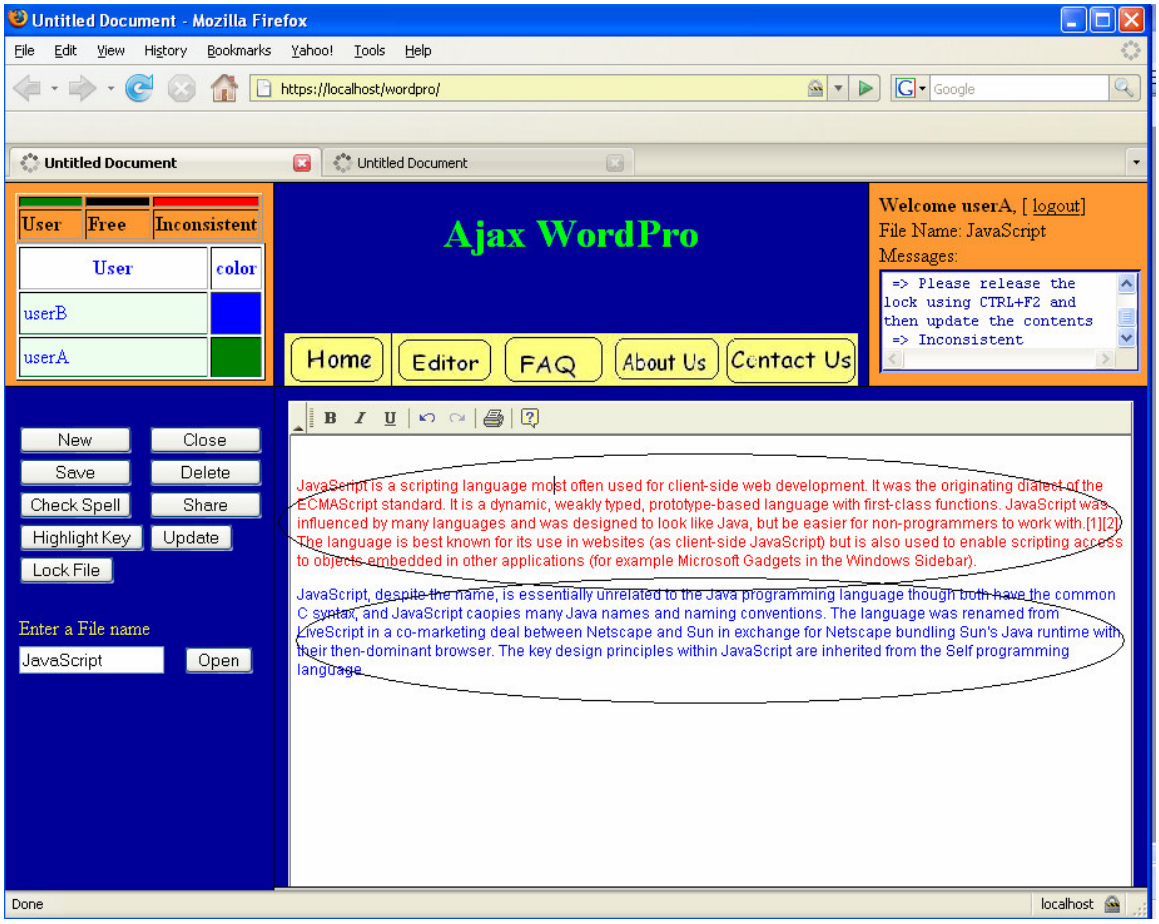


Figure 11: Displays an inconsistent paragraph and another locked paragraph.

- **Save the file**

  To save the file, the user clicks the "save" button. The status box displays the message "File Saved" after successful completion (Figure 12).

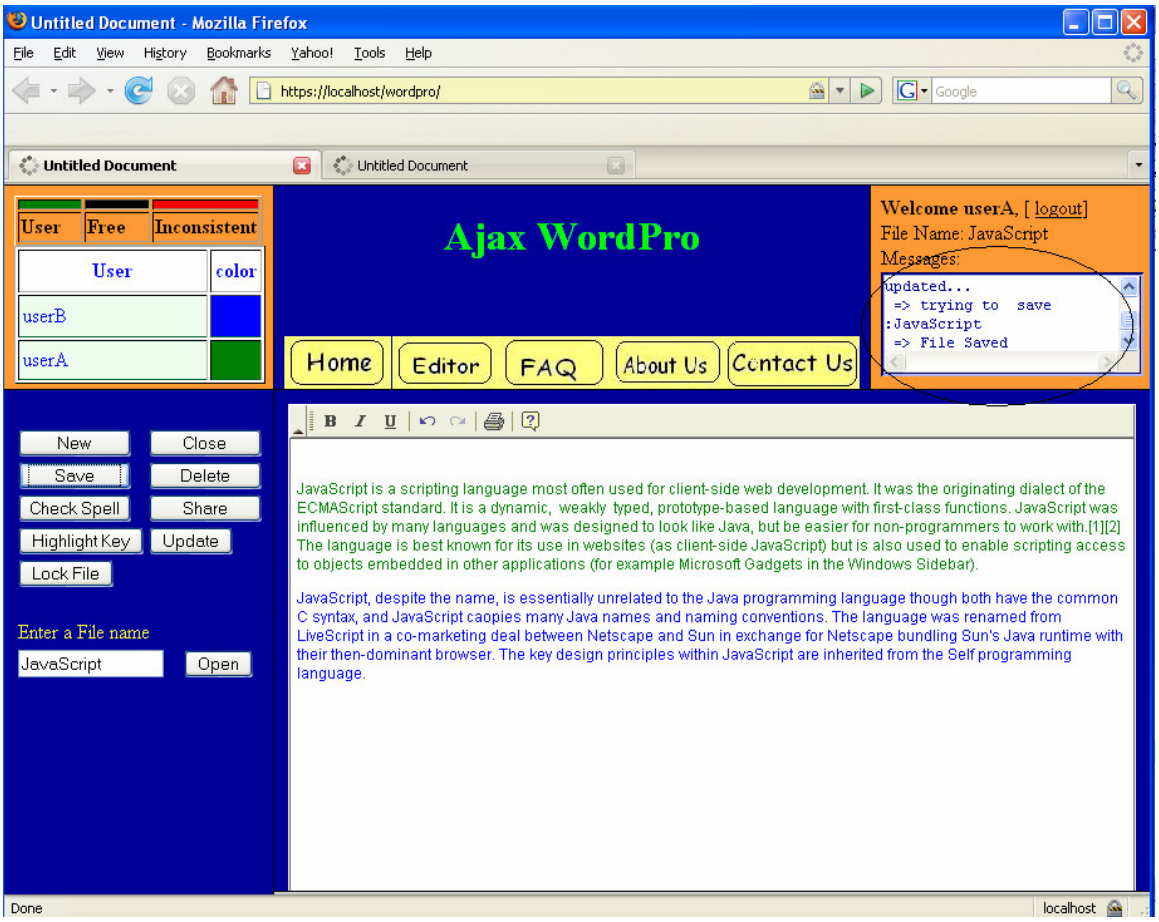


Figure 12: Saving a file

- **Spell check**

The user clicks the “check spell” button to highlight the misspelled words. The user performs a right-click operation on each misspelled word to display a context menu with the list of suggested words or with “No suggestion”. The user can click on the suggested word to replace all the occurrences of the misspelled word or ignore the misspelled word (Figures 13 and 14).

Figure 13: Highlight all the misspelled words

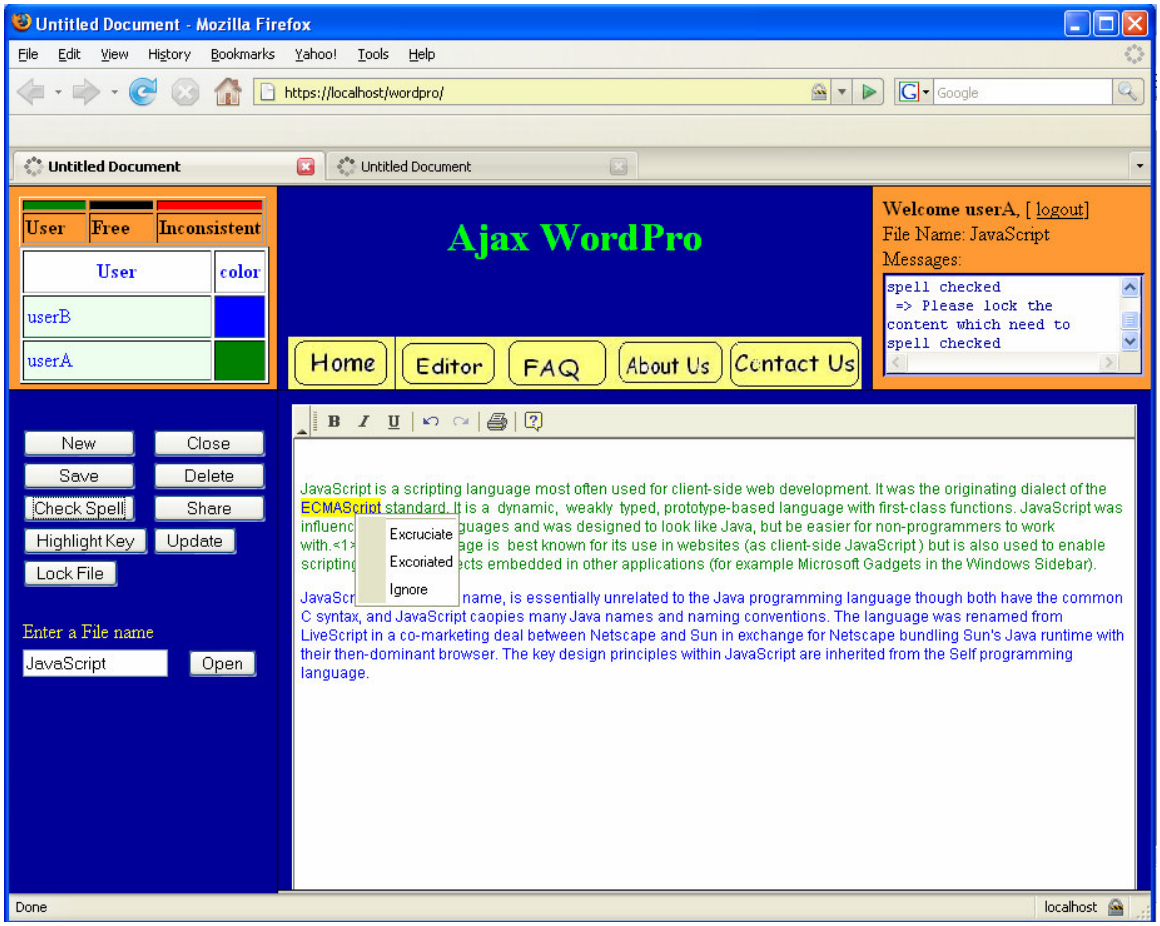


Figure 14: Context menu with suggestions for misspelled word.

- **Highlight a search word**
  The user can highlight a search word by clicking the "Highlight Key" button. The user enters the word to be highlighted. The user removes the highlight feature on words using the cancel highlight button (Figures 15 and 16).

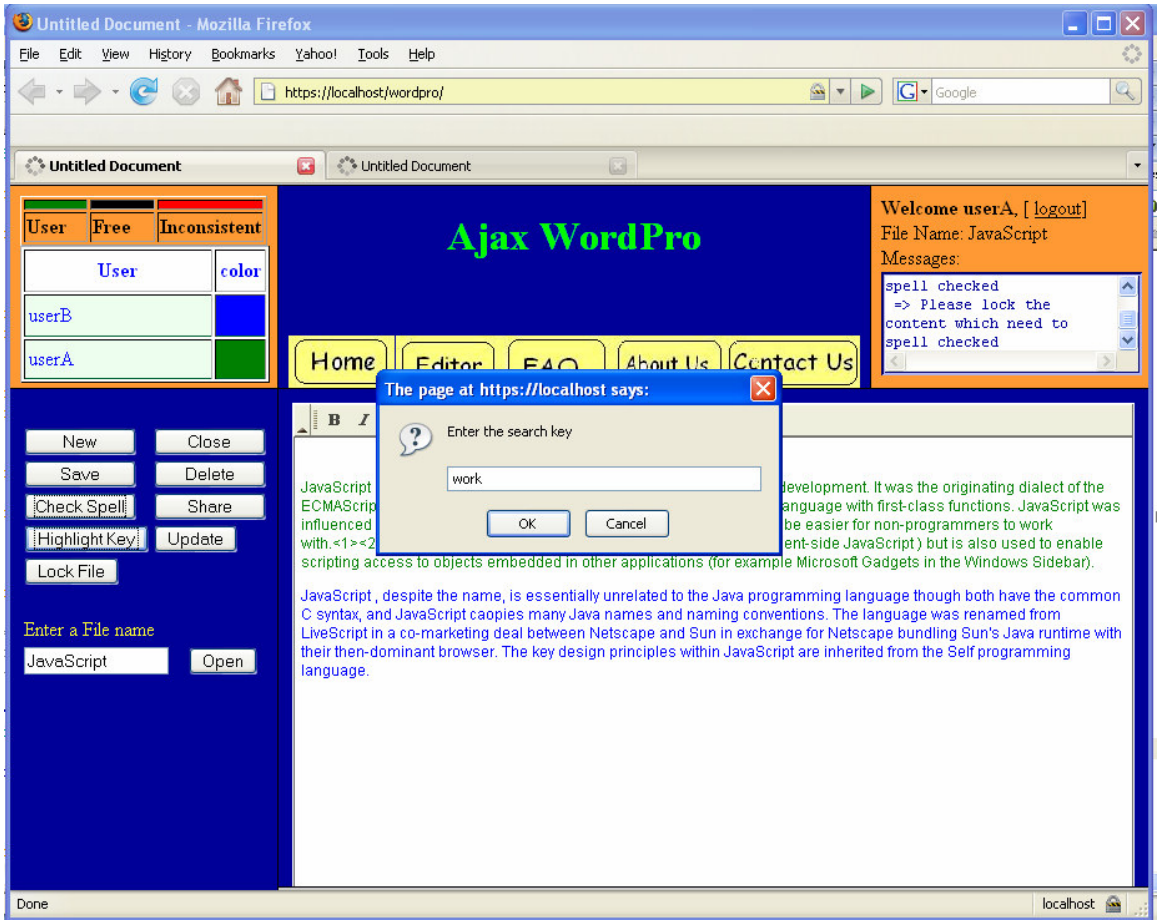


Figure 15: Perform a full-text search and highlight its occurrences

Figure 16: Searched word is highlighted

6. What I learned

- PHP: This project gave me a wonderful opportunity to learn object-oriented server-side scripting.
- This project helped me to understand the concept of concurrency control through locking, which enforces exclusive access rights to a paragraph of the document for editing.
- This project gave me an opportunity to learn about Ajax and how to implement Ajax using the JavaScript `XMLHttpRequest`. From this project, I understood that interactive web applications can be created without refreshing the web browser and by transferring minimal amount of information between the client and the server.
- This project taught me how to generate a good technical report, thereby communicating my ideas and concepts to the users of the document.

7. What was hard to implement?

- Searching a word was hard, if the parts of the word were enclosed by some basic HTML text formatting tags. For example, if I am trying to search the word "computer" it was easy to identify the word which had a single HTML tag like " computer ". But it was hard to find the word with some text formatting tags for some characters of the word like " comput<u>er</u>". In these cases, when a beginning tag was found in the middle of the word, the search is performed for the remaining part of the word.
- Modifying the source code of FCKeditor, so that the user can perform the undo and redo operations within a locked paragraph. By default, FCKeditor stores all the actions performed by the user in the undo array. I created a new function to flush the contents of the undo array in FCKeditor whenever a lock is acquired or released.
- Creating new event handlers to handle enter, backspace and delete key operations on a locked paragraph. The source code of FCKeditor had several event handlers for these events. It was hard to modify the existing event handlers. I created a user-defined event that fires before existing key-event handlers, to decide whether to propagate the key-event or not.
- Preventing the user from performing cut, paste, delete and backspace operations across paragraphs. By default, FCKeditor allows the user to perform these operations across paragraphs. But these events need to be allowed, if and only if both the paragraphs are locked by this user. I created a user-defined event which was fired prior to the execution of existing key event handlers, to decide whether to propagate the key event or not.

8. References

1. S. Xia, D. Sun, C. Sun, D Chen, and H. F. Shen: "Leveraging single-user applications for multi-user collaboration: the CoWord approach," In *Proceedings of ACM 2004 Conference on Computer Supported Cooperative Work,* Nov 6-10, Chicago, IL USA, pp.162-171.

2. Chengzheng Sun, Steven Xia, David Sun, David Chen, Haifeng Shen, Wentong Cai: "Transparent adaptation of single-user applications for multi-user real-time collaboration," *ACM Transactions on Computer-Human Interaction, Vol*. 13, No. 4, December 2006, pp.531-582.

3. Sachin Mullick, Raphael Finkel: "MUSE: A Collaborative Editor", a Master's project done in the Department of Computer Science, University of Kentucky, November 5th, 1998.

4. “FCKeditor Documentation”, http://docs.fckeditor.net/.

5. Jesse James Garrett: “Ajax: A New Approach to Web Applications”, February 2005.

6. Greg Murray: "Asynchronous JavaScript Technology and XML (Ajax) With the Java Platform", June 9, 2005.

7. “FCKeditor, the text editor for an internet”, http://www.fckeditor.net/.

8. J. Schlichter, U. Borghoff, “Concurrency Control for Multiuser Editors”, ACM, 1992.

9. An Operating System Vade Mecum. Second Edition, Raphael A. Finkel, Prentice Hall, 1988. Pages 270-272. ftp://ftp.cs.uky.edu/cs/manuscripts/vade.mecum.2.pdf.

10. C. A. Ellis, S. J. Gibbs, “Concurrency Control in Groupware Systems”, ACM, 1989.

11. “JavaScript”, http://w3schools.com/js, “Ajax”, http://w3schools.com/ajax.

12. By Lee Babin, “ Beginning AJAX with PHP--From Novice to Professional ”, ISBN13: 978-1-59059-667-8, ISBN10: 1-59059-667-6, 272 pp, Published Oct 2006, Apress.

13. Rafael Dohms, “Ajax: What is it?”, October 2006.

14. “JavaScript DOM”, www.w3schools.com.

15. “SSL”, www.webopedia.com/TERM/SSL.html

16. Rob, “XAMPP: SSL Encrypt the transmission of passwords with https”, July 15th, 2007, http://robsnotebook.com/xampp-ssl-encrypt-passwords.

17. “GNU Aspell”, http://aspell.net/.

18. Yet Another PKI: “How to - private PKI system just with OpenSSL”, http://www.nissle.ch/ssl/PKI-OpenSSL.pdf.

19. Paul Johnston, Greg Holt, Andrew Kepert, Ydnar, Lostinet, “MD5 Message Digest Algorithm Implementation”, version 2.1, 1999-2002, http://pajhome.org.uk/crypt/md5/md5src.html.