I developed my own mini-project in .NET, inspired from a project suggestion on sciencebuddies.org. The application was written in C# and it communicates with a remote MySQL database via PHP scripts. The MySQL database is stored on the web space the University provides all engineering students with.

In a nutshell, the application allows one to upload an essay (title and author) stored on a text file and retrieve statistics on word frequencies, punctuation usage, sentence count, word count, and character count. The data is then store in a remote database, and it can be retrieved whenever there is an internet connection.

Given more time, I would have filled the database with a large number of common authors and their works, and I would write an algorithm to determine the author of a new essay automatically by comparing the statistics.

**How to use it:**

To upload an essay, select ‘browse’ and open a text file containing the essay. Click ‘Read’; this will upload the data to the two DataGridViews on the right of the window. If you want the data to be uploaded to the database, check the ‘Upload to Database?’ checkbox and click ‘Read’.

To view data in the database, click the ‘connect’ button. This will update the ‘author’ drop down box with a list of authors in the database. After selecting an author, the ‘title’ drop down box will include all of the author’s essays in the database. Selecting a title then uploads all the essay data to the DataGridViews.

**How it works:**

The application reads all the text from the text file and breaks it into a space and dash delimited list, a list of words. The application then loops through each word in the list, updating a hash table of word frequencies, with each word string being the key. The word is also passed to a checkPunctuation function, which checks for occurrences of punctuation to update the essay statistics, and if a period, exclamation, or question mark, it determines if the word is the end of the sentence. The program attempts to discount abbreviations by using two assumptions: the word is the last in a sentence if the next work is capitalized, and it also compares the word an array of titles (because titles may be followed by capitalized names). This algorithm runs in O(n) time, and 1000+ word essays were read and had their data updated on a DataGridView in an imperceptibly short period of time.

If the user checked the option to also upload the data to the database, the program will then send some insert queries to the database via a PHP script. The program uploads all data to the database, a ~1500 word essay took about a minute to upload, although it varies with connection speed.

The program can read results of queries to the database and upload them to the DataGridViews. This is done by sending a query to a PHP script, and reading the result echoed. The results are in JSON format and the C# program performs a couple of string manipulations to extract the tuples returned and upload them to the data grid.