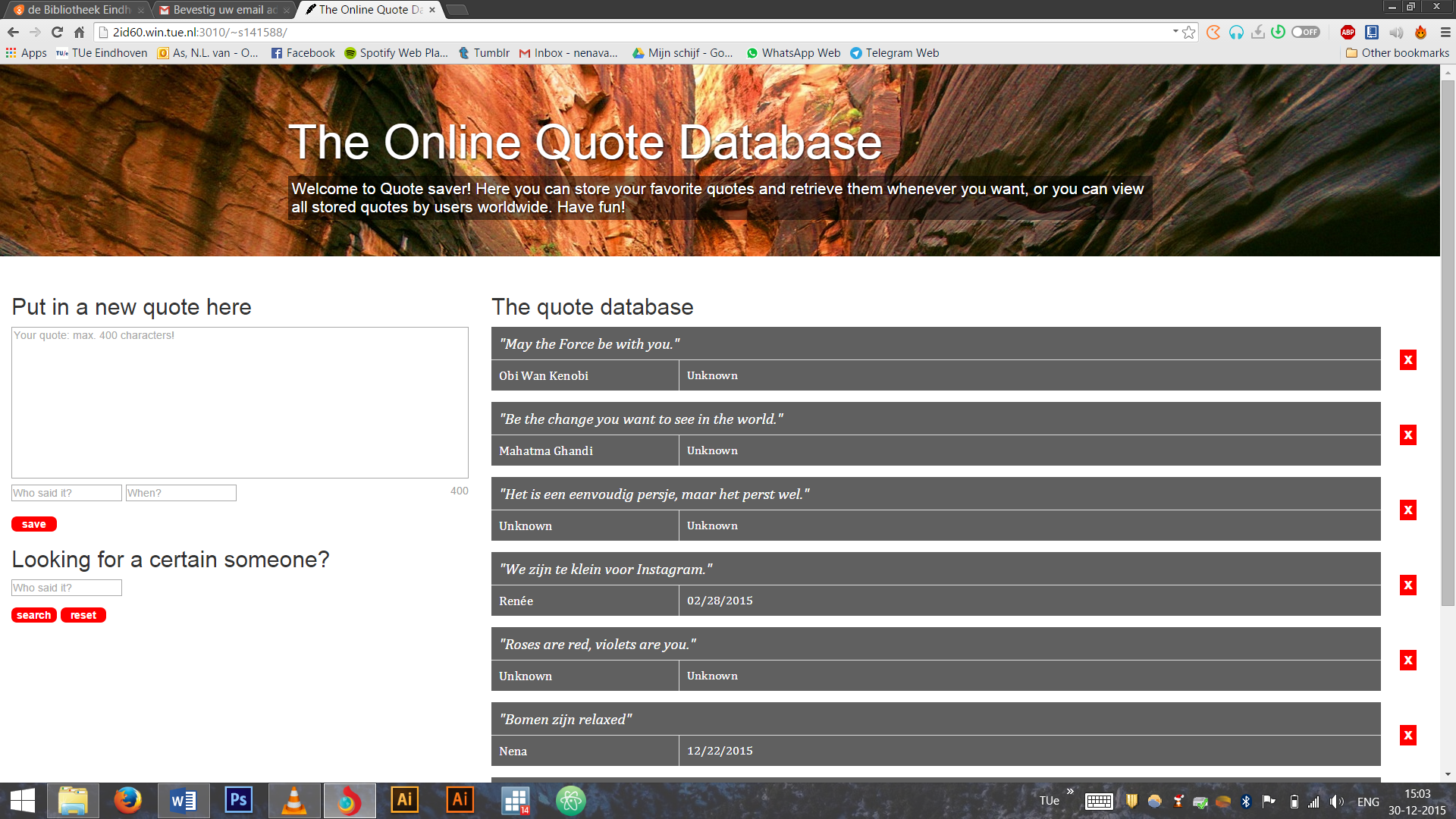
The Quote Database

# General

My web app is a place to store quotes overheard in public/read somewhere/remembered so that you can access them later. It is a universal app that can be accessed and adapted by everyone. The app can be accessed from <http://2id60.win.tue.nl:3010/~s141588/>   
I tested the app both in Torch, which is a Chrome-based browser, and Firefox.

# Design

I made use of Node.js and Express for a quick-and-easy app set-up. I chose this framework because it was the easiest for me to understand and use properly, which fits the purpose of getting my app to actually work ☺ Other components I used are Bootstrap for layout purposes and jQuery for interaction. Below you can find a picture that displays the general view of the app.



Most layout/visual adaptations were made using CSS. The site contains a jumbotron element, and two col-md’s of width 4 and 8 (one for the quote input & search fields, and one for the quote output). The entire page is fully responsive to changing screen sizes.

# Databases

I store quote information in a MongoDB database which is hosted externally. I store my quotes as an array filled with JSON objects (one per quote), similar to the one below:

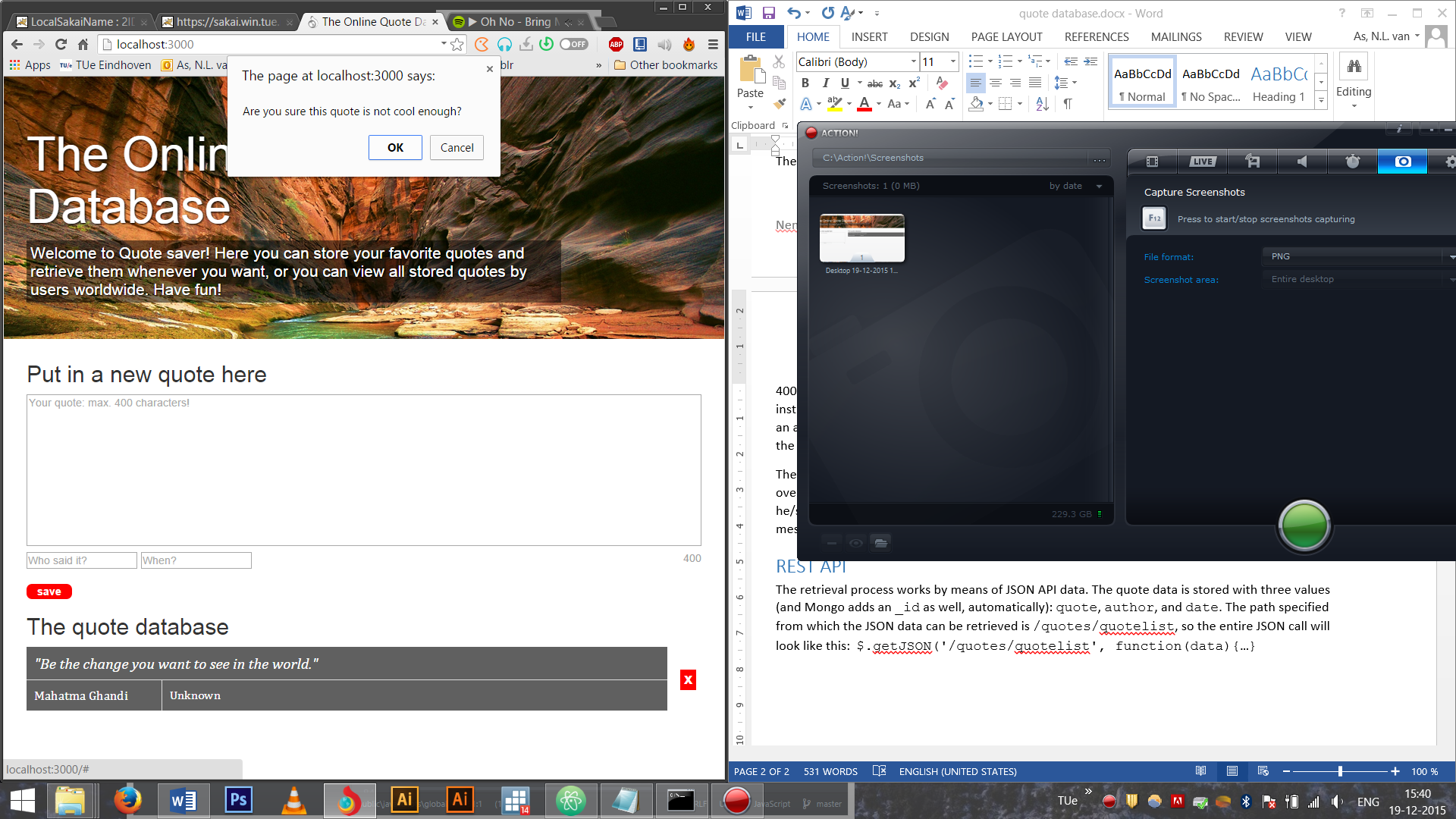
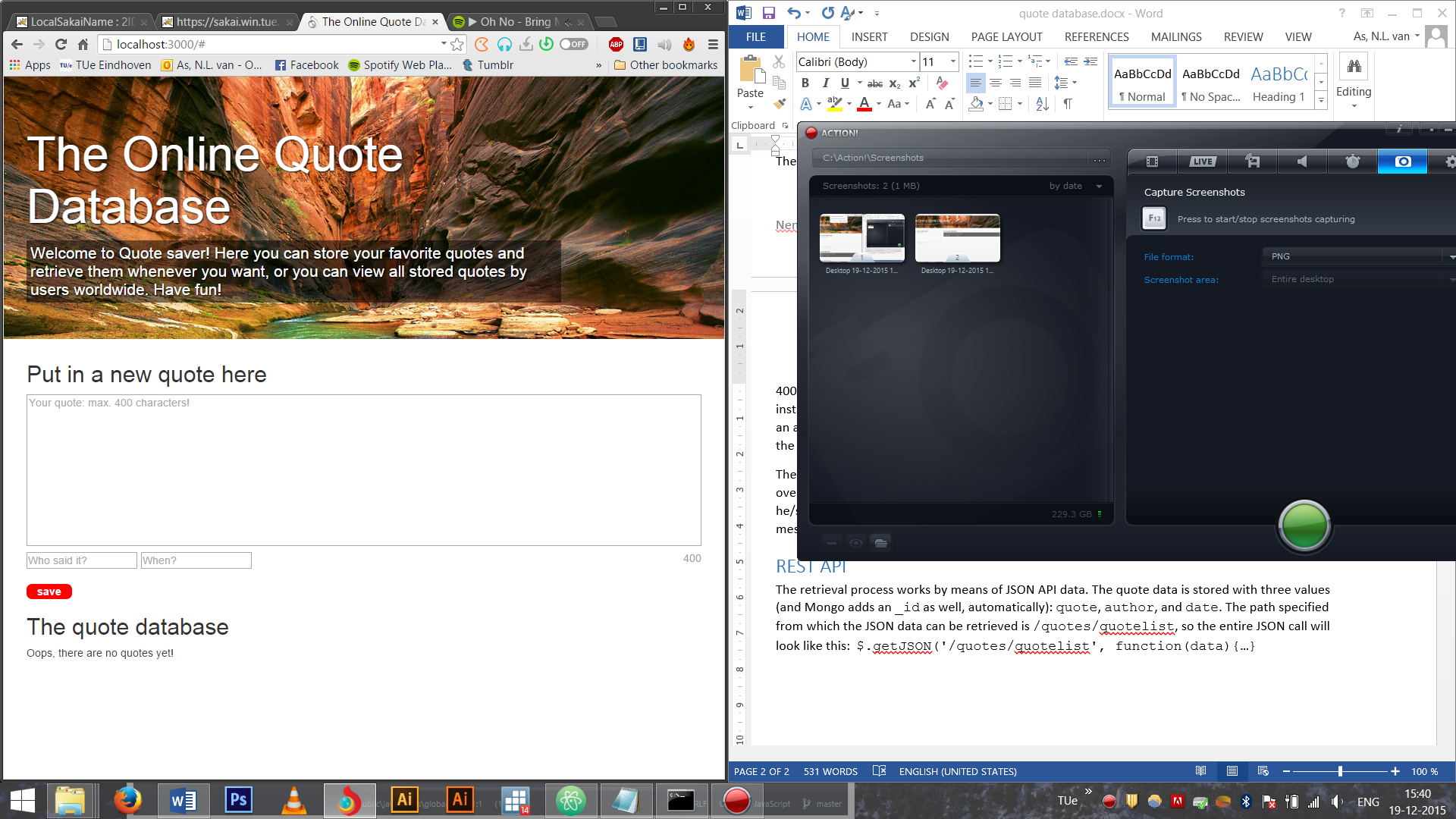
{"quote":"Insanity: doing the same thing over and over and expecting different results.","author":"Albert Einstein","date":"Unknown"}

# Interaction

I used JQuery stored in a global.js file, under ./public/javascripts to call different functions when the page is ready: showQuotes, addQuote, deleteQuote, searchAuthor and trackChar (which is an extra function to keep track of the amount of characters in the text area). Also in the $(document).ready(function){…} are a couple of event handlers to define what needs to happen when certain buttons are clicked.

The user can add a quote to the database by filling it in in the areas on the left side of the screen. The text-area is a fixed container (so no user-based tinkering in size here) with a character limit of 400. A quote can be stored without author and date specified, which will generate two “Unknown”-s instead automatically. However, when an attempt is made to post an empty quote, the user receives an alert stating: “please fill in a quote!”. I also attached a spellchecker function to the text area to aid the user in putting in correct spelling. To make sure that the date format would be the same for all quotes, I attach a date-picker function to the “When?” input field – similar to the ones being used on ticket purchasing websites. This date picker does not allow users to put in a future date: only dates up until today can be filled in.

The user can also delete quotes from the database itself by clicking the red square with the white “x” over it. To make sure that it was no accidental click, an alert pops up first asking the user if he/she is sure that the quote is not cool enough. If there are no more quotes left after deleting, a message is shown saying that there are no quotes yet. The pictures below illustrate this output.



The last function to discuss is the one that allows the user to search for a certain author. It uses the user-specified input, capitalizes it to avoid upper- and lowercase problems with validation, and checks this input against all the “author” fields in the JSON object. If the requested author is the same as the quote author, that quote is generated in the output screen. To see all quotes again instead of the searched filter, the user can click the reset button.

I originally intended to also include some sort of input-check function for the specified quotes, such as whether it was actually text and not mumbo-jumbo, but soon found out that if I wanted to develop a function that would actually work, that would cost me more time than the rest of the application combined. I then decided that it would be a nice future improvement, but not necessary for the app itself.

# REST API

The retrieval process works by means of JSON API data. The quote data is stored with three values (and Mongo adds an \_id as well, automatically): quote, author, and date. The path specified from which the JSON data can be retrieved is <http://2id60.win.tue.nl:3010/quotes/quotelist>. The JSON call in my code looks like this: $.getJSON('/quotes/quotelist', function(data){…}).