

**WEB-PALVELINOHJELMOINTI**

**FINAL PROJECT**

**Technical documentation**

**Individual work.**

**Ta Tuan Anh**

*([anh.ta@helsinki.fi](mailto:anh.ta@helsinki.fi) - student number: 014687765)*

**UNIVERSITY OF HELSINKI**  
**Department of Computer Science**

**Helsinki, December 2016**

## **1. Table of Contents**

<b>2. INTRODUCTION .....</b>	<b>3</b>
<b>3. PROJECT SPECIFICATIONS.....</b>	<b>4</b>
<b>3.1. Technical descriptions for the project:.....</b>	<b>4</b>
<b>3.2. Functional descriptions for the first version:.....</b>	<b>6</b>
<b>3.3. User interface .....</b>	<b>7</b>
<b>3.4. Models design: .....</b>	<b>8</b>
<b>3.5. Algorithm for the program.....</b>	<b>9</b>

## **2. INTRODUCTION**

In the Information Technology era, typing skill is one of the most essential skills that help people to get familiar with computers and the Internet. Besides, the need of learning new language is always the high priority of people all over the world, those who want to be more internationalized. However, one of the most challenging things about learning a new language is learning new words.

From that problem, and with the experience of using some typing applications, I started to have an idea, which is the combination of the typing test application and new words learning.

The application not only tests the typing speed but also has a database for authenticated user, to help them type their own words which they are allowed to input into the database. With that improvement, the application is believed to help the users to memorize the vocabularies easily, instead of just writing by hand on papers.

Overall, “Type your words” application will help the customers to both improve typing skill and enhance language learning at the same time. It is undoubtedly more interesting than just doing either of them.

### 3. PROJECT SPECIFICATIONS

#### 3.1. Technical descriptions for the project:

The application is developed in Java for the server side scripting language, PostgreSQL for the database, HTML5, CSS, JavaScript and Twitter Bootstrap & JQuery library for the client side.

The servers are hosted on **Heroku**. There are 2 separate servers: production server and development server.

- Production server: <http://typeyourwords.herokuapp.com/>
- Development server: <https://typeyourwords-dev.herokuapp.com/>

The source code is hosted on **Github**:

[https://github.com/tuananh1993/type\\_your\\_words](https://github.com/tuananh1993/type_your_words)

There are 2 branches:

- Master branch for production server
- Dev branch for development server

Continuous integration testing by **Jenkins**:

[http://jenkins.tatuananh.com/job/type\\_your\\_words/](http://jenkins.tatuananh.com/job/type_your_words/)

Credentials:

User: demo

Pass: demo

#### Deployment process:

- The project is developed locally by the developer(s). After the functionality added successfully, developer will commit the code to “dev” branch on Github.
- **Github** is configured in such a way that when it receives a new commit, it will “hook” the continuous integration server (in this case, it is Jenkins) by the given url configured in Webhook setting.
- A new job is created on **Jenkins** server with the name *type\_your\_words*. The configuration is set by the following settings:
  - o Source Code Management:  
[https://github.com/tuananh1993/type\\_your\\_words.git/](https://github.com/tuananh1993/type_your_words.git/)
  - o Build trigger:  
*Build when a change is pushed to GitHub*
  - o Post-build actions:  
Send email for every unstable build  
**Set Github commit status (When the build processes are passed (including testing, etc.), status is set to success)**

- **Heroku** server is set by the following settings:
  - **Development server:**
    - Deployment method: Github, dev branch
    - Automatic deploy when the commit status is set as success (set by Jenkins)
  - **Production server:**
    - Deployment method: Github, master branch
    - Manual deploy (For a better administration, make sure that only the production admin has the possibility to deploy the app)

### **3.2. Functional descriptions for the first version:**

All users can use the application to test their typing speed; with the given time is 60 seconds. The given words are randomly chosen from the database, which stores thousands of vocabularies. However, the login users will have another privilege. They are allowed to input their own words, no matter what the language those words are. Therefore, they can basically choose either the default words from the system or their own words for the typing test.

To be more interesting, the application will divide the words into different levels, for example:

- Level 1: the most 200 common words
- Level 2: the most 400 common words
- Level 3: the most 1000 common words

(At the moment, due to the lack of development time, the application will only return the first created 200, 500, or 1000 words).

Last but not least, there is a statistic table given to users, so that they can check their last 10 performances.

### 3.3. User interface

#### UI Wire frame:

The application has several pages. At the beginning, I am using moqups to sketch the wireframe before coding it. The sketch of the main page is given below:

##### a. Anonymous user:

TYPING YOUR WORDS

293 × 57

☒ Top 200 words  
☐ Top 400 words  
☐ Top 1000 words

WORDS SAMPLE APPEAR HERE!!!

TYPE HERE

REFRESH

Register

Login

TABLE OF HONOURS

▼ USERS	▼ SPEED	▼ TIME
User 1	70	03/05/2015
User 2	60	03/05/2015
User 3	50	03/05/2015

##### b. Authorized user:

TYPING YOUR WORDS

293 × 57

☐ Top 200 words  
☐ Top 400 words  
☐ Top 1000 words  
☒ USE MY WORD


WORDS SAMPLE APPEAR HERE!!!

TYPE HERE

REFRESH

Add new word(s)

Log out

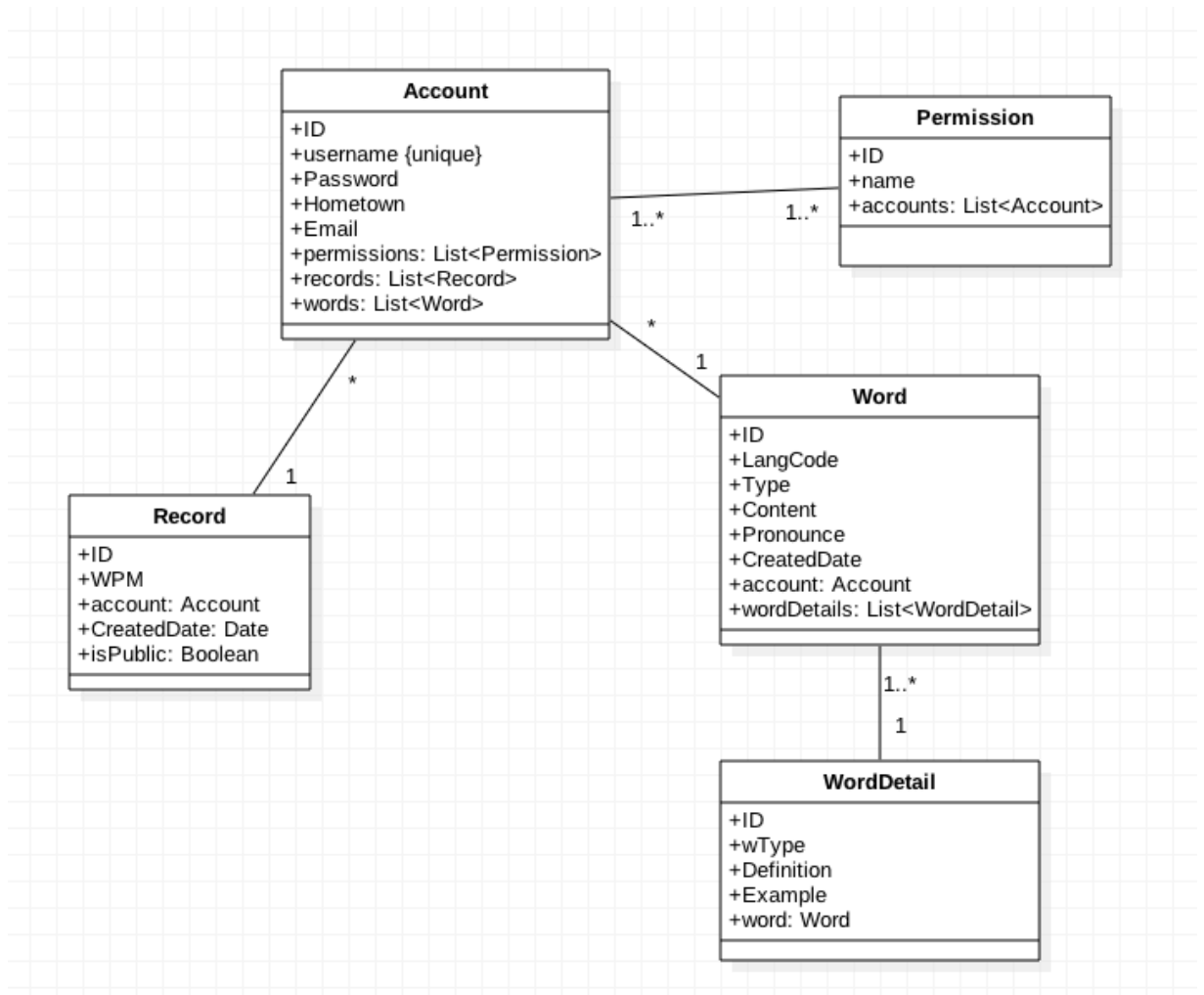


Your last 10 performances

▼ Speed	▼ Time
45	5/3/2015
42	5/3/2015
54	5/3/2015
64	5/3/2015
67	5/3/2015
86	5/3/2015

[Learn how to type with 10 fingers](#)

### 3.4. Models design:





### 3.5. Algorithm for the program

#### 1. Counting words

The algorithm for the program is mainly applied for the calculation of WPM (words per minute). The program will count the number of corrected words typed by users in a given time (60 seconds). By triggering the space button, the program will recognize the input. The algorithm is written in Javascript and JQuery.

#### 2. Request words from the database

The program will check if the user is authorized. If the user is not authorized (anonymous), the random words are generated from the default words. Otherwise, if the user has input their own words, they can choose either the default words or theirs.

#### 3. Sample snippet code

Some of the snippets written by our team developers are given below by using the above algorithm and idea (JQuery code):

##### a. Start counting words

```
function startPrg() {
    $('#typed').keydown(function (evt) {
        // total of input characters
        numChar++;
        $('#result3').val(numChar);

        if (evt.which === 32 && stop === false) {
            // total of words user has typed
            totalWords++;
            // update info
            $('#result2').val(totalWords);
            $('#result1').val(correctWords);

            // assign the class for the current index
            word = "#word" + crtIndex;
            $('#'+word).removeClass("btn-primary");
            $('#'+word).addClass("btn-default disabled");

            // check the condition if the user type right
            if ($('#typed').val() === inpMsg[crtIndex]) {
                correctWords++;
                $('#result1').val(correctWords);

                // paint the color for the correct word
```

```

        //$(word).removeClass("current");
        $(word).addClass("btn-success disabled");
    }
    else
        // paint the color for the wrong word
        $(word).addClass("btn-danger disabled");

    }
    })

    $('#typed').keyup(function (evt) {
        if (evt.which === 32 && stop === false) {
            $('#typed').val("");
            if (crtIndex < inpMsg.length - 1) {
                crtIndex = crtIndex + 1;
                word = "#word" + crtIndex;
                // highlight the current word
                $(word).addClass("btn-primary disabled");
            }
            else
                // if the text sample is exceeded, renew
                genMsg();
        }
    });
}

```

#### **b. Stop counting words**

```

function startPrg() {
    $('#typed').keydown(function (evt) {
        // total of input characters
        numChar++;
        $('#result3').val(numChar);

        if (evt.which === 32 && stop === false) {
            // total of words user has typed
            totalWords++;
            // update info
            $('#result2').val(totalWords);
            $('#result1').val(correctWords);

            // assign the class for the current index
            word = "#word" + crtIndex;
            //$(word).removeClass("btn-primary");

```

```

$(word).addClass("btn-default disabled");

// check the condition if the user type right
if ($('#typed').val() === inpMsg[crtIndex]) {
    correctWords++;
    $('#result1').val(correctWords);

    // paint the color for the correct word
    //$ (word).removeClass("current");
    $(word).addClass("btn-success disabled");
}
else
    // paint the color for the wrong word
    $(word).addClass("btn-danger disabled");

}
})

$('#typed').keyup(function (evt) {
    if (evt.which === 32 && stop === false) {
        $('#typed').val("");
        if (crtIndex < inpMsg.length - 1) {
            crtIndex = crtIndex + 1;
            word = "#word" + crtIndex;
            // highlight the current word
            $(word).addClass("btn-primary disabled");
        }
        else
            // if the text sample is exceeded, renew
            genMsg();
    }
});
}

```