***ask-rep***

***(on-line repository)***

Ventsislav Polimenov (1457989)

Dhiraj Narwani (1431555)

Cloud Computing COMSM0010

Contents:

1. Overview
2. Cloud Computing
3. Planning
4. Implementation

* Log In
* Database functionality
* Front-end design and implementation
* Customizing repositories
* Create repository
* Create/Upload Files
* Custom Search Engine
* Search functionality
* Trending repositories

1. Results
2. Future work
3. Summary
4. Overview

The main objective for this project is to create an online repository where users will be able to create, upload and share their files amongst other users. Users will be able to create public repositories in order to upload or create new files and store them in folders/sub-folders. The main reason for creating custom files is to allow the user to write pieces of code and if needs be whilst typing the application will recommend online code snippets to him/her. This will permit the user to search for code through the application instead of typing their query in Google. Furthermore, these files will be saved on the cloud and placed in new or existing repositories according to the users’ preference. The repositories will be available to the public and the files can be viewed straight from the browser.

1. Cloud Computing

* this section must be at least a page long
* what is Cloud
* how it works
* benefits related to our project
* safe and reliable
* spread out – avoid latency and allow many users

1. Planning

The project took us roughly two months to finish it and tasks were divided as follows:

**Dhiraj Narwani**

* Front-end design and implementation (HTML, CSS, JavaScript and Java code).
* Database Creation – All files, repositories, folders, users will be stored in a database.
* Sign in/out functionality – Users can login with their Google account to access restricted functionalities.
* Customizing repositories – Creating a repository where folders and sub-folders can be included.
* Upload files – Uploading existing code files with specific extensions to the database.
* Viewing files – View existing files from public repositories in the browser.

**Ventsislav Polimenov**

* Google Custom Search – Functionality that will query google in order to extract code tags.
* Extract code tags – Extracting code snippets through ‘code tags’ from pre-defined websites.
* Display code – Displaying code snippets in panels.
* Creating new files – Saving files as BLOBS in the database.
* Trending Repositories – List of all repositories in descending order (recent first).

1. Implementation

* a bit of overview of the implementation methods as we did for the abstract – environment and libraries

*Sub-sections as follows:*

* Log In
* Database functionality
* Trending repositories
* Front-end design and implementation
* Customizing repositories
* Create repository
* Create/Upload Files
* Search functionality
* Hot-Button for quick search

In order to make the user's experience with our Cloud Application, we decided to implement a hot-button for the search functionality. Once the right-arrow is pressed, the program invokes a Custom Search Engine (CSE) and send a query to Google. We also decided to provide the user with the choice of language – Java, Python, C++, and C#. At the end the query that is sent to Google through our CSE consists of the user's highlighted phrase and chosen language followed by the name of the website. However, currently the name of the website is hard-coded to be Stack Over Flow, as it is probably the best for any kind of code that one could possibly be interested at.

* Custom Search Engine

This is a function provided by Google, so that developers can integrate customizable search engines in each and every website and also in online applications. In order to integrate this to our Cloud Application, we used the search engine' URL (placed at “googleapis.com/customsearch/”), the cx\_key, which serves as an authenticator for the engine, the API\_KEY provided by google, and JavaScript Object Notation (JSON) output format.

* Jsoup lib

When user's query is sent to Google, our App gets all results as links to web pages and stores them all to a certain data structure (ArrayList<String>) allocated on the Cloud. Once this process is done, the same data structure is sent to an iterator, which opens each link in order to access the html code for every web page. The iterator seeks for “<code>” tags in the “html”, when found it extracts the text in between and add it to another data structure (HashMap<String, Element>). However, for this functionality we used an external library for html stripping called Jsoup.

* Code snippets rating

In order to get the best possible result for the user and satisfy his needs, we implemented a rating method, which rates the extracted code snippets before they actually get displayed for the user. The rating method works as follows. Firstly, we have some predefined language code tags that serve as language identifiers. If the language of the extracted snippet does not match the language chosen by the user, snippet is getting penalized. However, if a key word occur more than once the rating increases by 5 points.

* Displaying code snippets

For displaying the snippets we decided to use an “accordion” type of container that manages a list of sections of widgets, each with a header [<http://www.smartclient.com/smartgwt/javadoc/com/smartgwt/client/widgets/layout/SectionStack.html>], provided by Google Web-Tool. We gather a minimum of twenty code snippets and display next to the text box assigned for coding.

1. Results

* end product functionality
* our satisfaction with this project

1. Future work

* language processing for compilation
* We could integrate a panel, which detects tokens and add colours. Also run and debug functionalities and turn it into an on-line IDE.
* rating improvements and language classification
* So that we use more sophisticated method for rating, based on word occurrences again, but with better performance for better results. Also, currently we have hard coded language code tags and in the future we could use self learning techniques so that the application itself can improve on understanding the language on its own.

1. Summary

Appendix

References

* everything that goes in [] must be included here in this section.

Gantt Chart