Search Engine for CSE Website

User Manual

April 30, 2021

1 User Manual

1.1 Installation Guide

1.1.1 Installation Environment

We recommend running the application in an UNIX-like environment. It is not required but some of the scripts may not be able to run outside UNIX-like environment.

The project is well-tested in the following environment:

- CentOS 7 (Cloud VM environment)
- Arch Linux
- MacOS Big Sur

Although theoretically this project should have no problem running in Windows, it is not tested and not guaranteed if it would work.

1.1.2 Types of Installation

There are several setups for deploying the search engine stack. Choose the one that suits your needs and constraints.

- Hassle-free Deployment with Docker (HDD) (Recommended)
- Tomcat + Static Web Server
- Tomcat only

1.1.3 Hassle-free Deployment with Docker (HDD)

The setup is preferable if you have Docker installed in your machine. You may deploy the full search engine stack only with a few commands.

Docker Installation

Please refer to the Docker website for installation of Docker in different distributions.

Installing Docker in CentOS:

1. Set up repository

```
sudo yum install yum-utils
sudo yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo
```

2. Install docker engine

```
sudo yum install docker-ce docker-ce-cli containerd.io
```

3. Start Docker

```
sudo systemctl start docker
```

4. Verify that docker is installed correctly by running the `hello-world` image

```
sudo docker run hello-world
```

For more information, please refer to docker's installation guide.

Docker Compose Installation

Please refer to the Docker website for installation of Docker Compose in different distributions.

Installing docker compose in CentOS:

1. Curl a stable release of docker compose

```
sudo curl -L "https://github.com/docker/compose/releases/download/1.29.1/docker-compose-$(un
```

2. Add executable permission

```
sudo chmod +x /usr/local/bin/docker-compose
```

3. Create symbolic link

```
sudo ln -s /usr/local/bin/docker-compose /usr/bin/docker-compose
```

For more information, please refer to the installation guide of docker compose.

Starting the stack

It is recommended to use docker compose for starting up the whole search engine stack instead of installing it separately. It is done in the backend directory `COMP4321-Project` where `docker-compose.yml` resides.

By default, it assumes the RocksDB is stored in `$PWD/db`. You may want to change the mount path or use docker volume instead by modifying `docker-compose.yml`.

```
docker-compose build
docker-compose up -d
```

You may want to change the listen port in `docker-compose.yml`. The default port is `3000`.

1.1.4 Tomcat + Static Web Server

Pre-requisite

Java

The project runs with Java 8. It is recommended to have JDK 8 installed.

Gradle is used for dependency management for this project. The gradle wrapper script will be able to detect your Java installation. Make sure the java binary directory is in `$PATH` and `$JAVA_HOME` is set to the Java installation directory, e.g. `/usr/bin/java`.

Tomcat

The search engine backend APIs are served with `Apache Tomcat` server. The recommended version is `10.0.5`.

Download and extract the package to any location you prefer. Alternatively, you may install Tomcat through your system's package manager. You may want to set the environmental variable `CATALINA_HOME` variable to the installation location `apache-tomcat-10.0.5`.

Node.js and Yarn

The project is created with `create-react-app`. To setup the dependencies for building the app, the yarn package manager needs to be set up.

To install nodejs (and npm), it can be done with package manager.

Debian and Ubuntu:

```
sudo apt update
sudo apt install npm
```

CentOS 7:

```
curl -L https://rpm.nodesource.com/setup_10.x | sudo bash -
sudo yum install nodejs
```

MacOS:

```
brew install yarn
```

To install `yarn`, it can with installed with npm

```
sudo npm install --global yarn
```

Please refer to the yarn website for alternative setup instructions.

Static Web Server

To serve the UI website, you will need a static web server to serve the files. Moreover, it should be able to proxy requests to support API calls to the backend JSP web server. We recommend using `nginx` for this purpose.

Building the Backend

The commands and relative paths listed in this section `Building the Backend` are executed in/relative to directory `COMP4321-Project/`. Please change directory into the directory `COMP4321-Project/` to execute the commands.

Dependency Management

Dependency in backend directory `COMP4321-Project` is handled by Gradle.

Building the CLI for backend

For simplicity, you can build a jar file with all libraries packaged in a single jar. Run the shadowJar Gradle task for building a fat jar.

```
./gradlew shadowJar
```

The compiled jar file will be located at `build/libs/COMP4321Project-1.0-all.jar`.

Building the Web Application for backend

Web application WAR file can be built for deployment with Tomcat server through `war` Gradle task.

```
./gradlew war
```

The compiled war file will be located at `build/libs/COMP4321Project-1.0.war`.

Building the Frontend

The commands and relative paths listed in this section `Building the Frontend` are executed in/relative to directory `comp4321-proj-ui/`.
Please change directory into the directory `comp4321-proj-ui/` to execute the commands listed below.

Dependency Management

Dependency in frontend directory `comp4321-proj-ui` is handled with yarn package manager. All the dependencies should first be installed with the following command.

```
yarn install
```

Building the frontend UI

To build the static web pages, you may use the `build` script in `react-create-app`. The resulting pages will be stored in the `./build` folder.

```
yarn build
```

Deploying the UI

The commands and relative paths listed in this section `Deploying the UI` are executed in/relative to directory `comp4321-proj-ui/`. Please change directory into the directory `comp4321-proj-ui/` to execute the commands listed below.

Setting up the Static Web Server You may setup any of the static web server of your preference. Copy all the build files to the web server directory for serving. The below command assumes nginx server is employed and files in `/usr/share/nginx/html` are served as the root.

```
cp -r ./build/. /usr/share/nginx/html
```

The setup can be verified by loading the home page.

Linking the Backend API

For the actual search functionality, you need to setup proxy to point all `/api` requests to the backend JSP server. For example, if you are using nginx, you may want to setup like the config below (assume backend is deployed at port `3001`) .

```
server {
    location /api {
        proxy_pass  http://localhost:3001;
    }
}
```

Deploying the Backend Search Engine

The commands and relative paths listed in this section `Deploying the Backend Search Engine` are executed in/relative to directory `COMP4321-Project/`. Please change directory into the directory `COMP4321-Project/` to execute the commands.

The search engine web application is packaged in a single WAR file. There are several possible ways to deploy it.

Crawling and Pre-requisites

Before deploying the web application, it is required for the pages to be crawled and indexed into RocksDB for at least once. It can be done with:

1. convenient script `phase1.sh`

   ```
   ./phase1.sh --crawl --num-docs=20000
   ```

   Options available for the script can be shown by

   ```
   ./phase1.sh --help
   ```

2. shadowJar built in a section before

   ```
   java -jar build/libs/COMP4321Project-1.0-all.jar --crawl --num-docs=20000
   ```

   Again, available arguments can be list by

   ```
   java -jar build/libs/COMP4321Project-1.0-all.jar --help
   ```

Environmental variable `SE_DB_BASE_PATH` needs to be set to the RocksDB storage folder. By default, the RocksDB is stored in `$PWD/db`.

Method 1: Convenient Script

To your convenience, we provide a script that do all the things for you. The script will download and extract Tomcat automatically. Then, the web application will be built (with `war` Gradle task) and deployed to Tomcat.

To start the server, run the following:

```
./phase2.sh startup
```

By default, the application is deployed at port `8080`. You may visit http://localhost:8080 to see it in action.

To stop the server, run the following:

```
./phase2.sh shutdown
```

The port can be changed by modifying `tomcat/conf/server.xml` AFTER running both `startup` and `shutdown` of `phase2.sh`. By changing `port="8080"` in the following block to `port="3001"` or other desired port, the Tomcat server will serve at port `3001` or other port when it is started.

```
<Connector port="8080" protocol="HTTP/1.1"
           connectionTimeout="20000"
           redirectPort="8443"/>
```

Method 2: Manual Deploying

The WAR file is located at `build/libs/COMP4321Project-1.0.war` after running the `war` Gradle task. You may follow the instruction of your web server distribution for deploying a WAR application.

For Tomcat, copy the generated WAR file to `$CATALINA_HOME/webapps/` to deploy the application. You may refer to the official documentation for details.

1.2 Using the Search

This part will be a detailed guide of using the web-based search engine we had developed.

1.2.1 Inputting Query

In inputting a search query, user can simply type in the keywords in the text box as shown in Figure 1. There will also be a drop down list of suggestions based on the prefix a user had inputted. If user finds any of the suggestions useful and related, user can simply click on the suggestion, then the input query will automatically selected to be the selected suggestion. After typing in a search query, user can select the Search button next to the text box to start the triggering the searching process.

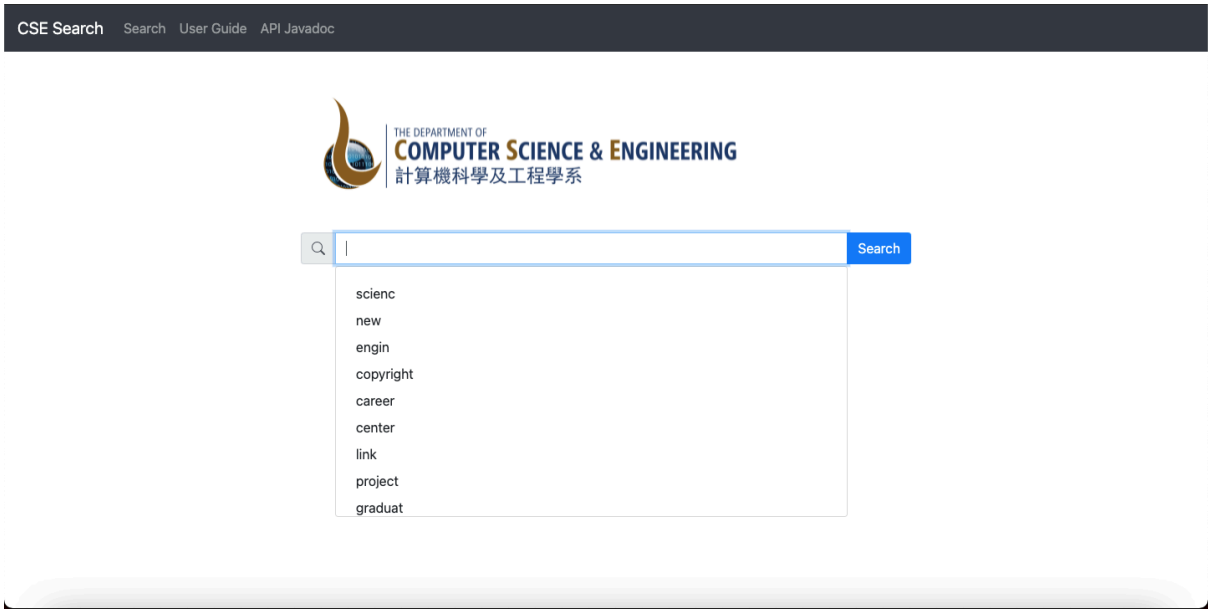

Figure 1: Query Input Page

1.2.2 Waiting for Result

When the program is continuously running to find relevant web pages, Figure 2 will be shown. In this stage, user simply has to patiently wait for the results retrieved by the search engine.

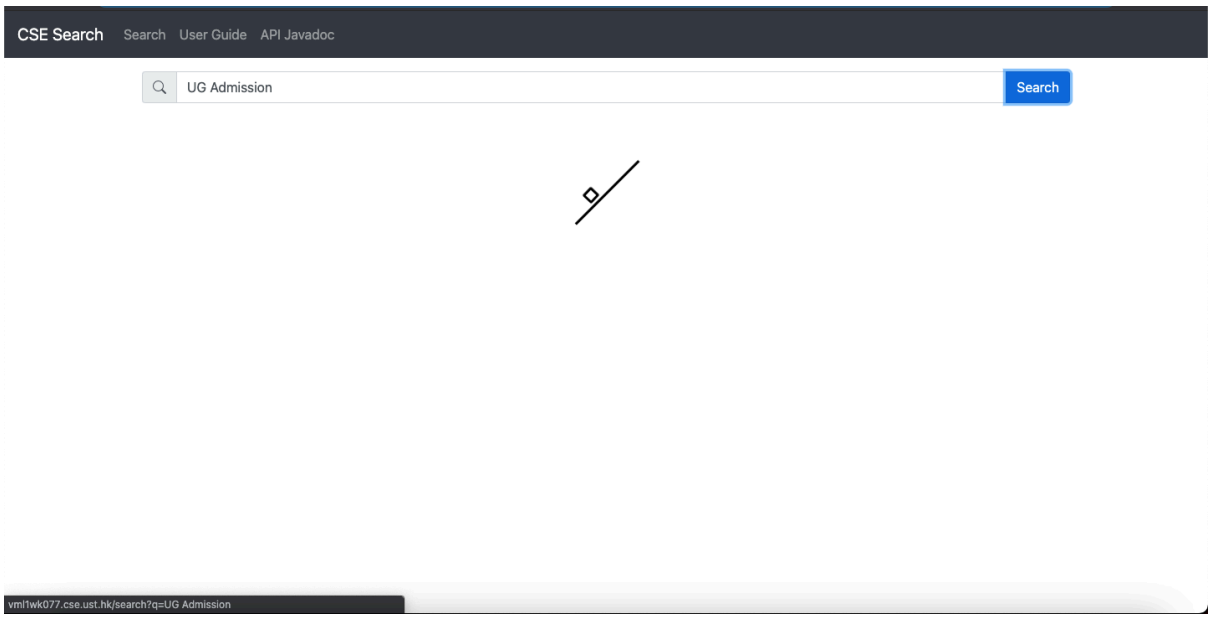

Figure 2: Loading Page

1.2.3 Viewing Result

If no relevant web pages are found, Figure 3 will be shown. In this case, user is suggested to modify their inputted query so as to obtain relevant results.

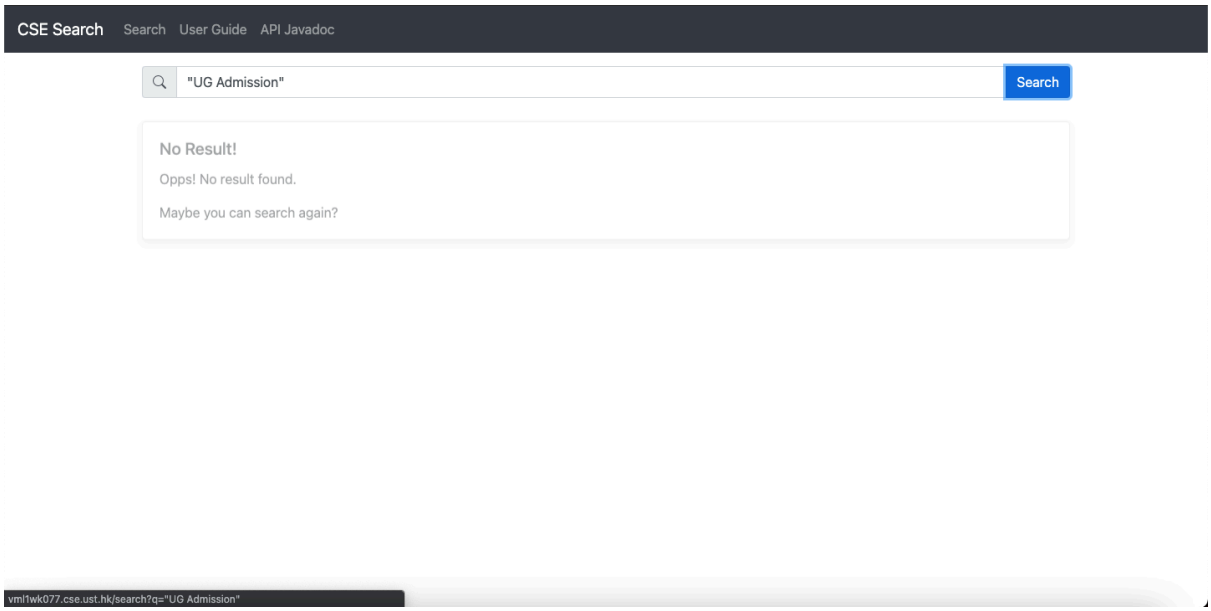

Figure 3: Result Page 1

If relevant web pages are found, the results will be displayed in a format as shown in Figure 4. For each relevant results found, user can view the similarity score as well as other basic information including page title, url, last modification date, size of page, keywords and their respective frequencies and child links. User can directly click on the url and child links if one is interested to know more about the web page. To view 5 more child links, user can click on `[Show more...]` and by recursively clicking on that, user will be able to view all related child links.

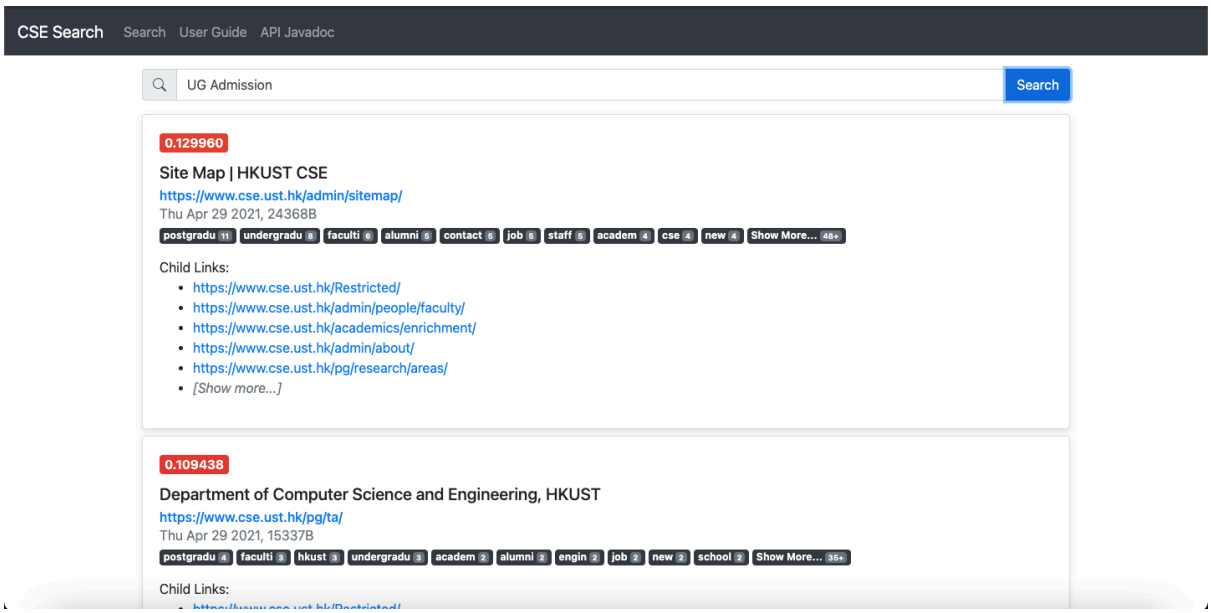

Figure 4: Result Page 2

1.3 Advanced Search

We had implemented phrase search and extended boolean model in our search engine. Here, will be a guide on how to specify these two searches.

1.3.1 Phrase Search

To specify phrase search, user has to use double quotes to quote the phrase. Figure 5 is an example illustrating the use of phrase search when inputting search query.

Figure 5: Phrase Search Example

1.3.2 Extended Boolean Model

As for specifying boolean queries, user has to use terms “AND” or “OR” according to one’s own preference. Figures 6 and 7 are examples illustrating the input of boolean search query.

Figure 6: Boolean Search AND Example

Figure 7: Boolean Search OR Example