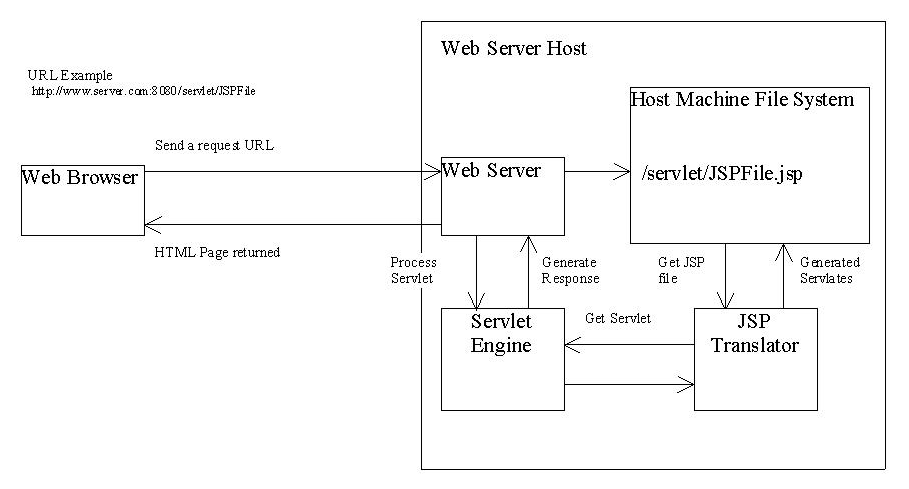
Name: Poon Ka Hang

ITSC: khpoon

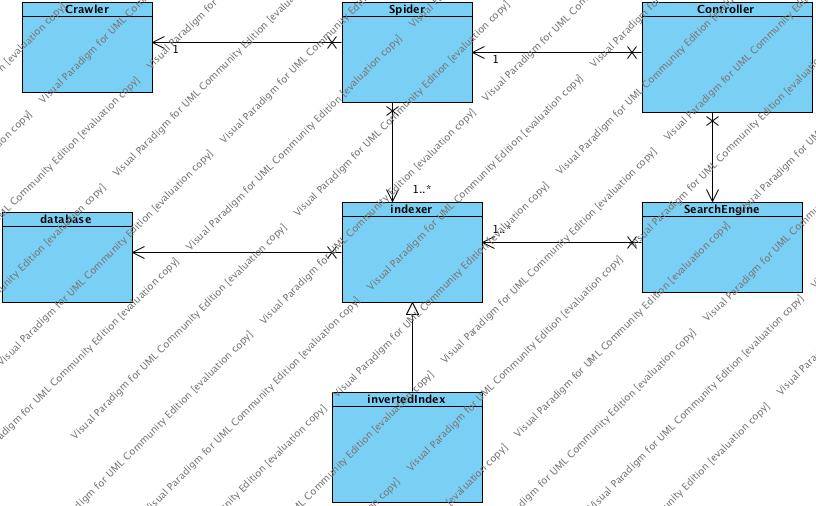
SID: 10462347

COMP4321 Project Report

# Overall design of the system



We implement JSP to search the result from the database using the Java search engine. We have used JDBM and htmlparser library.



The system is implement in two parts. The first part is spider. Spider can save all pages to the index. Then save the index into the database. The second part is search engine. Input keywords into the search engine and calculate the ranking score for each related page. Then return the result to the webpage.

# The file structures used in the index database

## Indexer class

A class contains a hashtable. It can get can value using index or get index using value.

Page **Indexer**

URL <-> PageID

Word Indexer

Word <-> WordID

Title Indexer

Title <-> TitleID

ChildParent

Parent-ID <-> ChildID

## InvertedIndex class

A class contains a hashtable. It can get can value using index or get index using value.

inverted index

word ID -> {pageID, Freq}

Forward index

Page-ID -> {keywords}

Link-based index

Parent-ID -> {ChildID}

## PageProperty class

Page properties

Page-ID -> title, URL, last-date-of-modification, size,

## Explanation

For each fetch, we will get the page id from the url. Then we can find out another information.

* Page-ID -> title, URL, last-date-of-modification, size
* Page-ID -> {keywords}
* Word <-> WordID
* Word ID -> {pageID, Freq}
* Parent-ID -> {ChildID}

Finally, we can generate the spider\_result.txt using this method

# Algorithms used

## Top frequency in a page

We count the maximum number of keywords frequency in a file when we are fetching a page.

## Term Weight

*wij =* (*tfij /maxl*{*tflj*}) *• idfj*

## Cosine Similarity Measures



Query weight is set to 1

## Matching

We use stopword.txt to filter out the stopword. Then stem the keyword. After that

## Title Search

If keyword find in title

Link.Ranking += 1;

Since all of the score are between 0 to 1, the link which match the title will go to the top ranking.

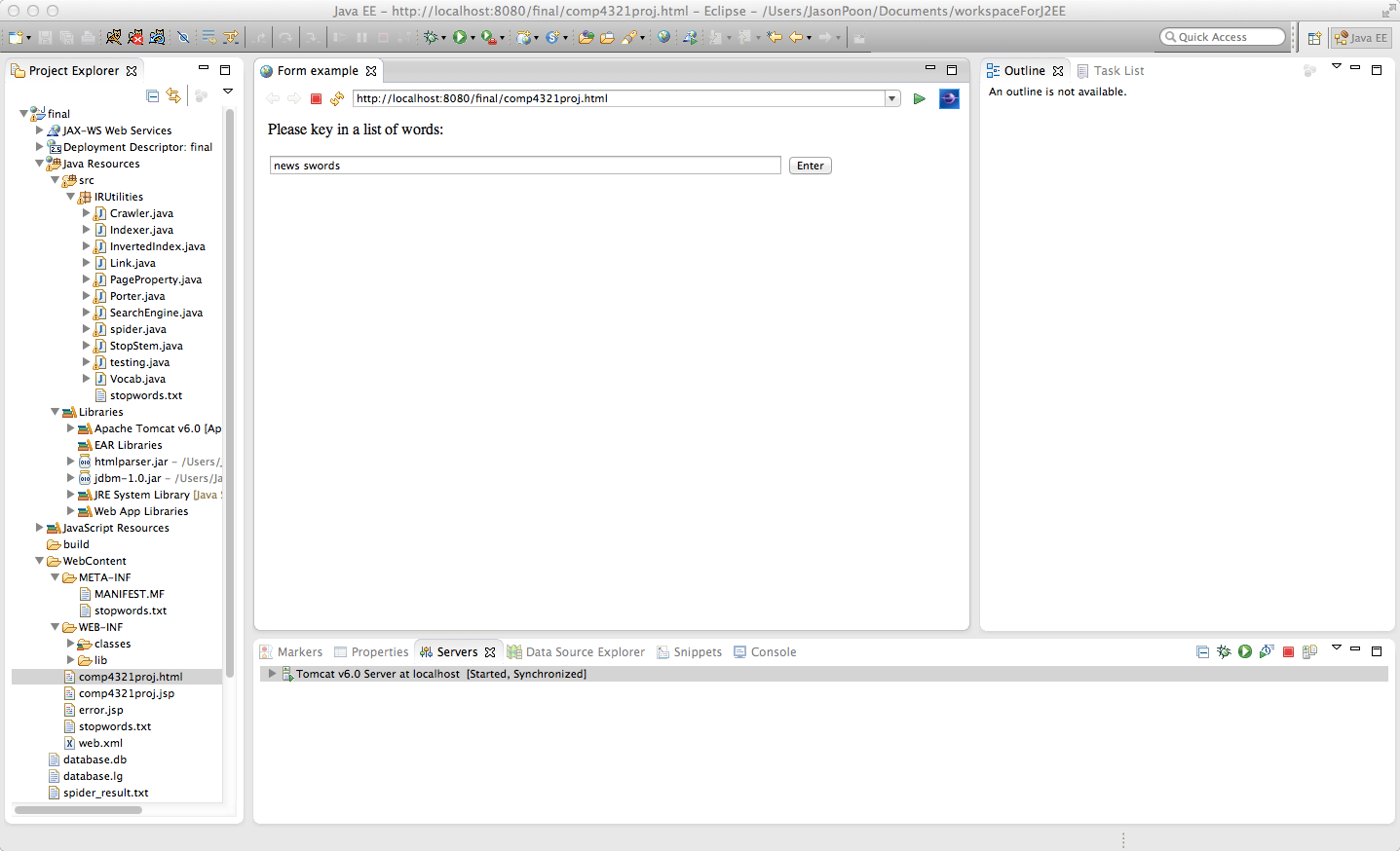
# Installation procedure (it could be as simple as “Type make in the project directory”)

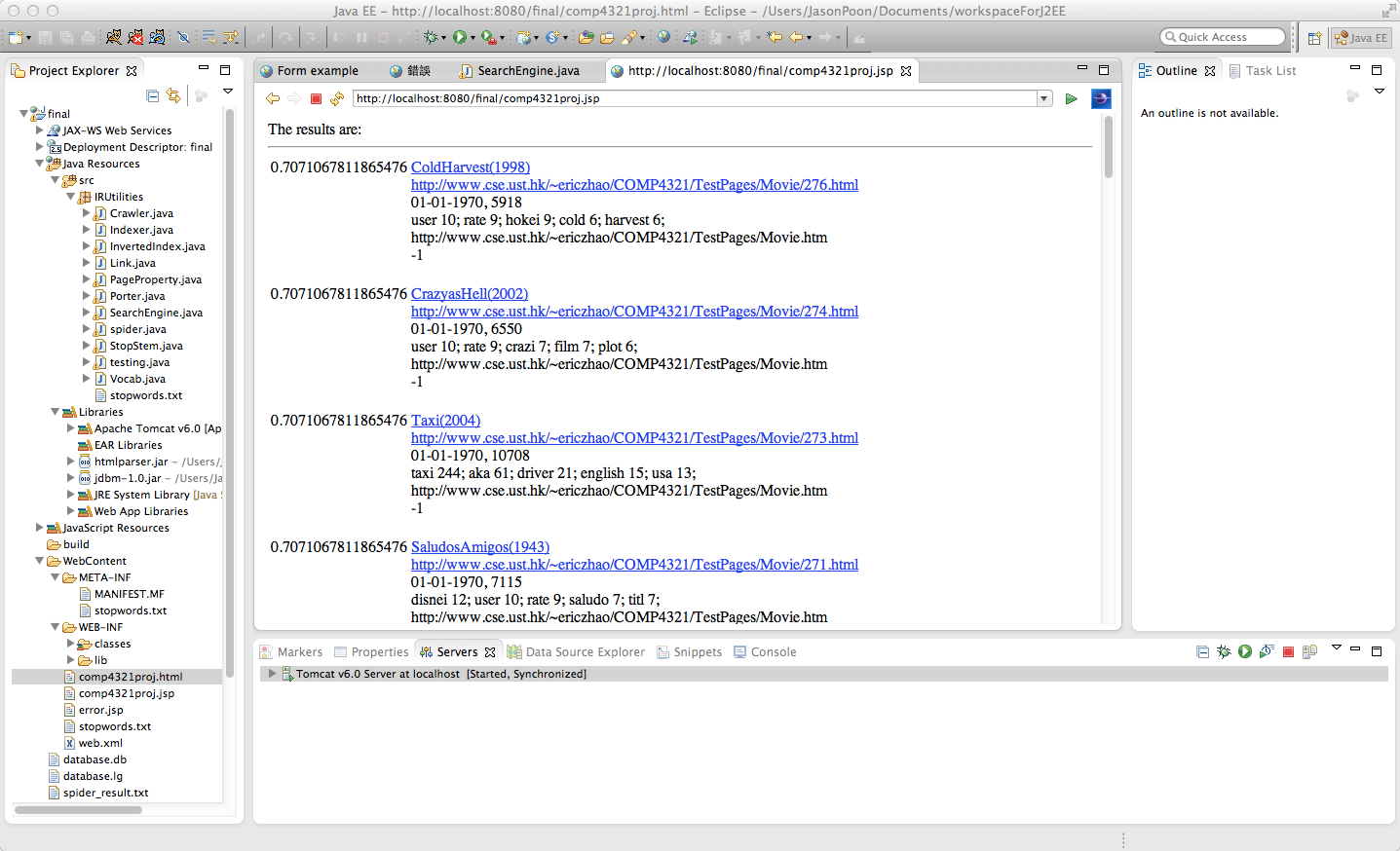
1. Place \*.java in eclipse.
2. Run spider.java in eclipse to generate database.db and database.lg
3. Place them to public\_html/
4. Upload the package to the FTP Server.
5. Place all \*.jsp and public\_html folder.
6. Place all database file and public\_html folder.
7. Place all \*.class file to WEB-INF/classes/
8. Place all \*.jar to WEB-INF/lib/
9. Change directory to public\_html/
10. Type chmod g+r WEB-INF –R
11. Type chmod g+rx \*.jsp

# Highlight of features beyond the required specification

N/A

# Testing of the functions implemented; include screenshots if applicable in the report





# Conclusion: What are the strengths and weaknesses of your systems; what you would have done differently if you could re-implement the whole system; what would be the interesting features to add to your system, etc.

## Strengths

I cannot see any strengths in my program if it compare to google.

## Weaknesses

Speed is slow

## have done differently

use another algorithm. This algorithm is a little bit slow

## interesting features

I will add search with some operator. Like “&” and “|”.