RMIT UNIVERSITY

ASSIGNMENT REPORT

November 2013

Assignment Title: DHIS2

Student name	:**Ho Xuan Hieu**
	Le Nguyen Huan
	Luong Hong Ngoc
Course	: **COSC2442- Software Architecture**
Lecturer	: **Nguyen Ngoc Thanh**

DHIS2 INVESTIGATION

I. Introduction

This technical report will introduce one of the best way for managing data in the industrial world, DHIS-2, how to implement it and guide you through a way of sending JSON data through an url by using HTTPClient. DHIS-2 is an abbreviation of District Health Information System 2. It is first developed by "the Health Information System Programme (HISP) as an open and globally distributed process among developers". **()** It is used for import and export data, manipulate data, and even draw a report out of those data. Moreover, JSON has been used a lot when a client want to communicate to a server. It is much faster and more convenient than to send a whole file to the server.

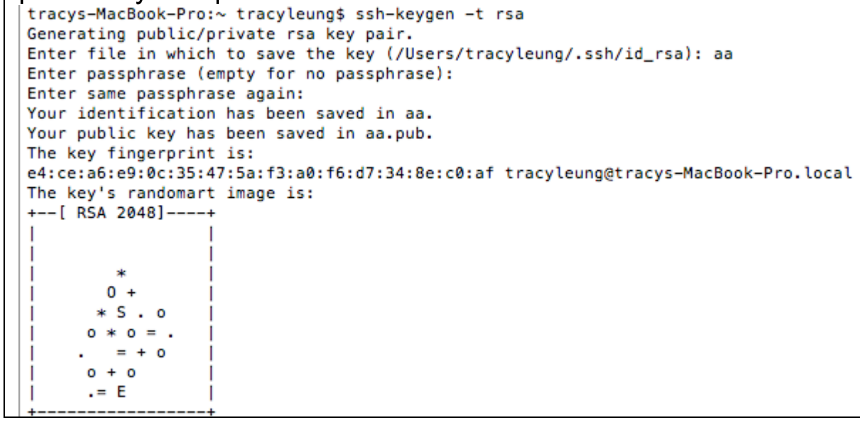
II. Body

A. The implementation of DHIS-2

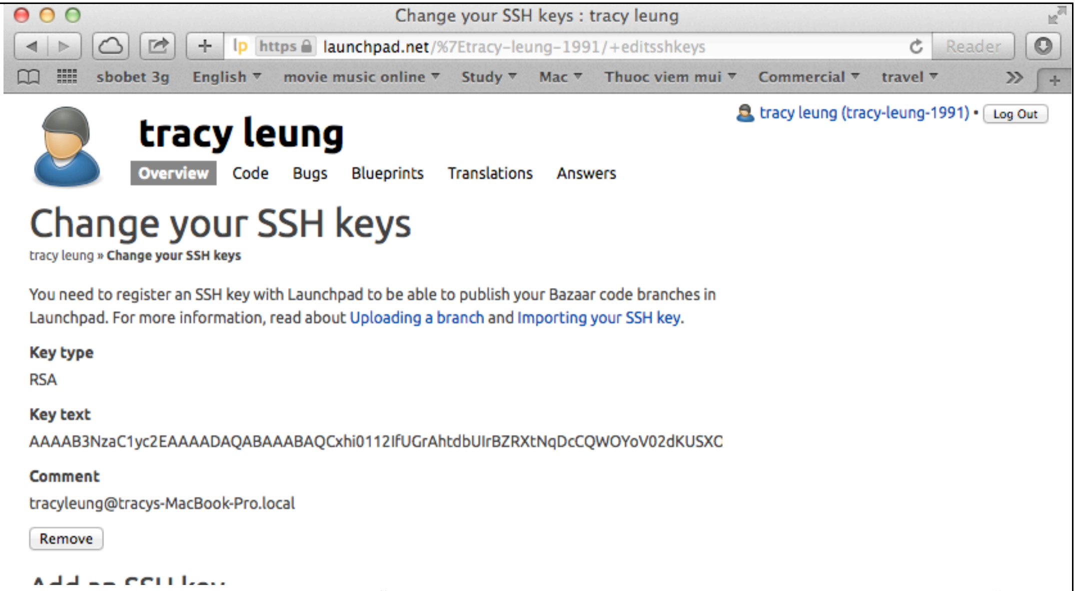
The implementation here does not mean downloading a package or an executable file and installing it. Developers have to go step by step. Since our teammates are all use a Mac, so the instruction is only useful for Mac users. According to the instruction from the website, developers have to install maven, Java SDK, Bazzar, and so on and it is obviously right.

There is a little notice that bazzar is not installed in Mac OSX by default so developers have to install it by using MacPorts with the following command "*sudo port install brz*". One more step before the dhis-2 can be run.
Go to the launch pad website (launchpad.net) and register an account.

Then create a ssh key by using the following command "ssh-keygen –t –rsa". The public and private rsa key pair will be generated; copy a content of the public key and paste in the text area of the website.



```
tracys-MacBook-Pro:~ tracyleung$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/Users/tracyleung/.ssh/id_rsa): aa
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in aa.
Your public key has been saved in aa.pub.
The key fingerprint is:
e4:ce:a6:e9:0c:35:47:5a:f3:a0:f6:d7:34:8e:c0:af tracyleung@tracys-MacBook-Pro.local
The key's randomart image is:
+--[ RSA 2048]----+
|                 |
|                 |
|        *        |
|       O +       |
|      * S . o    |
|     o * o = .   |
|    .   = + o    |
|     o + o       |
|     .= E        |
+-----------------+
```



tracy leung

Change your SSH keys

tracy leung » Change your SSH keys

You need to register an SSH key with Launchpad to be able to publish your Bazaar code branches in Launchpad. For more information, read about Uploading a branch and Importing your SSH key.

Key type

RSA

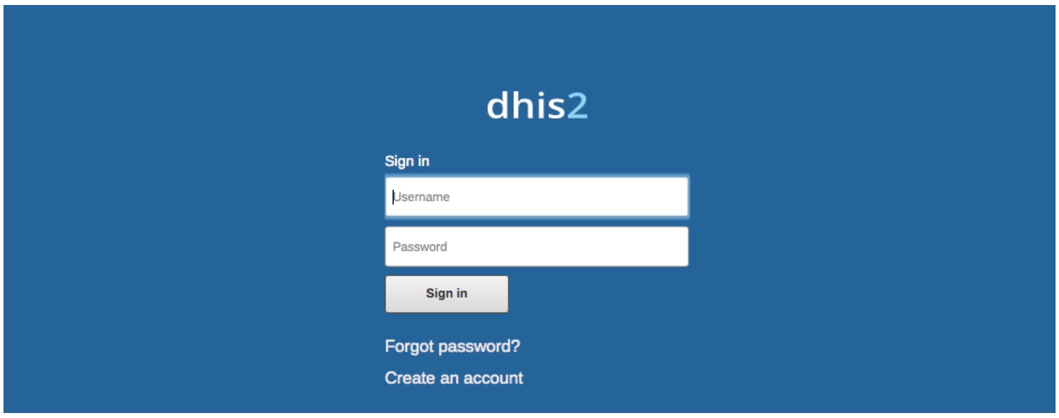
Key text

AAAAB3NzaC1yc2EAAAADAQABAAABAQCxhi0112IfUGrAhtdbUIrBZRXtNqDcCQWOYoV02dKUSXC

Comment

tracyleung@tracys-MacBook-Pro.local

Back to the terminal, type in *"bzr lb-login <usernameFromLaunchPad>"* and "*bzr branch lp:dhis2*". Now the dhis-2 can be run with the following from its directory "*dhis-2/dhis-web/dhis-web-portal/*" with a command "*mvn jetty:run-war*" and a local server will be
created.



The default account will be

Username: **admin**

Password: **district**

In some computers, "Out of java memory" error will occur. For fixing it, all you have to do is editing you ".bash_profile" file in your home by using: *vim ~/.bash_profile* or *nano ~/.bash_profile*.. In the edit mode, add these following lines:

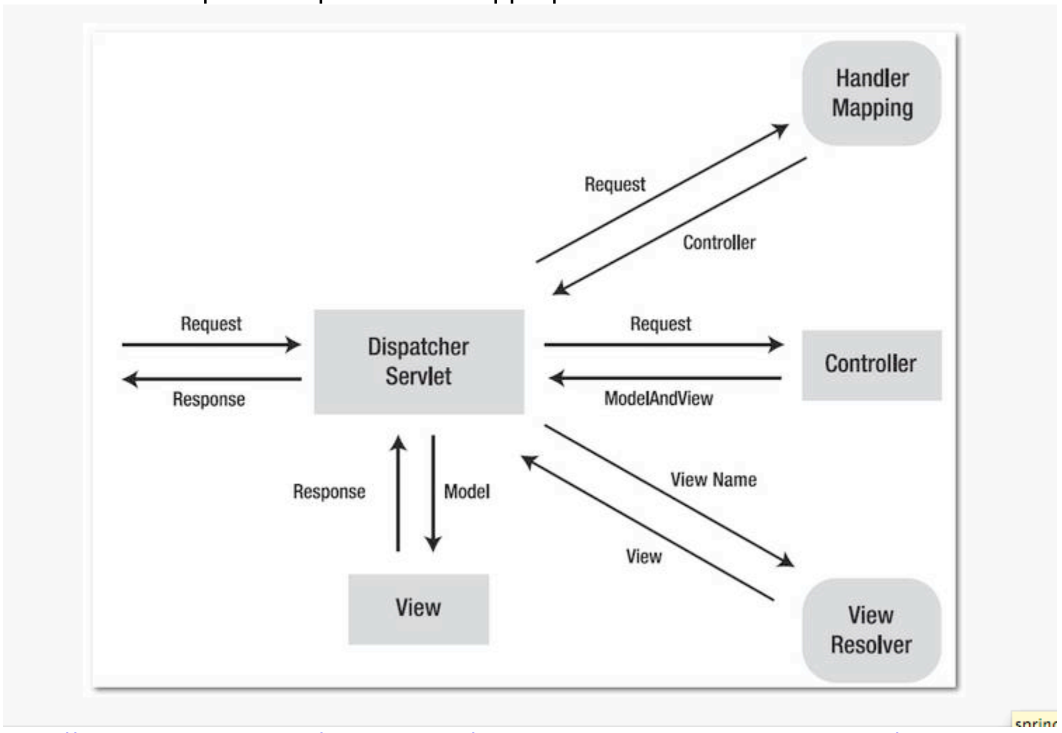
```
export MAVEN_OPTS="-Xmx512m –XX:MaxPermSize=512m"
export DHIS2_HOME="/Users/<your home>/DHIS2_HOME"
```

DHIS-2 is used for data management so we also need to connect to a database in order insert, update, and delete data. Developers can easily connect to the database with 3 steps. First go to the home directory, create a folder name "DHIS_HOME" as shown above and create a file that named "hibernate.properties". Finally developers need to copy code belows and save. The database will be created automatically in your mysql database app such as mysql workbench, sequel pro, and so on.

```
hibernate.dialect = org.hibernate.dialect.MySQLDialect
hibernate.connection.driver_class = com.mysql.jdbc.Driver
hibernate.connection.url = jdbc:mysql://localhost/dhis2
hibernate.connection.username = dhis
hibernate.connection.password = dhis
hibernate.hbm2ddl.auto = update
```

B. Data Transfering Investigation

For this project, it uses Spring MVC. This means it consists 3 types model, view, and controller. These 3 types are quite similar to developers in other languages but there are a little bit difference. First of all, view will contain jsp files which will be used to display the web page. Secondly and also the last difference, there is a servlet, which is called "Dispatcher Servlet", in the middle to work as the front controller to dispatch requests to an appropriate handler.



http://www.mkyong.com/spring-mvc/spring-mvc-hello-world-example/

After a long time of research, a java file, "DataValueSetController.java", is found. This controller will play a role of sending and receiving data from the url with both GET and POST method. By default, it is able to received and interpreted JSON, XML, SDMX+SML file. After the interpretation, the data will be put in an object and saved into the database, our local database.

C. Building an application (server – client) for transfering data via url

By default, it is much more easier for developers to program a spring mvc in the Intellij Ultimate but here the standard version is used. For the instruction, please refer to the following link:

http://alfasin.com/setting-up-spring-web-project-on-intellij-using-maven/

In order to work with jetty, one of the lightest servlet engin and http server, deverlopers should copy and paste these line in their project pom.xml file

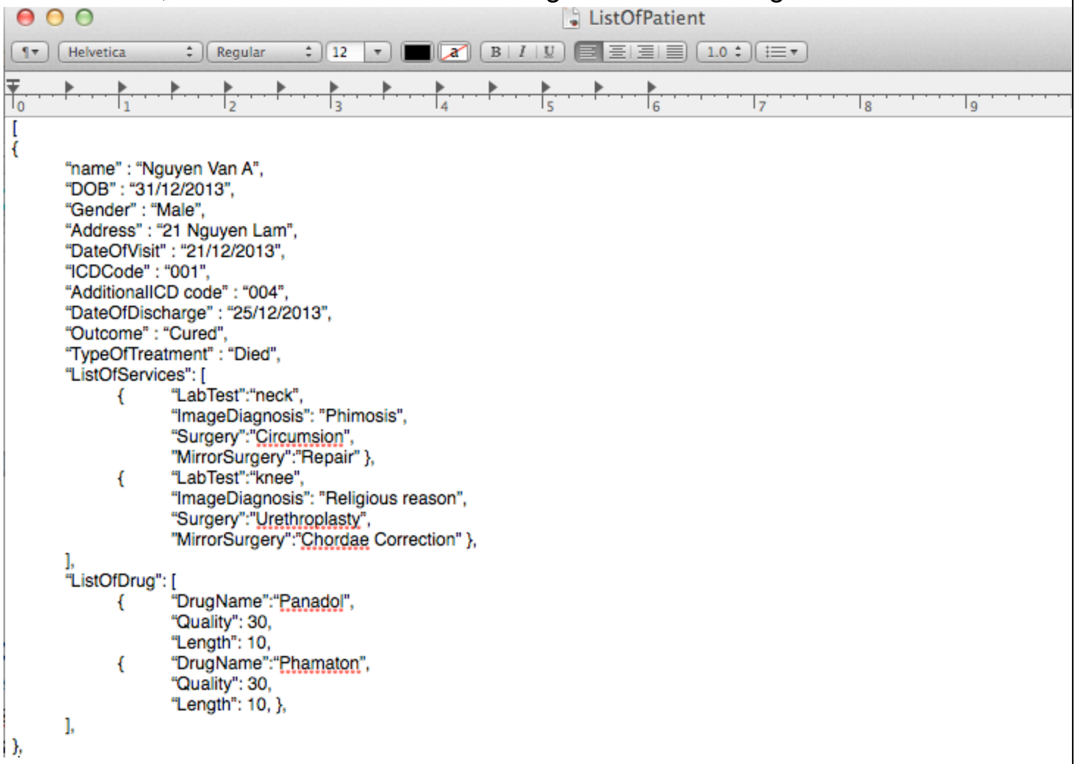
```
<repositories>
   <repository>
      <id>java.net</id>
      <url>http://download.java.net/maven/2/</url>
   </repository>
</repositories>
```

```
  <build>
    <plugins>
        <plugin>
           <groupId>org.apache.maven.plugins</groupId>
           <artifactId>maven-compiler-plugin</artifactId>
           <configuration>
              <source>1.7</source>
              <target>1.7</target>
           </configuration>
        </plugin>

        <plugin>
           <groupId>org.mortbay.jetty</groupId>
           <artifactId>maven-jetty-plugin</artifactId>
        </plugin>
     </plugins>
  </build>
```

Now it works find with jetty.

D. JSON file

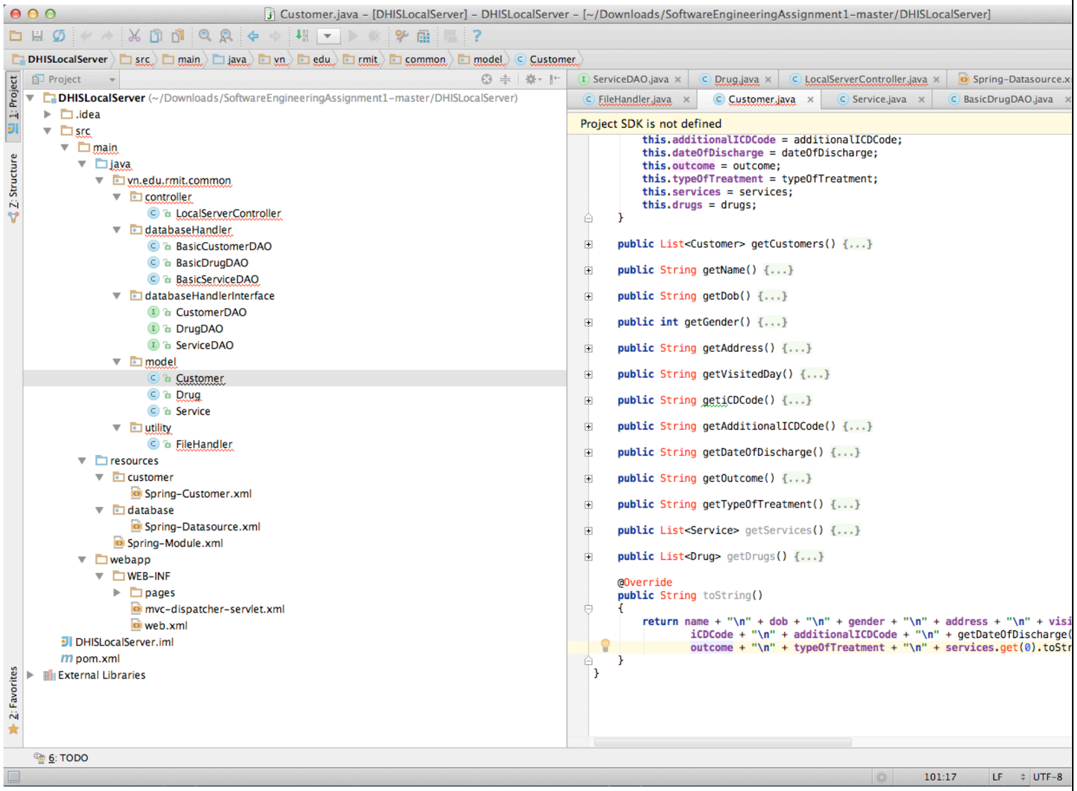
In this tutorial, we create Patient list in json file. This file is contain name, date of birth, gender, address, date of visit, ICD code, Date of discharge, Outcome, Type of treatment, list of services and list of drug like the following:



```
[
{
        "name" : "Nguyen Van A",
        "DOB" : "31/12/2013",
        "Gender" : "Male",
        "Address" : "21 Nguyen Lam",
        "DateOfVisit" : "21/12/2013",
        "ICDCode" : "001",
        "AdditionalICD code" : "004",
        "DateOfDischarge" : "25/12/2013",
        "Outcome" : "Cured",
        "TypeOfTreatment" : "Died",
        "ListOfServices": [
                {       "LabTest":"neck",
                        "ImageDiagnosis": "Phimosis",
                        "Surgery":"Circumsion",
                        "MirrorSurgery":"Repair" },
                {       "LabTest":"knee",
                        "ImageDiagnosis": "Religious reason",
                        "Surgery":"Urethroplasty",
                        "MirrorSurgery":"Chordae Correction" },
        ],
        "ListOfDrug": [
                {       "DrugName":"Panadol",
                        "Quality": 30,
                        "Length": 10,
                {       "DrugName":"Phamaton",
                        "Quality": 30,
                        "Length": 10, },
        ],
},
```

E. Server side

In model we have Customer, Drug and Service class to contain our data.



Customer.java - [DHISLocalServer] - DHISLocalServer - [~/Downloads/SoftwareEngineeringAssignment1-master/DHISLocalServer]

```
DHISLocalServer (~/Downloads/SoftwareEngineeringAssignment1-master/DHISLocalServer)
  .idea
  src
    main
      java
        vn.edu.rmit.common
          controller
            LocalServerController
          databaseHandler
            BasicCustomerDAO
            BasicDrugDAO
            BasicServiceDAO
          databaseHandlerInterface
            CustomerDAO
            DrugDAO
            ServiceDAO
          model
            Customer
            Drug
            Service
          utility
            FileHandler
      resources
        customer
          Spring-Customer.xml
        database
          Spring-Datasource.xml
        Spring-Module.xml
      webapp
        WEB-INF
          pages
          mvc-dispatcher-servlet.xml
          web.xml
  DHISLocalServer.iml
  pom.xml
External Libraries
```

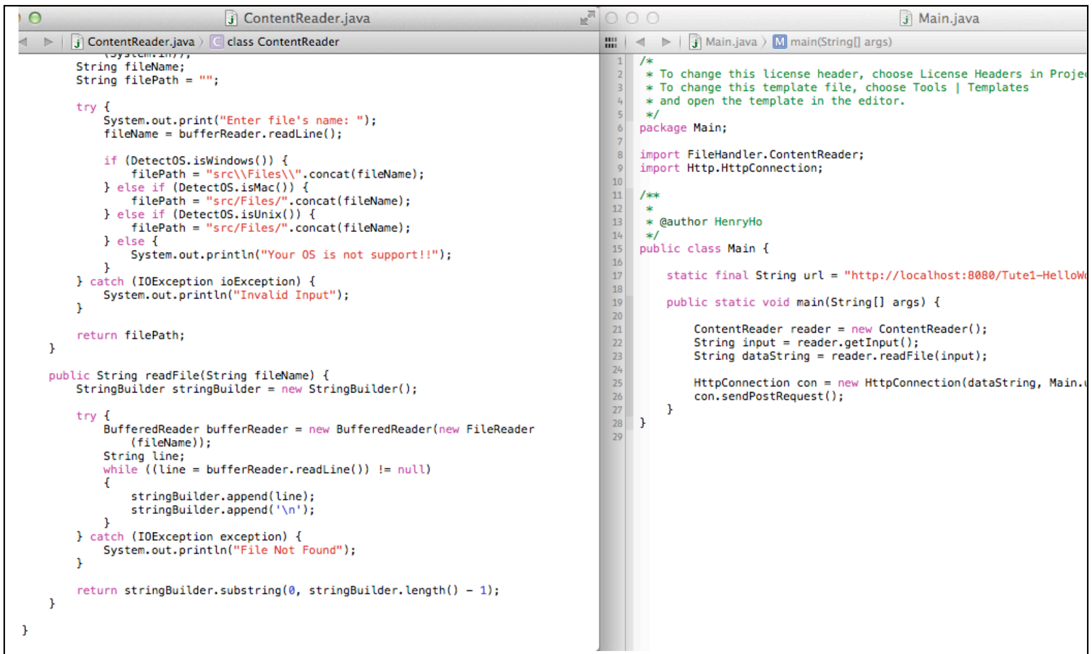
Project SDK is not defined

```java
            this.additionalICDCode = additionalICDCode;
            this.dateOfDischarge = dateOfDischarge;
            this.outcome = outcome;
            this.typeOfTreatment = typeOfTreatment;
            this.services = services;
            this.drugs = drugs;
        }

        public List<Customer> getCustomers() {...}

        public String getName() {...}

        public String getDob() {...}

        public int getGender() {...}

        public String getAddress() {...}

        public String getVisitedDay() {...}

        public String getiCDCode() {...}

        public String getAdditionalICDCode() {...}

        public String getDateOfDischarge() {...}

        public String getOutcome() {...}

        public String getTypeOfTreatment() {...}

        public List<Service> getServices() {...}

        public List<Drug> getDrugs() {...}

        @Override
        public String toString()
        {
            return name + "\n" + dob + "\n" + gender + "\n" + address + "\n" + visi
                    iCDCode + "\n" + additionalICDCode + "\n" + getDateOfDischarge(
                    outcome + "\n" + typeOfTreatment + "\n" + services.get(0).toStr
        }
    }
```

In the controller we use HttpServletRequest.getParameter function to get the data. Then we convert Json data to an Object through JsonObject class and insert them to our database.

F. Client side

We use HttpClient to send our json file. In the ContentReader.java and Main.java, it will read the content of our file and send file through the URL. In Main.java class, it use HttpConnection.java to setting a header, Entity and the body.



```java
        String fileName;
        String filePath = "";

        try {
            System.out.print("Enter file's name: ");
            fileName = bufferReader.readLine();

            if (DetectOS.isWindows()) {
                filePath = "src\\Files\\".concat(fileName);
            } else if (DetectOS.isMac()) {
                filePath = "src/Files/".concat(fileName);
            } else if (DetectOS.isUnix()) {
                filePath = "src/Files/".concat(fileName);
            } else {
                System.out.println("Your OS is not support!!");
            }
        } catch (IOException ioException) {
            System.out.println("Invalid Input");
        }

        return filePath;
    }

    public String readFile(String fileName) {
        StringBuilder stringBuilder = new StringBuilder();

        try {
            BufferedReader bufferReader = new BufferedReader(new FileReader
                (fileName));
            String line;
            while ((line = bufferReader.readLine()) != null)
            {
                stringBuilder.append(line);
                stringBuilder.append('\n');
            }
        } catch (IOException exception) {
            System.out.println("File Not Found");
        }

        return stringBuilder.substring(0, stringBuilder.length() - 1);
    }

}
```

```java
/*
 * To change this license header, choose License Headers in Proje
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package Main;

import FileHandler.ContentReader;
import Http.HttpConnection;

/**
 *
 * @author HenryHo
 */
public class Main {

    static final String url = "http://localhost:8080/Tute1-HelloW

    public static void main(String[] args) {

        ContentReader reader = new ContentReader();
        String input = reader.getInput();
        String dataString = reader.readFile(input);

        HttpConnection con = new HttpConnection(dataString, Main.
        con.sendPostRequest();
    }
}
```

In addition, we have detectOS.java file in Utilities folder to configuration our application will run in different OS.

G. Source Code

For those who want to go for more details, please visit my github repository for a source code: https://github.com/henryho1612/SoftwareEngineeringAssignment1

III. Reference

http://www.mkyong.com/java/how-to-send-http-request-getpost-in-java/
http://www.mkyong.com/java/how-to-detect-os-in-java-systemgetpropertyosname/
http://stackoverflow.com/questions/3324717/sending-http-post-request-in-java
http://stackoverflow.com/questions/8120220/how-to-use-parameters-with-httppost
http://www.mkyong.com/spring/maven-spring-jdbc-example/
http://stackoverflow.com/questions/8779631/jdbc-driver-class-not-found-com-mysql-jdbc-driver
http://www.mkyong.com/spring-mvc/spring-mvc-hello-world-example/