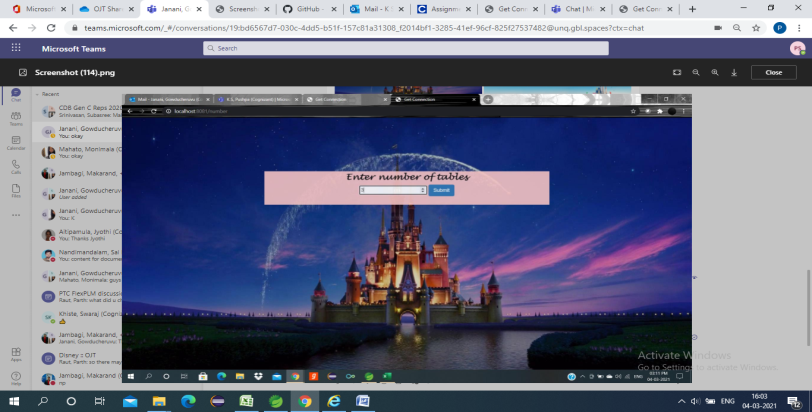
**Project Model :**

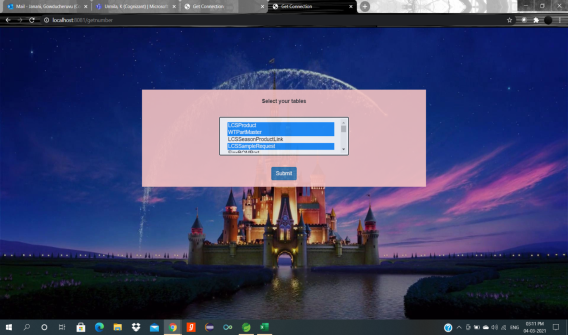
Localhost:portnumber /number

****

Enter the number of tables

OK

**Returned to select table page**

****

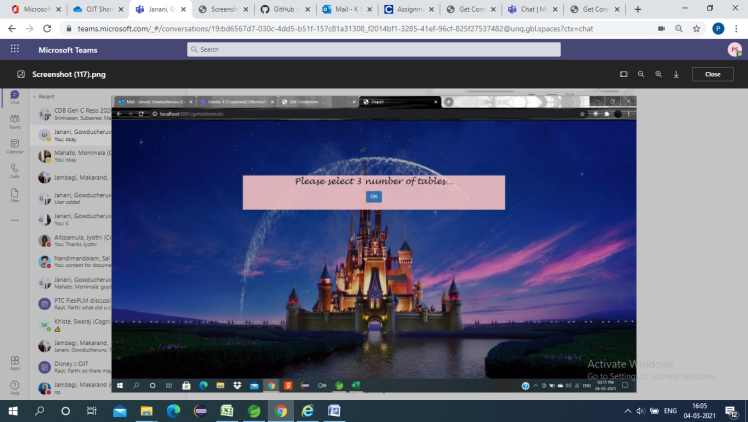
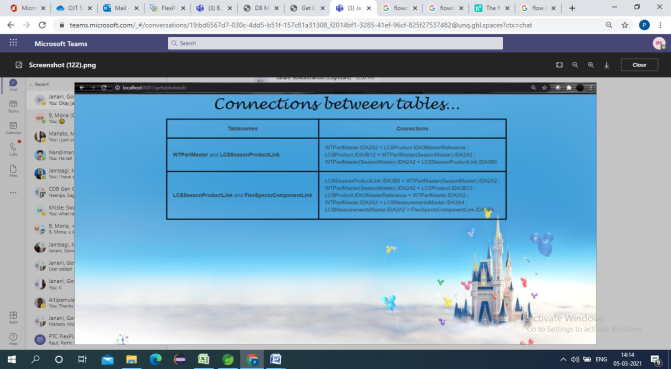
OK

Select the required tables

**Unmatched**

Matched

Entered number== Selected tables

** **

Display data connections

Ask to select the entered number of tables.

**Product/SKU-Season Model**

**Objective:**

* The main goal of designing data model is to retrieve the connections between the tables based on user given inputs.
* The information in the data model is used for defining the relationship between tables.
* Apache POI library is used to read, write and evaluate cells on generated excel file.
* Spring Framework using to create high performing, easily testable and reusable code.
* JSP stands for Java Server Pages.JSP program is a java code which supports HTML tags too. To be more precise, JSP embed html in JAVA using JSP tag.

**Introduction:**

**Technical Stack:**

**Database** Design is a collection of processes that facilitate the designing, development, implementation and maintenance of enterprise data management systems. Properly designed database are easy to maintain, improves data consistency and are cost effective in terms of disk storage space. The database designer decides how the data elements correlate and what data must be stored. The main objectives of database designing are to produce logical and physical designs models of the proposed database system.

**Spring Frame work** will create and manage the instances it will also connects these objects just the way we want that in our object graph picture. It access sort of like a container for object instances, it wraps application in a wrapper that’s called the application context. It manages object life cycle in dependencies for the components and services in our code through application context and through dependency injection. To facilitate this spring frame work comes with web frame work called **Spring MVC**. It easily creates web applications and REST API using the some spring application model and dependency injection concepts.

**Apache POI** (**Poor Obfuscation Implementation)** is an API provided by **Apache foundation** which is a collection of different java libraries. This library gives the facility to read, write and manipulate different Microsoft files such as excel sheet, power-point, and word files.**Maven** is a build automation tool used primarily for Java projects. Maven can also be used to build and manage projects.

**Excel sheet** is mainly used for Data entry, Data management and it enhances the ability to analyze large amount of data.

**Purpose:**

Data model is designed to make certain that data objects offered by the functional teams are represented accurately. It helps business to communicate within and across organizations and recognize correct source of data to populate the model. The data Model should be detailed enough to be used for building the physical database.

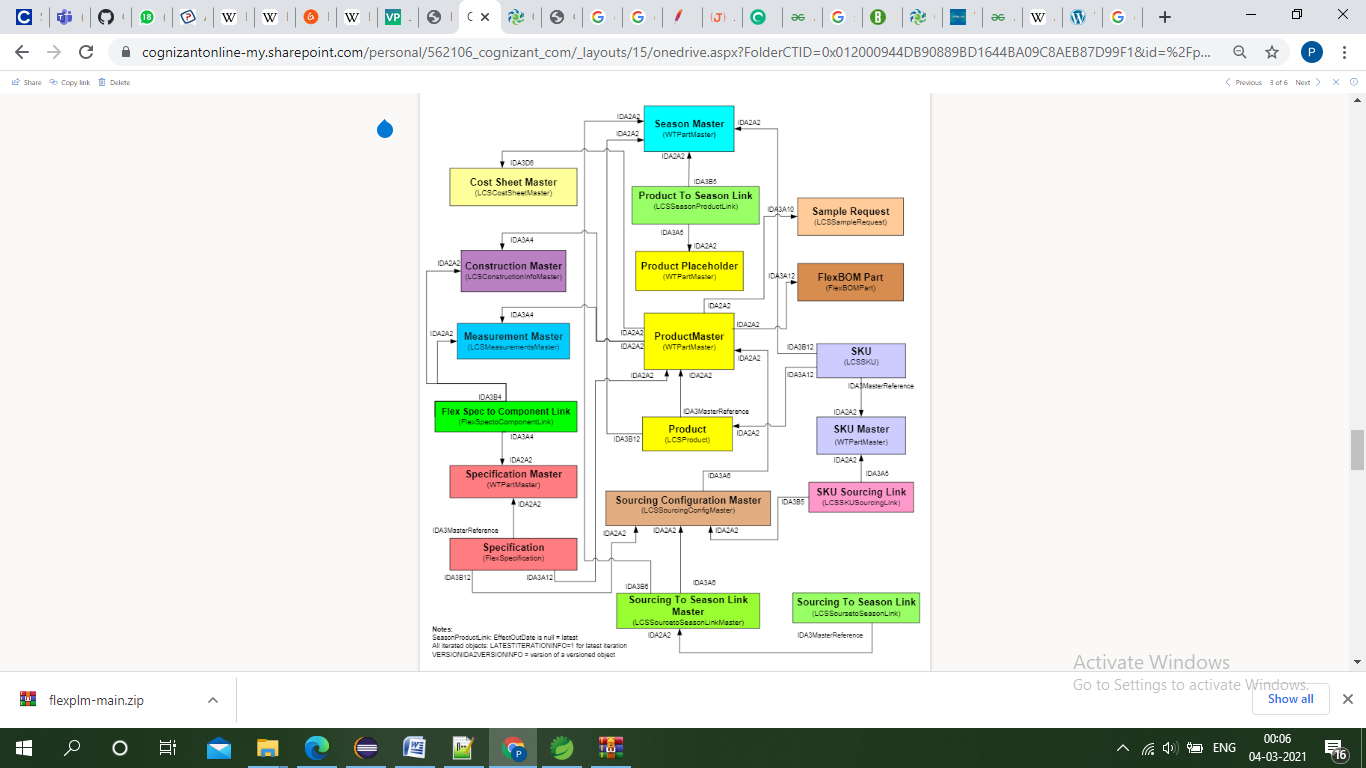
**LOGICAL MODEL:**

Fig 1.Product/SKU-Season Schema (logical model)

**Implementation:**

1. Home Controller:

Spring Controller annotation is a specialization of @Component annotation. Spring Controller annotation is typically used in combination with annotated handler methods based on the Request Mapping annotation.

**Code:**

**package** com.example.demo.controller;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.stereotype.Controller;

**import** org.springframework.web.bind.annotation.RequestMapping;

**import** org.springframework.ui.Model;

**import** org.springframework.ui.ModelMap;

**import** org.springframework.web.bind.annotation.GetMapping;

**import** org.springframework.web.bind.annotation.ModelAttribute;

**import** org.springframework.web.bind.annotation.PostMapping;

**import** org.springframework.web.bind.annotation.RequestMethod;

**import** org.springframework.web.bind.annotation.RequestParam;

**import** org.springframework.web.servlet.ModelAndView;

**import** java.util.ArrayList;

**import** java.util.List;

**import** com.example.demo.model.Store;

**import** com.example.demo.model.\*;

@Controller

**public** **class** HomeController {

**public** String a,b;

String res;

**public** **int** num1;

@Autowired

ReadData rd = **new** ReadData();

@ModelAttribute("tablename")

**public** List<String> tablename()

{

List<String> tablename=**new** ArrayList<String>();

tablename.add("LCSProduct");

tablename.add("WTPartMaster");

tablename.add("LCSSeasonProductLink");

tablename.add("LCSSampleRequest");

tablename.add("FlexBOMPart");

tablename.add("LCSSKU");

tablename.add("FlexSpectoComponentLink");

tablename.add("LCSSKUSourcingLink");

tablename.add("LCSSourcetoSeasonLinK");

tablename.add("LCSSourcetoSeasonLinkMaster");

tablename.add("FlexSpecification");

tablename.add("LCSMeasurementMaster");

tablename.add("LCSConstructionMaster");

tablename.add("LCSCostSheetMaster");

tablename.add("LCSSourcingConfigMaster");

tablename.add("WTPartMaster(SeasonMaster)");

tablename.add("WTPartMaster(ProductPlaceholder)");

tablename.add("WTPPartMaster(SKU Master)");

tablename.add("WTPartMaster(Specification Master)");

**return** tablename;

}

@RequestMapping("/home")

**public** String showHomePage(ModelMap mp)

{

//Store st = new Store();

mp.addAttribute("message",res);

**return** "home";

}

@RequestMapping("/errors")

**public** String details()

{

**return** "enterdata";

}

@GetMapping("/number")

**public** String takeNumber()

{

**return** "number";

}

@PostMapping("/getnumber")

**public** String displayNumber(Model model,@RequestParam **int** number)

{

model.addAttribute("number", number);

num1=number;

**return** "enterdata";

}

@PostMapping("/error")

**public** String error1(Model mo)

{

**return** "enterdata";

}

@PostMapping("/gettabledetails")

**public** String displayformdetails(ModelMap model,@RequestParam String tablename)

{

String[] arr = tablename.split(",");

**if**(arr.length!=num1)

{

model.put("num1", num1);

**return** "error";

}

res=rd.getTableNames(arr);

//System.out.println(tablename);

model.put("result",res);

System.***out***.println(res);

**return** "home";

}

}

1. Read Data:

write to excel file in Java

**Writing excel using POI** is very simple and involve following steps:

1. Create a workbook
2. Create a sheet in workbook
3. Create a row in sheet
4. Add cells in sheet
5. Repeat step 3 and 4 to write more data

**Reading an excel file using POI** is also very simple if we divide this in steps.

1. Create workbook instance from excel sheet
2. Get to the desired sheet
3. Increment row number
4. iterate over all cells in a row
5. repeat step 3 and 4 until all data is read

**Code:**

package com.example.demo.controller;

import java.io.FileInputStream;   
import java.io.FileNotFoundException;   
import java.io.IOException;   
import org.apache.poi.ss.usermodel.Cell;   
import org.apache.poi.ss.usermodel.\*;   
import org.apache.poi.ss.usermodel.Sheet;   
import org.apache.poi.ss.usermodel.Workbook;   
import org.apache.poi.xssf.usermodel.XSSFWorkbook;   
import org.springframework.context.annotation.Configuration;

@Configuration  
public class ReadData   
{   
public static void main(String[] args)   
{   
ReadData rc=new ReadData(); //object of the class   
//reading the value of 2nd row and 2nd column   
  
}

public String getTableNames(String[] arr)  
{  
ReadData rc1=new ReadData();  
String res="<center><table style='border-spacing: 5px;border: 3px solid black;border-collapse: collapse;'>";  
res=res+"<tr><td style='border: 3px solid black;padding: 15px;border-collapse: collapse;'><center><b>Tablenames</b></center></td><td style='padding: 15px;border: 3px solid black;border-collapse: collapse;'><center><b>Connections</b></center></td></tr>";  
int j;  
for(int i=0;i<arr.length-1;i++)  
{  
j=i+1;  
res=res+"<tr>"+"<td style='padding: 15px;border: 3px solid black;border-collapse: collapse;'>"+"<b>"+arr[i]+"</b>"+"&nbspand&nbsp"+"<b>"+arr[j]+"</b>"+"<br>"+"</td>"+"<td style='padding: 15px;border: 3px solid black;border-collapse: collapse;'>"+rc1.ReadCellData(arr[i],arr[j])+"<br>"+"</td>"+"</tr>";  
}  
return res+"</table></center>";  
}

//method defined for reading a cell   
public String ReadCellData(String Row, String Column)   
{   
String[] tables= {"","LCSProduct","WTPartMaster","LCSSeasonProductLink","LCSSampleRequest","FlexBOMPart","LCSSKU","FlexSpectoComponentLink","LCSSKUSourcingLink","LCSSourcetoSeasonLinK","LCSSourcetoSeasonLinkMaster","FlexSpecification","LCSMeasurementMaster","LCSConstructionMaster","LCSCostSheetMaster","LCSSourcingConfigMaster","WTPartMaster(SeasonMaster)","WTPartMaster(ProductPlaceholder)","WTPPartMaster(SKU Master)","WTPartMaster(Specification Master)"};   
String[] tables1= {"","LCSProduct","WTPartMaster","LCSSeasonProductLink","LCSSampleRequest","FlexBOMPart","LCSSKU","LCSSKUSourcingLink","LCSSourcetoSeasonLinkMaster","FlexSpecification","FlexSpectoComponentLink","LCSMeasurementMaster","LCSConstructionMaster","LCSCostSheetMaster","LCSSourcingConfigMaster","LCSSourcetoSeasonLinK","WTPartMaster(SeasonMaster)","WTPartMaster(ProductPlaceholder)","WTPartMaster(Specification Master)","WTPPartMaster(SKU Master)",};

String value=null; //variable for storing the cell value   
Workbook wb=null; //initialize Workbook null   
int vRow=0,vColumn=0;  
int flag=0;  
for(int i=0;i<tables.length;i++)  
{  
if(tables[i].equals(Row))  
{  
vRow=i;  
flag++;  
}  
if(tables1[i].equals(Column))  
{  
vColumn=i;  
flag++;  
}  
if(flag==2)  
{  
break;  
}  
}  
try   
{   
//reading data from a file in the form of bytes   
FileInputStream fis=new FileInputStream("C:\\Users\\ADMIN\\Desktop\\DBModel.xlsx");   
//constructs an XSSFWorkbook object, by buffering the whole stream into the memory   
wb=new XSSFWorkbook(fis);   
}   
catch(FileNotFoundException e)   
{   
e.printStackTrace();   
}   
catch(IOException e1)   
{   
e1.printStackTrace();   
}   
Sheet sheet=wb.getSheetAt(0); //getting the XSSFSheet object at given index   
Row row=sheet.getRow(vRow); //returns the logical row   
Cell cell=row.getCell(vColumn); //getting the cell representing the given column   
value=cell.getStringCellValue(); //getting cell value   
value=value.replace(";",";<br>");  
//System.out.println();  
return value; //returns the cell value   
}   
}

1. **Number:**

System will ask user to enter the number of tables.

**Code:**

<%@ page language=*"java"* contentType=*"text/html; charset=ISO-8859-1"*

pageEncoding=*"ISO-8859-1"*%>

<!DOCTYPE html>

<html>

<head>

<meta charset=*"ISO-8859-1"*>

<title>Get Connection</title> <meta name=*"viewport"* content=*"width=device-width, initial-scale=1"*>

<link rel=*"stylesheet"* href=*"https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.css"*>

<style>

**body** {

background-image: *url('img/disneybg.jpg')*;

background-repeat: *no-repeat*;

background-attachment: *fixed*;

background-size: *100% 100%*;

}

</style>

</head>

<body>

<div class=*"container"* style="margin-top: *150px*;margin-left: *120px*;" >

<form action=*"/getnumber"* method=*"POST"*>

<div style="display: *inline-block*;width: *90%*; text-align:*center*;">

<label style="color: *yellow*; font-size: *25px*; font-family:*Lucida Handwriting*;display: *inline-block*;width: *90%*; text-align:*center*;">Enter number of tables</label>

<input type=*"number"* name=*"number"* >

<button type=*"submit"* class=*"btn btn-primary"*>Submit</button></div>

</form>

</div>

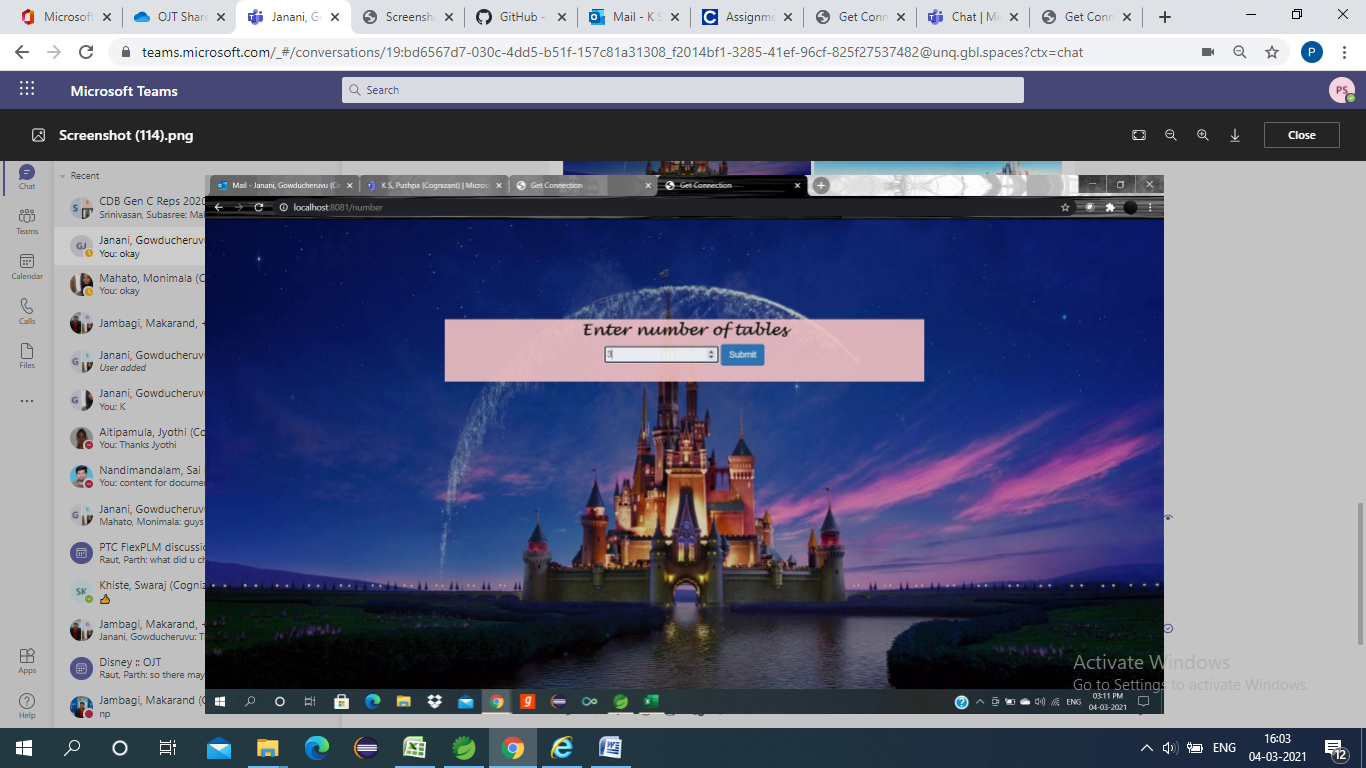
<script src=*"https://ajax.googleapis.com/ajax/libs/jquery/1.12.0/jquery.min.js"*></script>

<script src=*"https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/js/bootstrap.min.js"*></script>

</body>

</html>

**Output:**



1. **Enter data:**

User has to select multiple number of tables.

**Code:**

<%@ page language=*"java"* contentType=*"text/html; charset=ISO-8859-1"*

pageEncoding=*"ISO-8859-1"*%>

<%@ taglib prefix=*"form"* uri=*"http://www.springframework.org/tags/form"* %>

<%@taglib prefix=*"c"* uri=*"http://java.sun.com/jsp/jstl/core"* %>

<html>

<meta charset=*"ISO-8859-1"*>

<title>Get Connection</title> <meta name=*"viewport"* content=*"width=device-width, initial-scale=1"*>

<link rel=*"stylesheet"* href=*"https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.css"*>

<style>

**body** {

background-image: *url('img/disneybg.jpg')*;

background-repeat: *no-repeat*;

background-attachment: *fixed*;

background-size: *100% 100%*;

}

**select** {

width: *300px*;

border: *2px solid #000*;

padding: *0*;

margin: *0*;

align:*center*;

}

**select***:focus* {

min-width: *150px*;

width: *auto*;

}

</style>

</head>

<body>

<div class=*"container"* style="margin-top: *150px*; margin-left: *120px*;">

<form action=*"/gettabledetails"* method=*"post"*>

<label >Select your tables</label>

<select multiple=*"multiple"* name=*"tablename"* style="size: *50 px*">

<c:forEach items=*"*${tablename}*"* var=*"table"*>

<option value=*"*${table}*"*>${table}</option>

</c:forEach>

</select>

<button type=*"submit"*>submit</button>

</form>

</div>

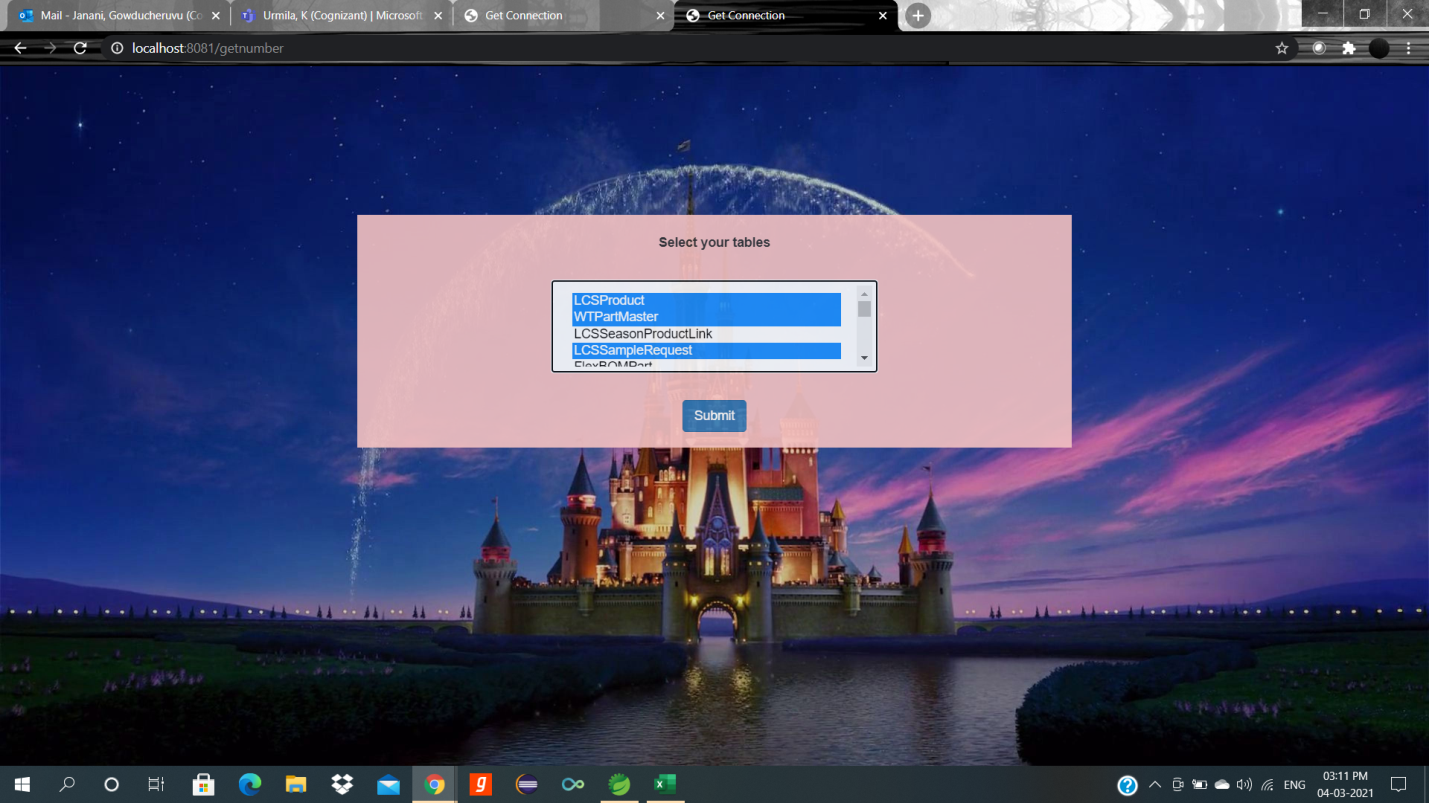
<script src=*"https://ajax.googleapis.com/ajax/libs/jquery/1.12.0/jquery.min.js"*></script>

<script src=*"https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/js/bootstrap.min.js"*></script>

</body>

</html>

**Output:**



1. **Error:**

If user selects wrong number of tables, it will ask to select number of tables based on entered number.

**Code:**

<%@ page language=*"java"* contentType=*"text/html; charset=ISO-8859-1"*

pageEncoding=*"ISO-8859-1"*%>

<!DOCTYPE html>

<html>

<head>

<meta charset=*"ISO-8859-1"*>

<title>Oops!!</title> <meta name=*"viewport"* content=*"width=device-width, initial-scale=1"*>

<link rel=*"stylesheet"* href=*"https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.css"*>

<style>

**body** {

background-image: *url('img/disneybg.jpg')*;

background-repeat: *no-repeat*;

background-attachment: *fixed*;

background-size: *100% 100%*;

}

</style>

</head>

<body>

<div class=*"container"* style="margin-top: *150px*; margin-left: *120px*;">

<form action=*"/errors"* method=*"POST"*>

<div style="display: *inline-block*;width: *90%*; text-align:*center*;">

<p style="color: *blue*; font-size: *25px*; font-family:*Lucida Handwriting*"> Please select ${num1} number of tables...</p>

<button type=*"submit"* class=*"btn btn-primary"*>OK</button>

</div>

</form>

</div>

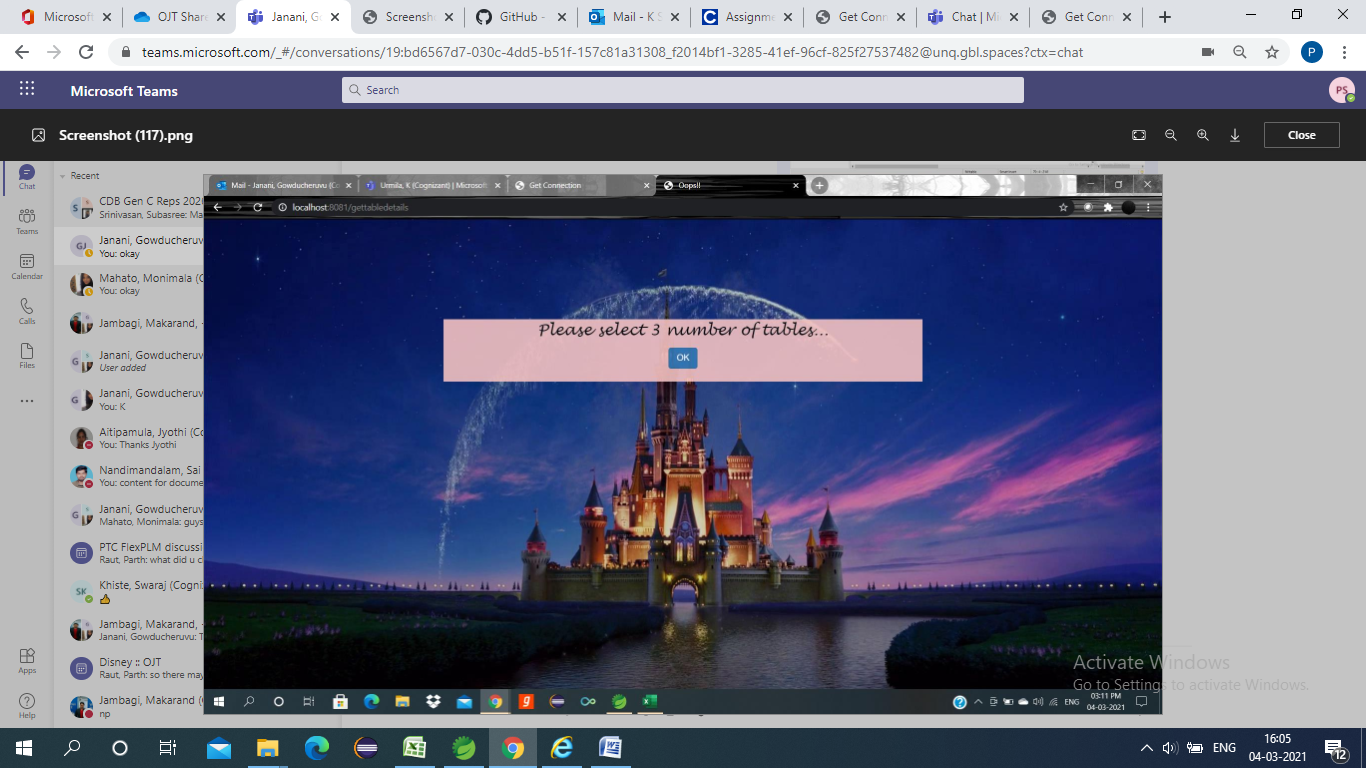
<script src=*"https://ajax.googleapis.com/ajax/libs/jquery/1.12.0/jquery.min.js"*></script>

<script src=*"https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/js/bootstrap.min.js"*></script>

</body>

</html>

**Output:**



1. **Home:**

It will display the data connection between the tables.

**Code:**

<!DOCTYPE html>

<html>

<head>

<meta charset=*"ISO-8859-1"*>

<title>Get Connection</title> <meta name=*"viewport"* content=*"width=device-width, initial-scale=1"*>

<link rel=*"stylesheet"* href=*"https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.css"*>

<style>

**body** {

background-image: *url('img/disneybg.jpg')*;

background-repeat: *no-repeat*;

background-attachment: *fixed*;

background-size: *100% 100%*;

}

</style>

</head>

<body>

<div class=*"container"* style="margin-top: *150px*; margin-left: *120px*;">

<p style="color: *yellow*; font-size: *60px*; font-family:*Lucida Handwriting*">DB Connection </p>

<p style="color: *yellow*; font-size: *25px*; font-family: *Courier New*">Data is ${result}</p>

</div>

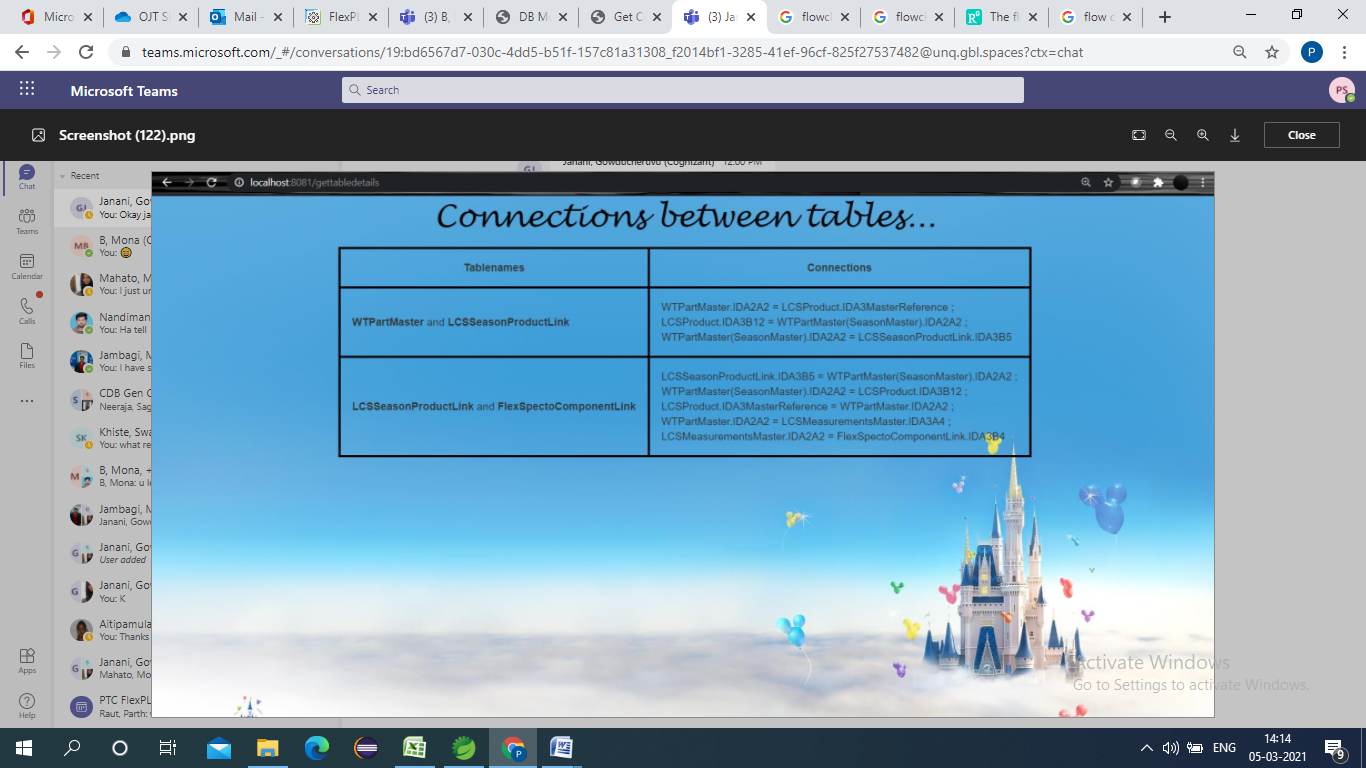
<script src=*"https://ajax.googleapis.com/ajax/libs/jquery/1.12.0/jquery.min.js"*></script>

<script src=*"https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/js/bootstrap.min.js"*></script>

</body>

</html>

**Output:**



**SUMMARY:**

* Storing the data from ER diagram to Excel sheet.
* Retrieving the excel sheet data using java Apache POI into the java class.
* Connecting the classes using Spring and reflecting this data into JSP files by using Spring controller class.
* Starting the local host using tom cat server and running the web page.
* Displaying the connections between the tables in the webpage based on the user requirements.