<Group 9: NinersZone >

Introduction:

Niners Zone is a learning platform designed to provide educators, administrators and learners with a single robust, secure and integrated system to create personalized learning environments. Our project 'Niners Zone' was designed with following functionalities.

As a student he will be able to register, login into the application. Once logged in he will be able to view his courses, academic and assignment materials posted by faculty. Discussion forum allows him to interact with faculty and other students.

As a faculty he will be able to register, login into the application. Once logged in he will be able to register for courses, create assignment and academic posts and add materials. Discussion forum allows him to interact with faculty and other students.

As an admin he will be able to approve the user requests and will be able to approve the course requests raised by the faculty.

Installing the required software:

- **JAVA:**

  Our Project code is written in OOP language JAVA. To make use of the language we need to have the JAVA environment installed called JRE version : 1.8.0_45. The Java compiler and library files comes with Java Development Kit (JDK) version : 1.8.0_40.
  Download Links:
  1. Java: https://java.com/en/download/
  2. JDK: http://www.oracle.com/technetwork/java/javase/downloads/

- **Apache Tomcat 7.0.59:**

  Apache Tomcat is an open source software implementation of the Java Servlet and Java Server Pages technologies. We used tomcat 7 as server to host our application on the localhost.
  Download Link:
  https://tomcat.apache.org/download-70.cgi

- **MySQL:**

  MySQL is the world's most popular open source database. With its proven performance, reliability and ease-of-use, MySQL has become the leading database choice for web-based applications. We used MySQL Community Server database version: 5.6.24 for our project.
  Download Link:
  http://dev.mysql.com/downloads/mysql/

- **JUnit:**

  JUnit is a simple framework to write repeatable tests. It is an instance of the xUnit architecture for unit testing frameworks. We have included junit .jar file into library of our project.
  Download Link:
  https://github.com/junit-team/junit/wiki/Download-and-Install

- Amazon Web Services (AWS):

  We have used AWS to upload files to the bucket. Once the user is in need of downloading he can directly use the s3 bucket link to download the file. All the necessary AWS jar files are in the 'WebContent/WEB-INF/lib' folder of the root directory. For more information on using the AWS services one could visit the below site.
  http://docs.aws.amazon.com/AmazonS3/latest/gsg/GetStartedWithS3.html

  To Download the AWS SDK use the below link.
  http://aws.amazon.com/sdk-for-java/

  One has to make changes to 'AmazonUploadServlet.java', in this file one has to provide the access key and secret key, which Amazon provides once they create an account with Amazon. The servers contents currently uploaded from a default location '/Users/rohan/Documents/Test/' which has to be changed based on actual path of the server where the application has been deployed. Once the above changes have been made the upload functionality of the application will start working. For more documentation refer the comments in each section of the servlet.

- JAR Files

  All the necessary jar files have been listed below and they are located in the 'lib' folder as well as in 'WebContent/WEB-INF/lib'

lib/

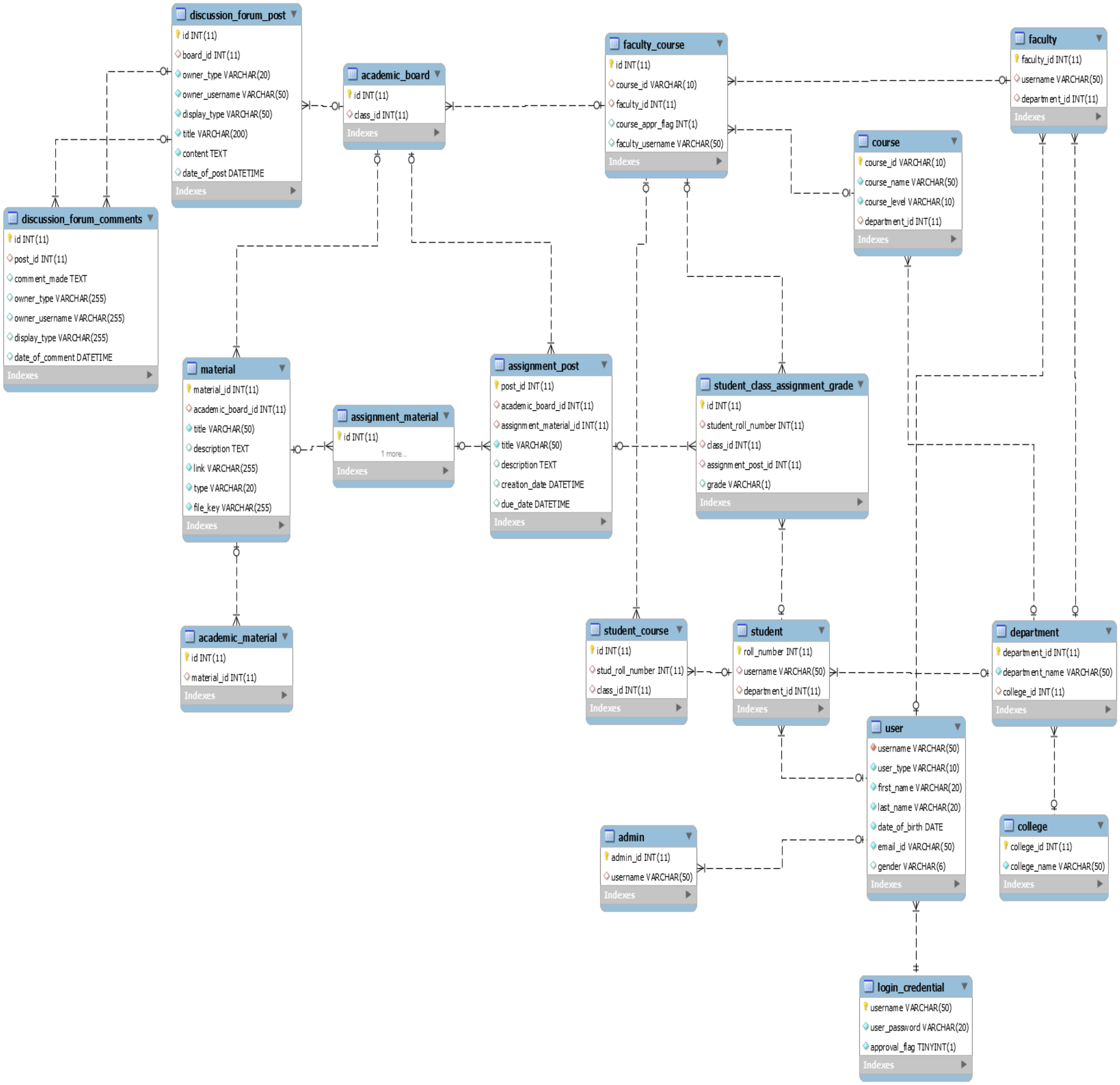
```
├── annotations-api.jar
├── catalina-ant.jar
├── catalina-ha.jar
├── catalina-tribes.jar
├── catalina.jar
├── ecj-4.4.jar
├── el-api.jar
├── jasper-el.jar
├── jasper.jar
├── jsp-api.jar
├── portal-kernel-5.2.3.jar
├── servlet-api.jar
├── tomcat-api.jar
├── tomcat-coyote.jar
├── tomcat-dbcp.jar
├── tomcat-i18n-es.jar
├── tomcat-i18n-fr.jar
├── tomcat-i18n-ja.jar
├── tomcat-jdbc.jar
├── tomcat-util.jar
├── tomcat7-websocket.jar
└── websocket-api.jar
```

WebContent/WEB-INF/lib/

```
├── aspectjrt.jar
├── aspectjweaver.jar
├── aws-java-sdk-1.9.28.1-javadoc.jar
├── aws-java-sdk-1.9.28.1-sources.jar
├── aws-java-sdk-1.9.28.1.jar
├── aws-java-sdk-flow-build-tools-1.9.28.1.jar
├── commons-codec-1.6.jar
├── commons-fileupload-1.3.1.jar
├── commons-io-2.4-javadoc.jar
├── commons-io-2.4-sources.jar
├── commons-io-2.4-test-sources.jar
├── commons-io-2.4-tests.jar
├── commons-io-2.4.jar
├── commons-logging-1.1.3.jar
├── cos.jar
├── freemarker-2.3.18.jar
├── gson-2.3.1.jar
├── httpclient-4.3.jar
├── httpcore-4.3.jar
├── jackson-annotations-2.3.0.jar
├── jackson-core-2.3.2.jar
├── jackson-databind-2.3.2.jar
├── javax.mail-api-1.4.6.jar
├── joda-time-2.2.jar
├── jstl-1.2.jar
├── mysql-connector-java-5.1.34-bin.jar
├── org.json-20120521.jar
├── spring-beans-3.0.7.jar
├── spring-context-3.0.7.jar
└── spring-core-3.0.7.jar
```

Database Schema:

The schema of the database is as given below.



For further reference we are attaching the MySQL Workbench file 'ninerZone.mwb' in the zip folder, one can interactively view the relations between the tables used in the application. (Provided the user has MySQL Workbench installed).
Download Link:

https://dev.mysql.com/downloads/workbench/

Setting Up the Database:

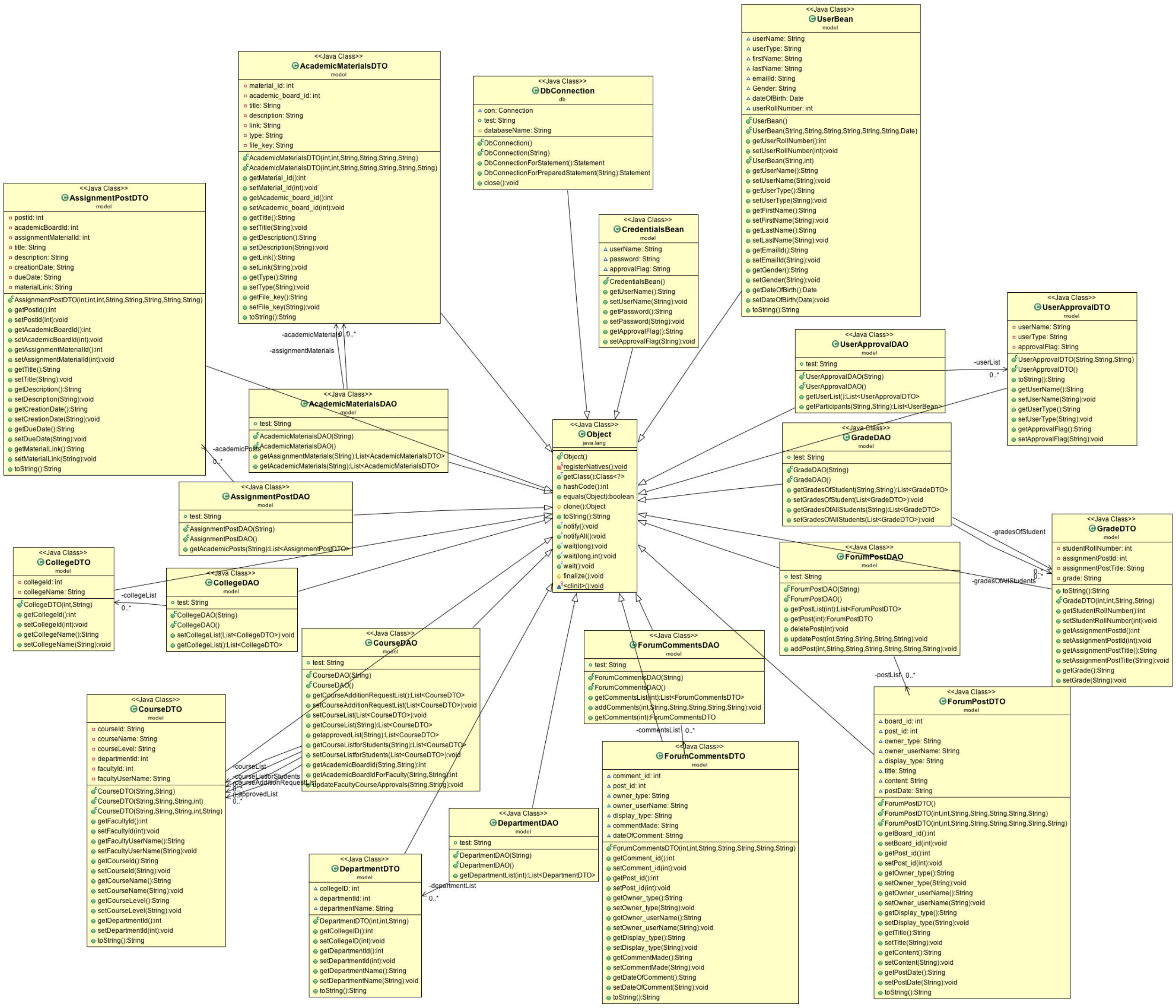
We are attaching the mysql dump file. If mysql is in the path variable then, run the below commands to load the mysql dump and get started.

```
mysql -u <username> -p<password> <database name>  < <dump file with absolute path>
```

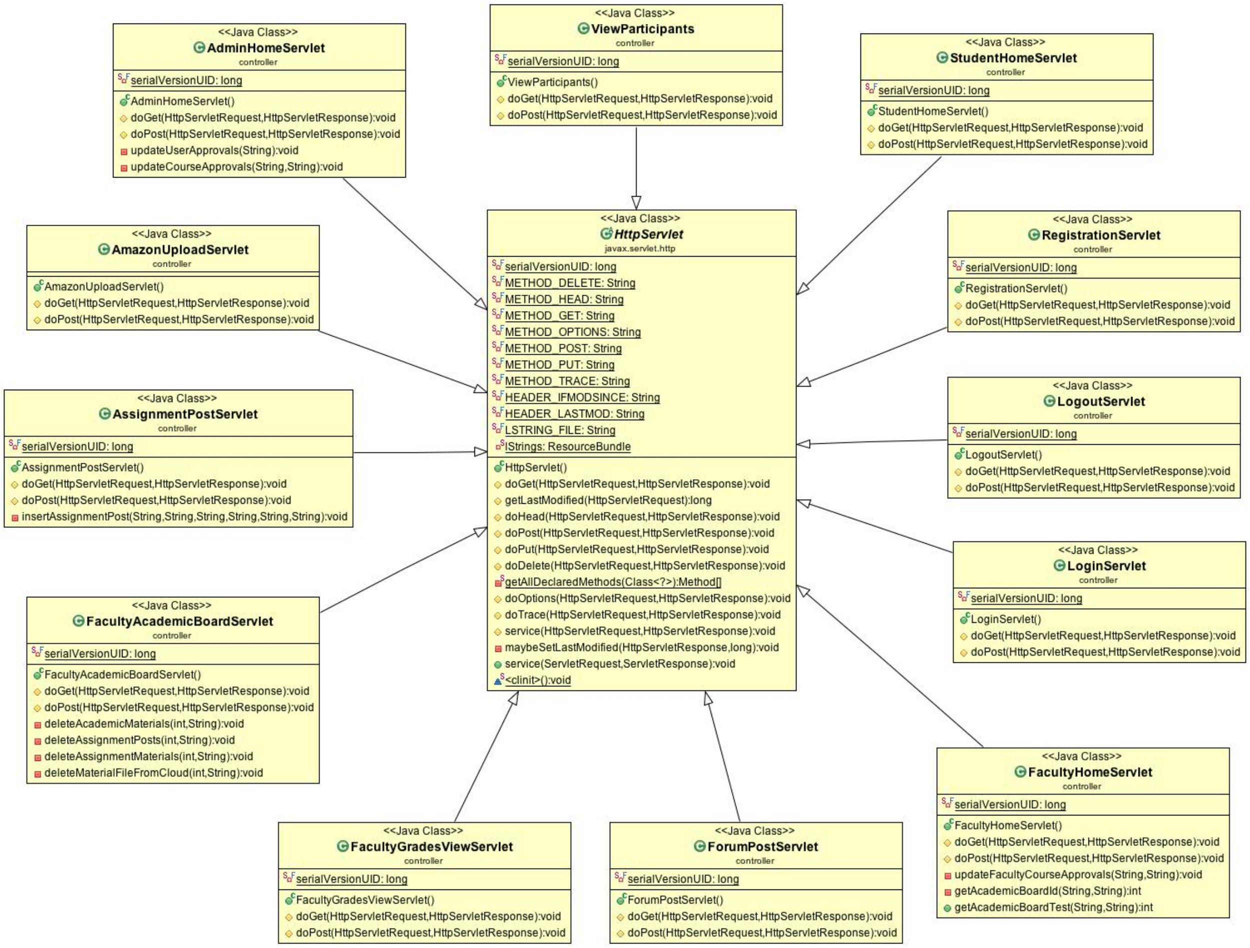
Make sure the tables and contents are loaded into the database. Once this has been done minor changes to the java file 'src/db/DbConnection.java'. The connection string has to updated, the username and password for the database has to be updated in the java file. The 'ssdi_final.sql' has been attached in the zip file.

Class diagram:

Model:



Controller:


<<Java Class>>
AdminHomeServlet
controller
serialVersionUID: long
AdminHomeServlet()
doGet(HttpServletRequest,HttpServletResponse):void
doPost(HttpServletRequest,HttpServletResponse):void
updateUserApprovals(String):void
updateCourseApprovals(String,String):void
<<Java Class>>
ViewParticipants
controller
serialVersionUID: long
ViewParticipants()
doGet(HttpServletRequest,HttpServletResponse):void
doPost(HttpServletRequest,HttpServletResponse):void
<<Java Class>>
StudentHomeServlet
controller
serialVersionUID: long
StudentHomeServlet()
doGet(HttpServletRequest,HttpServletResponse):void
doPost(HttpServletRequest,HttpServletResponse):void
<<Java Class>>
AmazonUploadServlet
controller
AmazonUploadServlet()
doGet(HttpServletRequest,HttpServletResponse):void
doPost(HttpServletRequest,HttpServletResponse):void
<<Java Class>>
HttpServlet
javax.servlet.http
serialVersionUID: long
METHOD_DELETE: String
METHOD_HEAD: String
METHOD_GET: String
METHOD_OPTIONS: String
METHOD_POST: String
METHOD_PUT: String
METHOD_TRACE: String
HEADER_IFMODSINCE: String
HEADER_LASTMOD: String
LSTRING_FILE: String
lStrings: ResourceBundle
HttpServlet()
doGet(HttpServletRequest,HttpServletResponse):void
getLastModified(HttpServletRequest):long
doHead(HttpServletRequest,HttpServletResponse):void
doPost(HttpServletRequest,HttpServletResponse):void
doPut(HttpServletRequest,HttpServletResponse):void
doDelete(HttpServletRequest,HttpServletResponse):void
getAllDeclaredMethods(Class<?>):Method[]
doOptions(HttpServletRequest,HttpServletResponse):void
doTrace(HttpServletRequest,HttpServletResponse):void
service(HttpServletRequest,HttpServletResponse):void
maybeSetLastModified(HttpServletResponse,long):void
service(ServletRequest,ServletResponse):void
<clinit>():void
<<Java Class>>
RegistrationServlet
controller
serialVersionUID: long
RegistrationServlet()
doGet(HttpServletRequest,HttpServletResponse):void
doPost(HttpServletRequest,HttpServletResponse):void
<<Java Class>>
AssignmentPostServlet
controller
serialVersionUID: long
AssignmentPostServlet()
doGet(HttpServletRequest,HttpServletResponse):void
doPost(HttpServletRequest,HttpServletResponse):void
insertAssignmentPost(String,String,String,String,String,String):void
<<Java Class>>
LogoutServlet
controller
serialVersionUID: long
LogoutServlet()
doGet(HttpServletRequest,HttpServletResponse):void
doPost(HttpServletRequest,HttpServletResponse):void
<<Java Class>>
LoginServlet
controller
serialVersionUID: long
LoginServlet()
doGet(HttpServletRequest,HttpServletResponse):void
doPost(HttpServletRequest,HttpServletResponse):void
<<Java Class>>
FacultyAcademicBoardServlet
controller
serialVersionUID: long
FacultyAcademicBoardServlet()
doGet(HttpServletRequest,HttpServletResponse):void
doPost(HttpServletRequest,HttpServletResponse):void
deleteAcademicMaterials(int,String):void
deleteAssignmentPosts(int,String):void
deleteAssignmentMaterials(int,String):void
deleteMaterialFileFromCloud(int,String):void
<<Java Class>>
FacultyHomeServlet
controller
serialVersionUID: long
FacultyHomeServlet()
doGet(HttpServletRequest,HttpServletResponse):void
doPost(HttpServletRequest,HttpServletResponse):void
updateFacultyCourseApprovals(String,String):void
getAcademicBoardId(String,String):int
getAcademicBoardTest(String,String):int
<<Java Class>>
FacultyGradesViewServlet
controller
serialVersionUID: long
FacultyGradesViewServlet()
doGet(HttpServletRequest,HttpServletResponse):void
doPost(HttpServletRequest,HttpServletResponse):void
<<Java Class>>
ForumPostServlet
controller
serialVersionUID: long
ForumPostServlet()
doGet(HttpServletRequest,HttpServletResponse):void
doPost(HttpServletRequest,HttpServletResponse):void

***Note:** All the images and attachments used in the document are in 'assets' folder*