Enterprise Application Development

(IDS 517)

Design Document

for

Online Test Application

'White Board'

Group Name - F16G319

- Mrinal Dhawan (mdhawa3@uic.edu)
- Amey Pophali (apopha2@uic.edu)

Table of Contents

Version History

Version	Author	Date	Comments
0.1	Amey Pophali	20 Sep 2016	First draft created
1.0	Mrinal Dhawan	28 Sep 2016	Draft version upgraded for circulation
1.1	Amey Pophali/Mrinal Dhawan	30 Oct 2016	Added UML diagrams and miscellaneous updates
1.2	Amey Pophali/Mrinal Dhawan	23 Nov 2016	Updated Relational Schema and added additional UML

- **About the Application**

'White Board' is an Online Test management tool for students and instructors. It provides students an ability to take a test and provides a detailed feedback on student's performance. The tool also enables instructors to create a new online assessment.

Additionally, it also provides the instructor to check for a visual graph analysis of performances of students in assessments.

- **Low Level Requirements**

The objective of the project is to create an online test application. The low level requirements for the projects are as follows.

1. Develop a login page for the web based online test application such that only authorized users could login.
2. Create tables in the back end Relational Database to save important details in relational format. The tables to be introduced as a part of this application are mentioned in the document.
3. To Design and create a Homepage for the application. The homepage will contain user's and programmer's guide links. Clicking on the link will direct the user to the respective documentations.
4. Provide the user with an ability to 'Logout' from the application. Logging out will end the particular user's current active sessions and database links.
5. Ability to upload CSV data into a relational database table. The application will provide the ability for the instructors to generate a CSV file with data from the Database.
6. The ability to load data from the database to the CSV file will also be provided for the person with Instructor (Administrator) access.
7. Analyze and display data with range select and sort capability.
8. Generate numeric and graphic reports with range select and sort capability.
9. The student homepage will give the student option to select from multiple tests and take one of the tests.
10. The feedback of the test will be provided to the students only when the test has completed.
11. To enable the instructor to create a new user, if required.

- **Alternative Approaches to Implementing and Deploying Enterprise Application over Internet**

➢ **Alternative 1:**

One of the approaches to the implementation of an enterprise application over the internet is using Web Application Archive file (.war file). We can package the application as a WAR file using an IDE like Eclipse.

Tomcat is a webserver that can host web applications. $CATALINA environmental variable points to the directory where Tomcat has been installed on a machine. The WAR file is copied into $CATALINA_HOME\webapps directory. The way forward would be to restart the server. The WAR file gets unpacked and application is deployed.

➢ **Alternative 2:**

Another alternative would be to use Amazon Web Services(AWS) for the purpose. The application could be developed and then implemented to a cloud. Going forward, we can use various AWS deployment services that give us right combination of control and automation.

- **Model View Controller (MVC) Pattern for the Test Application**

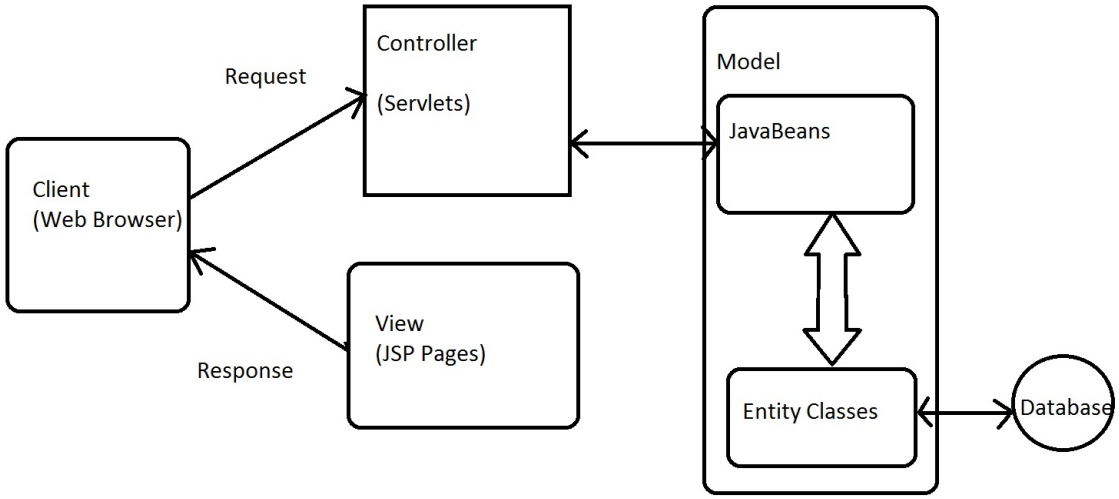
We use Model-View-Controller(MVC) Design pattern for the application. The terminologies in MVC stand for

- **Model**
  It represents a Java object carrying data. The Model class in the context of our application will consists of the data of the users of the application.

- **View**
  Represents the visualization of the data that the Model contains. The View class for the application will consist of the methods that display the data of the users of the application as per request. This component has various JSP files that help users navigate through the online application.

- **Controller**
  This is a class that operates on Model and View classes and controls the data flow. The Java classes would contain the business logic of the application.

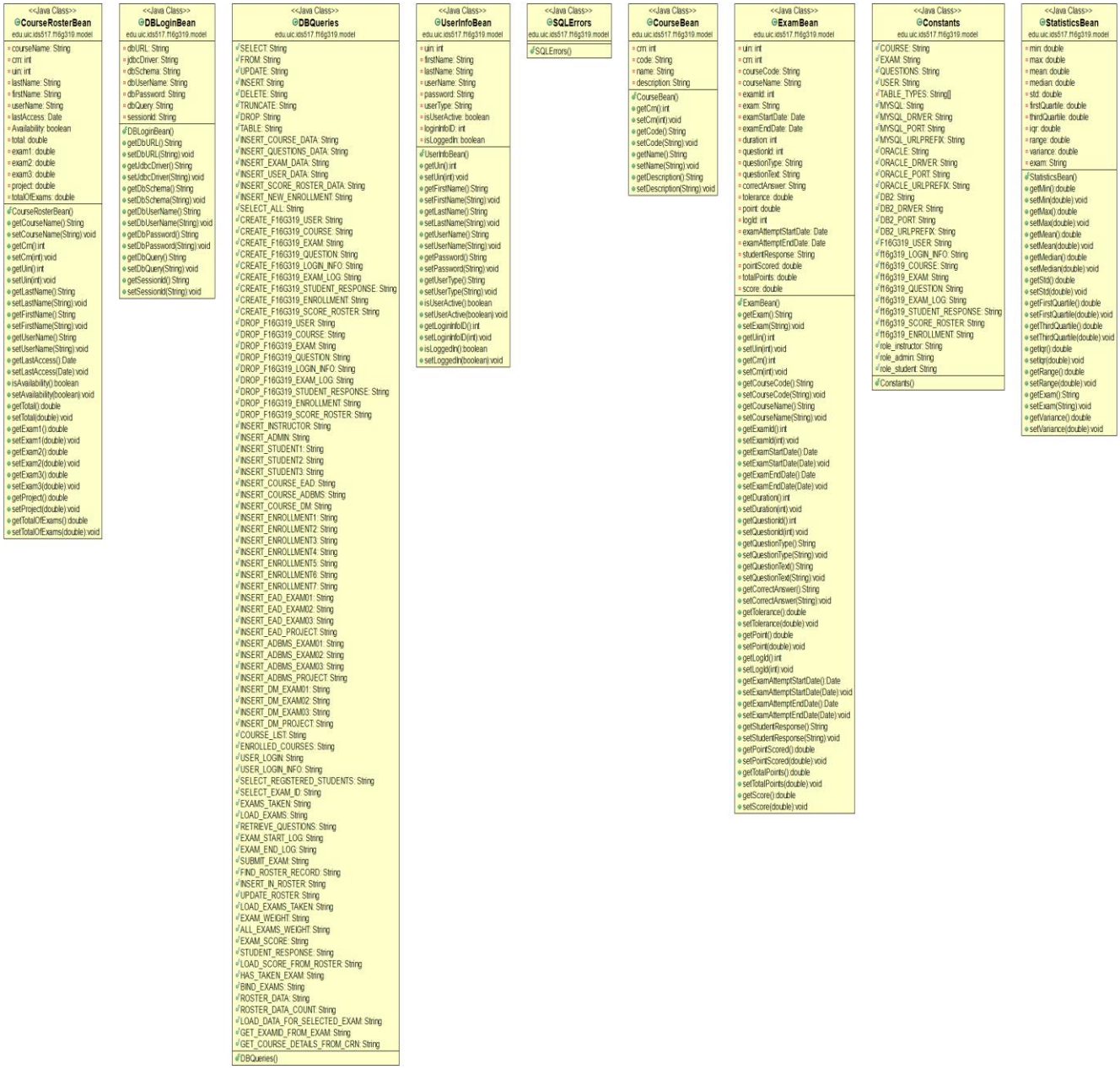
➢ **Flow of Data in the Application (MVC Model)**

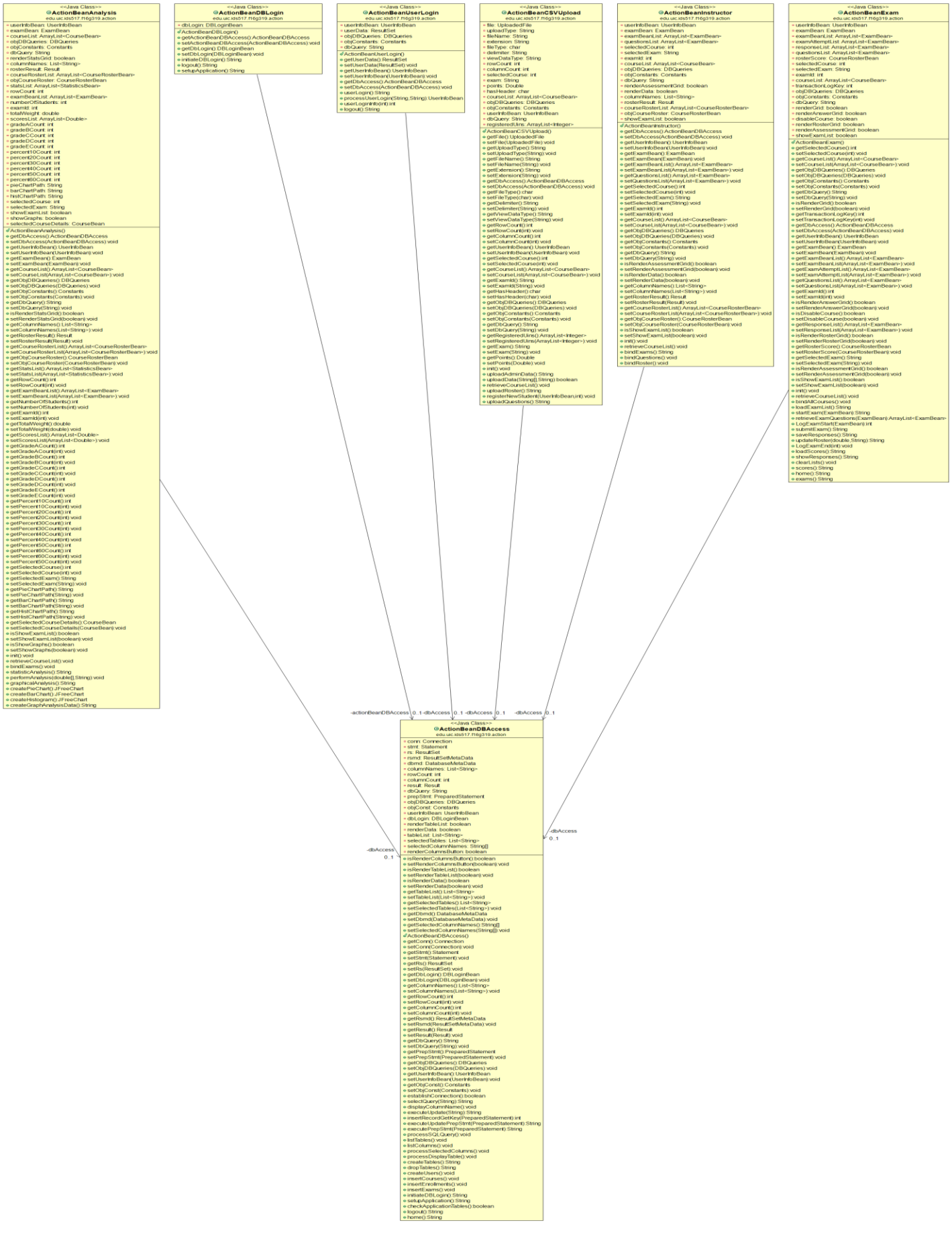


A graphical representation of the MVC model implemented for the application is shown above. The diagram provides a graphical representation of data flow in the application.

- **Classes, JSP and UML Diagrams**

The Java beans and their UML diagrams are shown as –


<<Java Class>>
CourseRosterBean
edu.uic.ids517.f16g319.model
courseName: String
crn: int
uin: int
lastName: String
firstName: String
userName: String
lastAccess: Date
Availability: boolean
total: double
exam1: double
exam2: double
exam3: double
project: double
totalOfExams: double
CourseRosterBean()
getCourseName():String
setCourseName(String):void
getCrn():int
setCrn(int):void
getUin():int
setUin(int):void
getLastName():String
setLastName(String):void
getFirstName():String
setFirstName(String):void
getUserName():String
setUserName(String):void
getLastAccess():Date
setLastAccess(Date):void
isAvailability():boolean
setAvailability(boolean):void
getTotal():double
setTotal(double):void
getExam1():double
setExam1(double):void
getExam2():double
setExam2(double):void
getExam3():double
setExam3(double):void
getProject():double
setProject(double):void
getTotalOfExams():double
setTotalOfExams(double):void
<<Java Class>>
DBLoginBean
edu.uic.ids517.f16g319.model
dbURL: String
jdbcDriver: String
dbSchema: String
dbUserName: String
dbPassword: String
dbQuery: String
sessionId: String
DBLoginBean()
getDbURL():String
setDbURL(String):void
getJdbcDriver():String
setJdbcDriver(String):void
getDbSchema():String
setDbSchema(String):void
getDbUserName():String
setDbUserName(String):void
getDbPassword():String
setDbPassword(String):void
getDbQuery():String
setDbQuery(String):void
getSessionId():String
setSessionId(String):void
<<Java Class>>
DBQueries
edu.uic.ids517.f16g319.model
SELECT: String
FROM: String
UPDATE: String
INSERT: String
DELETE: String
TRUNCATE: String
DROP: String
TABLE: String
INSERT_COURSE_DATA: String
INSERT_QUESTIONS_DATA: String
INSERT_EXAM_DATA: String
INSERT_USER_DATA: String
INSERT_SCORE_ROSTER_DATA: String
INSERT_NEW_ENROLLMENT: String
SELECT_ALL: String
CREATE_F16G319_USER: String
CREATE_F16G319_COURSE: String
CREATE_F16G319_EXAM: String
CREATE_F16G319_QUESTION: String
CREATE_F16G319_LOGIN_INFO: String
CREATE_F16G319_EXAM_LOG: String
CREATE_F16G319_STUDENT_RESPONSE: String
CREATE_F16G319_ENROLLMENT: String
CREATE_F16G319_SCORE_ROSTER: String
DROP_F16G319_USER: String
DROP_F16G319_COURSE: String
DROP_F16G319_EXAM: String
DROP_F16G319_QUESTION: String
DROP_F16G319_LOGIN_INFO: String
DROP_F16G319_EXAM_LOG: String
DROP_F16G319_STUDENT_RESPONSE: String
DROP_F16G319_ENROLLMENT: String
DROP_F16G319_SCORE_ROSTER: String
INSERT_INSTRUCTOR: String
INSERT_ADMIN: String
INSERT_STUDENT1: String
INSERT_STUDENT2: String
INSERT_STUDENT3: String
INSERT_COURSE_EAD: String
INSERT_COURSE_ADBMS: String
INSERT_COURSE_DM: String
INSERT_ENROLLMENT1: String
INSERT_ENROLLMENT2: String
INSERT_ENROLLMENT3: String
INSERT_ENROLLMENT4: String
INSERT_ENROLLMENT5: String
INSERT_ENROLLMENT6: String
INSERT_ENROLLMENT7: String
INSERT_EAD_EXAM01: String
INSERT_EAD_EXAM02: String
INSERT_EAD_EXAM03: String
INSERT_EAD_PROJECT: String
INSERT_ADBMS_EXAM01: String
INSERT_ADBMS_EXAM02: String
INSERT_ADBMS_EXAM03: String
INSERT_ADBMS_PROJECT: String
INSERT_DM_EXAM01: String
INSERT_DM_EXAM02: String
INSERT_DM_EXAM03: String
INSERT_DM_PROJECT: String
COURSE_LIST: String
ENROLLED_COURSES: String
USER_LOGIN: String
USER_LOGIN_INFO: String
SELECT_REGISTERED_STUDENTS: String
SELECT_EXAM_ID: String
EXAMS_TAKEN: String
LOAD_EXAMS: String
RETRIEVE_QUESTIONS: String
EXAM_START_LOG: String
EXAM_END_LOG: String
SUBMIT_EXAM: String
FIND_ROSTER_RECORD: String
INSERT_IN_ROSTER: String
UPDATE_ROSTER: String
LOAD_EXAMS_TAKEN: String
EXAM_WEIGHT: String
ALL_EXAMS_WEIGHT: String
EXAM_SCORE: String
STUDENT_RESPONSE: String
LOAD_SCORE_FROM_ROSTER: String
HAS_TAKEN_EXAM: String
BIND_EXAMS: String
ROSTER_DATA: String
ROSTER_DATA_COUNT: String
LOAD_DATA_FOR_SELECTED_EXAM: String
GET_EXAMID_FROM_EXAM: String
GET_COURSE_DETAILS_FROM_CRN: String
DBQueries()
<<Java Class>>
UserInfoBean
edu.uic.ids517.f16g319.model
uin: int
firstName: String
lastName: String
userName: String
password: String
userType: String
isUserActive: boolean
loginInfoID: int
isLoggedIn: boolean
UserInfoBean()
getUin():int
setUin(int):void
getFirstName():String
setFirstName(String):void
getLastName():String
setLastName(String):void
getUserName():String
setUserName(String):void
getPassword():String
setPassword(String):void
getUserType():String
setUserType(String):void
isUserActive():boolean
setUserActive(boolean):void
getLoginInfoID():int
setLoginInfoID(int):void
isLoggedIn():boolean
setLoggedIn(boolean):void
<<Java Class>>
SQLErrors
edu.uic.ids517.f16g319.model
SQLErrors()
<<Java Class>>
CourseBean
edu.uic.ids517.f16g319.model
crn: int
code: String
name: String
description: String
CourseBean()
getCrn():int
setCrn(int):void
getCode():String
setCode(String):void
getName():String
setName(String):void
getDescription():String
setDescription(String):void
<<Java Class>>
ExamBean
edu.uic.ids517.f16g319.model
uin: int
crn: int
courseCode: String
courseName: String
examId: int
exam: String
examStartDate: Date
examEndDate: Date
duration: int
questionId: int
questionType: String
questionText: String
correctAnswer: String
tolerance: double
point: double
logId: int
examAttemptStartDate: Date
examAttemptEndDate: Date
studentResponse: String
pointScored: double
totalPoints: double
score: double
ExamBean()
getExam():String
setExam(String):void
getUin():int
setUin(int):void
getCrn():int
setCrn(int):void
getCourseCode():String
setCourseCode(String):void
getCourseName():String
setCourseName(String):void
getExamId():int
setExamId(int):void
getExamStartDate():Date
setExamStartDate(Date):void
getExamEndDate():Date
setExamEndDate(Date):void
getDuration():int
setDuration(int):void
getQuestionId():int
setQuestionId(int):void
getQuestionType():String
setQuestionType(String):void
getQuestionText():String
setQuestionText(String):void
getCorrectAnswer():String
setCorrectAnswer(String):void
getTolerance():double
setTolerance(double):void
getPoint():double
setPoint(double):void
getLogId():int
setLogId(int):void
getExamAttemptStartDate():Date
setExamAttemptStartDate(Date):void
getExamAttemptEndDate():Date
setExamAttemptEndDate(Date):void
getStudentResponse():String
setStudentResponse(String):void
getPointScored():double
setPointScored(double):void
getTotalPoints():double
setTotalPoints(double):void
getScore():double
setScore(double):void
<<Java Class>>
Constants
edu.uic.ids517.f16g319.model
COURSE: String
EXAM: String
QUESTIONS: String
USER: String
TABLE_TYPES: String[]
MYSQL: String
MYSQL_DRIVER: String
MYSQL_PORT: String
MYSQL_URLPREFIX: String
ORACLE: String
ORACLE_DRIVER: String
ORACLE_PORT: String
ORACLE_URLPREFIX: String
DB2: String
DB2_DRIVER: String
DB2_PORT: String
DB2_URLPREFIX: String
F16G319_USER: String
f16g319_LOGIN_INFO: String
f16g319_COURSE: String
f16g319_EXAM: String
f16g319_QUESTION: String
f16g319_EXAM_LOG: String
f16g319_STUDENT_RESPONSE: String
f16g319_SCORE_ROSTER: String
f16g319_ENROLLMENT: String
role_instructor: String
role_admin: String
role_student: String
Constants()
<<Java Class>>
StatisticsBean
edu.uic.ids517.f16g319.model
min: double
max: double
mean: double
median: double
std: double
firstQuartile: double
thirdQuartile: double
iqr: double
range: double
variance: double
exam: String
StatisticsBean()
getMin():double
setMin(double):void
getMax():double
setMax(double):void
getMean():double
setMean(double):void
getMedian():double
setMedian(double):void
getStd():double
setStd(double):void
getFirstQuartile():double
setFirstQuartile(double):void
getThirdQuartile():double
setThirdQuartile(double):void
getIqr():double
setIqr(double):void
getRange():double
setRange(double):void
getExam():String
setExam(String):void
getVariance():double
setVariance(double):void


«Java Class»
ActionBeanAnalysis
edu.uic.ids517.fl6g319.action
+ userInfoBean: UserInfoBean
+ examBean: ExamBean
+ courseList: ArrayList<CourseBean>
+ objDBQueries: DBQueries
+ objConstants: Constants
+ dbQuery: String
+ renderStatsGrid: boolean
+ columnNames: List<String>
+ rosterResult: Result
+ courseRosterList: ArrayList<CourseRosterBean>
+ objCourseRoster: CourseRosterBean
+ statsList: ArrayList<StatisticsBean>
+ rowCount: int
+ examBeanList: ArrayList<ExamBean>
+ numberOfStudents: int
+ examId: int
+ totalWeight: double
+ scoresList: ArrayList<Double>
+ gradeACount: int
+ gradeBCount: int
+ gradeCCount: int
+ gradeDCount: int
+ gradeECount: int
+ percent10Count: int
+ percent20Count: int
+ percent30Count: int
+ percent40Count: int
+ percent50Count: int
+ percent60Count: int
+ pieChartPath: String
+ barChartPath: String
+ histChartPath: String
+ selectedCourse: int
+ selectedExam: String
+ showExamList: boolean
+ showGraphs: boolean
+ selectedCourseDetails: CourseBean
# ActionBeanAnalysis()
+ getDbAccess():ActionBeanDBAccess
+ setDbAccess(ActionBeanDBAccess):void
+ getUserInfoBean():UserInfoBean
+ setUserInfoBean(UserInfoBean):void
+ getExamBean():ExamBean
+ setExamBean(ExamBean):void
+ getCourseList():ArrayList<CourseBean>
+ setCourseList(ArrayList<CourseBean>):void
+ getObjDBQueries():DBQueries
+ setObjDBQueries(DBQueries):void
+ getObjConstants():Constants
+ setObjConstants(Constants):void
+ getDbQuery():String
+ setDbQuery(String):void
+ isRenderStatsGrid():boolean
+ setRenderStatsGrid(boolean):void
+ getColumnNames():List<String>
+ setColumnNames(List<String>):void
+ getRosterResult():Result
+ setRosterResult(Result):void
+ getCourseRosterList():ArrayList<CourseRosterBean>
+ setCourseRosterList(ArrayList<CourseRosterBean>):void
+ getObjCourseRoster():CourseRosterBean
+ setObjCourseRoster(CourseRosterBean):void
+ getStatsList():ArrayList<StatisticsBean>
+ setStatsList(ArrayList<StatisticsBean>):void
+ getRowCount():int
+ setRowCount(int):void
+ getExamBeanList():ArrayList<ExamBean>
+ setExamBeanList(ArrayList<ExamBean>):void
+ getNumberOfStudents():int
+ setNumberOfStudents(int):void
+ getExamId():int
+ setExamId(int):void
+ getTotalWeight():double
+ setTotalWeight(double):void
+ getScoresList():ArrayList<Double>
+ setScoresList(ArrayList<Double>):void
+ getGradeACount():int
+ setGradeACount(int):void
+ getGradeBCount():int
+ setGradeBCount(int):void
+ getGradeCCount():int
+ setGradeCCount(int):void
+ getGradeDCount():int
+ setGradeDCount(int):void
+ getGradeECount():int
+ setGradeECount(int):void
+ getPercent10Count():int
+ setPercent10Count(int):void
+ getPercent20Count():int
+ setPercent20Count(int):void
+ getPercent30Count():int
+ setPercent30Count(int):void
+ getPercent40Count():int
+ setPercent40Count(int):void
+ getPercent50Count():int
+ getPercent60Count():int
+ setPercent60Count(int):void
+ setPercent50Count(int):void
+ getSelectedCourse():int
+ setSelectedCourse(int):void
+ getSelectedExam():String
+ setSelectedExam(String):void
+ getPieChartPath():String
+ setPieChartPath(String):void
+ getBarChartPath():String
+ setBarChartPath(String):void
+ getHistChartPath():String
+ setHistChartPath(String):void
+ getSelectedCourseDetails():CourseBean
+ setSelectedCourseDetails(CourseBean):void
+ isShowExamList():boolean
+ setShowExamList(boolean):void
+ isShowGraphs():boolean
+ setShowGraphs(boolean):void
+ init():void
+ retrieveCourseList():void
+ bindExams():void
+ statisticAnalysis():String
+ performAnalysis(double[],String):void
+ graphicalAnalysis():String
+ createPieChart():JFreeChart
+ createBarChart():JFreeChart
+ createHistogram():JFreeChart
+ createGraphAnalysisData():String
«Java Class»
ActionBeanDBLogin
edu.uic.ids517.fl6g319.action
+ dbLogin: DBLoginBean
# ActionBeanDBLogin()
+ getActionBeanDBAccess():ActionBeanDBAccess
+ setActionBeanDBAccess(ActionBeanDBAccess):void
+ getDbLogin():DBLoginBean
+ setDbLogin(DBLoginBean):void
+ initiateDBLogin():String
+ logout():String
+ setupApplication():String
«Java Class»
ActionBeanUserLogin
edu.uic.ids517.fl6g319.action
+ userInfoBean: UserInfoBean
+ userData: ResultSet
+ objDBQueries: DBQueries
+ objConstants: Constants
+ dbQuery: String
# ActionBeanUserLogin()
+ getUserData():ResultSet
+ setUserData(ResultSet):void
+ getUserInfoBean():UserInfoBean
+ setUserInfoBean(UserInfoBean):void
+ getDbAccess():ActionBeanDBAccess
+ setDbAccess(ActionBeanDBAccess):void
+ userLogin():String
+ processUserLogin(String,String):UserInfoBean
+ userLogInInfo(int):int
+ logout():String
«Java Class»
ActionBeanCSVUpload
edu.uic.ids517.fl6g319.action
+ file: UploadedFile
+ uploadType: String
+ fileName: String
+ extension: String
+ fileType: char
+ delimiter: String
+ viewDataType: String
+ rowCount: int
+ columnCount: int
+ selectedCourse: int
+ exam: String
+ points: Double
+ hasHeader: char
+ courseList: ArrayList<CourseBean>
+ objDBQueries: DBQueries
+ objConstants: Constants
+ userInfoBean: UserInfoBean
+ dbQuery: String
+ registeredUins: ArrayList<Integer>
# ActionBeanCSVUpload()
+ getFile():UploadedFile
+ setFile(UploadedFile):void
+ getUploadType():String
+ setUploadType(String):void
+ getFileName():String
+ setFileName(String):void
+ getExtension():String
+ setExtension(String):void
+ getDbAccess():ActionBeanDBAccess
+ setDbAccess(ActionBeanDBAccess):void
+ getFileType():char
+ setFileType(char):void
+ getDelimiter():String
+ setDelimiter(String):void
+ getViewDataType():String
+ setViewDataType(String):void
+ getRowCount():int
+ setRowCount(int):void
+ getColumnCount():int
+ setColumnCount(int):void
+ getUserInfoBean():UserInfoBean
+ setUserInfoBean(UserInfoBean):void
+ getSelectedCourse():int
+ setSelectedCourse(int):void
+ getCourseList():ArrayList<CourseBean>
+ setCourseList(ArrayList<CourseBean>):void
+ getExamId():String
+ setExamId(String):void
+ getHasHeader():char
+ setHasHeader(char):void
+ getObjDBQueries():DBQueries
+ setObjDBQueries(DBQueries):void
+ getObjConstants():Constants
+ setObjConstants(Constants):void
+ getDbQuery():String
+ setDbQuery(String):void
+ getRegisteredUins():ArrayList<Integer>
+ setRegisteredUins(ArrayList<Integer>):void
+ getExam():String
+ setExam(String):void
+ getPoints():Double
+ setPoints(Double):void
+ init():void
+ uploadAdminData():String
+ uploadData(String[],String):boolean
+ retrieveCourseList():void
+ uploadRoster():String
+ registerNewStudent(UserInfoBean,int):void
+ uploadQuestions():String
«Java Class»
ActionBeanInstructor
edu.uic.ids517.fl6g319.action
+ userInfoBean: UserInfoBean
+ examBean: ExamBean
+ examBeanList: ArrayList<ExamBean>
+ questionsList: ArrayList<ExamBean>
+ selectedCourse: int
+ selectedExam: String
+ examId: int
+ courseList: ArrayList<CourseBean>
+ objDBQueries: DBQueries
+ objConstants: Constants
+ dbQuery: String
+ renderAssessmentGrid: boolean
+ renderData: boolean
+ columnNames: List<String>
+ rosterResult: Result
+ courseRosterList: ArrayList<CourseRosterBean>
+ objCourseRoster: CourseRosterBean
+ showExamList: boolean
# ActionBeanInstructor()
+ getDbAccess():ActionBeanDBAccess
+ setDbAccess(ActionBeanDBAccess):void
+ getUserInfoBean():UserInfoBean
+ setUserInfoBean(UserInfoBean):void
+ getExamBean():ExamBean
+ setExamBean(ExamBean):void
+ getExamBeanList():ArrayList<ExamBean>
+ setExamBeanList(ArrayList<ExamBean>):void
+ getQuestionsList():ArrayList<CourseBean>
+ setQuestionsList(ArrayList<ExamBean>):void
+ getExamId():int
+ setExamId(int):void
+ getCourseList():ArrayList<CourseBean>
+ setCourseList(ArrayList<CourseBean>):void
+ getObjDBQueries():DBQueries
+ setObjDBQueries(DBQueries):void
+ getObjConstants():Constants
+ setObjConstants(Constants):void
+ getDbQuery():String
+ setDbQuery(String):void
+ isRenderAssessmentGrid():boolean
+ setRenderAssessmentGrid(boolean):void
+ isRenderData():boolean
+ setRenderData(boolean):void
+ getColumnNames():List<String>
+ setColumnNames(List<String>):void
+ getRosterResult():Result
+ setRosterResult(Result):void
+ getCourseRosterList():ArrayList<CourseRosterBean>
+ setCourseRosterList(ArrayList<CourseRosterBean>):void
+ getObjCourseRoster():CourseRosterBean
+ setObjCourseRoster(CourseRosterBean):void
+ isShowExamList():boolean
+ setShowExamList(boolean):void
+ init():void
+ retrieveCourseList():void
+ bindExams():String
+ bindQuestions():void
+ bindRoster():void
«Java Class»
ActionBeanExam
edu.uic.ids517.fl6g319.action
+ userInfoBean: UserInfoBean
+ examBean: ExamBean
+ examBeanList: ArrayList<ExamBean>
+ examAttemptList: ArrayList<ExamBean>
+ responseList: ArrayList<ExamBean>
+ questionsList: ArrayList<CourseBean>
+ rosterScore: CourseRosterBean
+ selectedCourse: int
+ selectedExam: String
+ examId: int
+ courseList: ArrayList<CourseBean>
+ transactionLogKey: int
+ objDBQueries: DBQueries
+ objConstants: Constants
+ dbQuery: String
+ renderGrid: boolean
+ renderAnswerGrid: boolean
+ disableCourse: boolean
+ renderRosterGrid: boolean
+ renderAssessmentGrid: boolean
+ showExamList: boolean
# ActionBeanExam()
+ getSelectedCourse():int
+ setSelectedCourse(int):void
+ getCourseList():ArrayList<CourseBean>
+ setCourseList(ArrayList<CourseBean>):void
+ getObjDBQueries():DBQueries
+ setObjDBQueries(DBQueries):void
+ getObjConstants():Constants
+ setObjConstants(Constants):void
+ getDbQuery():String
+ setDbQuery(String):void
+ isRenderGrid():boolean
+ setRenderGrid(boolean):void
+ getTransactionLogKey():int
+ setTransactionLogKey(int):void
+ getDbAccess():ActionBeanDBAccess
+ setDbAccess(ActionBeanDBAccess):void
+ getUserInfoBean():UserInfoBean
+ setUserInfoBean(UserInfoBean):void
+ getExamBean():ExamBean
+ setExamBean(ExamBean):void
+ getExamBeanList():ArrayList<ExamBean>
+ setExamBeanList(ArrayList<ExamBean>):void
+ getExamAttemptList():ArrayList<ExamBean>
+ setExamAttemptList(ArrayList<ExamBean>):void
+ getQuestionsList():ArrayList<ExamBean>
+ setQuestionsList(ArrayList<ExamBean>):void
+ getExamId():int
+ setExamId(int):void
+ isRenderAnswerGrid():boolean
+ setRenderAnswerGrid(boolean):void
+ isDisableCourse():boolean
+ setDisableCourse(boolean):void
+ getResponseList():ArrayList<ExamBean>
+ setResponseList(ArrayList<ExamBean>):void
+ isRenderRosterGrid():boolean
+ setRenderRosterGrid(boolean):void
+ getRosterScore():CourseRosterBean
+ setRosterScore(CourseRosterBean):void
+ getSelectedExam():String
+ setSelectedExam(String):void
+ isRenderAssessmentGrid():boolean
+ setRenderAssessmentGrid(boolean):void
+ isShowExamList():boolean
+ setShowExamList(boolean):void
+ init():void
+ retrieveCourseList():void
+ bindAllCourses():void
+ loadExamList():String
+ startExam(ExamBean):String
+ retrieveExamQuestions(ExamBean):ArrayList<ExamBean>
+ LogExamStart(ExamBean):int
+ submitExam():String
+ saveResponses():String
+ updateRoster(double,String):String
+ LogExamEnd(int):void
+ loadScores():String
+ showResponses():String
+ clearLists():void
+ scores():String
+ home():String
+ exams():String
-actionBeanDBAccess 0..1
-dbAccess 0..1
-dbAccess 0..1
-dbAccess 0..1
-dbAccess 0..1
«Java Class»
ActionBeanDBAccess
edu.uic.ids517.fl6g319.action
+ conn: Connection
+ stmt: Statement
+ rs: ResultSet
+ rsmd: ResultSetMetaData
+ dbmd: DatabaseMetaData
+ columnNames: List<String>
+ rowCount: int
+ columnCount: int
+ result: Result
+ dbQuery: String
+ prepStmt: PreparedStatement
+ objDBQueries: DBQueries
+ objConst: Constants
+ userInfoBean: UserInfoBean
+ dbLogin: DBLoginBean
+ renderTableList: boolean
+ renderData: boolean
+ tableList: List<String>
+ selectedTables: List<String>
+ selectedColumnNames: String[]
+ renderColumnsButton: boolean
+ isRenderColumnsButton():boolean
+ setRenderColumnsButton(boolean):void
+ isRenderTableList():boolean
+ setRenderTableList(boolean):void
+ isRenderData():boolean
+ setRenderData(boolean):void
+ getTableList():List<String>
+ setTableList(List<String>):void
+ getSelectedTables():List<String>
+ setSelectedTables(List<String>):void
+ getDbmd():DatabaseMetaData
+ setDbmd(DatabaseMetaData):void
+ getSelectedColumnNames():String[]
+ setSelectedColumnNames(String[]):void
# ActionBeanDBAccess()
+ getConn():Connection
+ setConn(Connection):void
+ getStmt():Statement
+ setStmt(Statement):void
+ getRs():ResultSet
+ setRs(ResultSet):void
+ getDbLogin():DBLoginBean
+ setDbLogin(DBLoginBean):void
+ getColumnNames():List<String>
+ setColumnNames(List<String>):void
+ getRowCount():int
+ setRowCount(int):void
+ getColumnCount():int
+ setColumnCount(int):void
+ getRsmd():ResultSetMetaData
+ setRsmd(ResultSetMetaData):void
+ getResult():Result
+ setResult(Result):void
+ getDbQuery():String
+ setDbQuery(String):void
+ getPrepStmt():PreparedStatement
+ setPrepStmt(PreparedStatement):void
+ getObjDBQueries():DBQueries
+ setObjDBQueries(DBQueries):void
+ getUserInfoBean():UserInfoBean
+ setUserInfoBean(UserInfoBean):void
+ getObjConst():Constants
+ setObjConst(Constants):void
+ establishConnection():boolean
+ selectQuery(String):String
+ displayColumnName():void
+ executeUpdate(String):String
+ insertRecordGetKey(PreparedStatement):int
+ executeUpdatePrepStmt(PreparedStatement):String
+ executePrepStmt(PreparedStatement):String
+ processSQLQuery():void
+ listTables():void
+ listColumns():void
+ processSelectedColumns():void
+ processDisplayTable():void
+ createTables():String
+ dropTables():String
+ createUsers():void
+ insertCourses():void
+ insertEnrollments():void
+ insertExams():void
+ initiateDBLogin():String
+ setupApplication():String
+ checkApplicationTables():boolean
+ logout():String
+ home():String
-dbAccess 0..1
-dbAccess 0..1

- **Database Design and Constraints**

The new tables to be created as a part of the project are as follows -

- F16G319_UserTable
  The table will contain the details of the users who would use the application. The details include Last name, First name, user name and UIN. The table structure would look similar to this -

UIN	Interger
Last_Name	varchar(50)
First_name	varchar(50)
User_Name	Varchar(30)
Password	varchar(50)
User_Type	Varchar(15)
Is_User_Active	boolean

- F16G319_Course
  The table will store the data for different courses. The structure of the table would look similar to this -

CRN	Interger
code	varchar(50)
description	varchar(50)
Name	varchar(50)

- F16G319_Enrollment
  The table will store the data of the tests instances such as the user that takes the test and the corresponding course. The table would look as follows -

UIN	Interger
CRN	Interger

- F16G319_Question
  The table structure is provided here -

Question_ID	Interger
Exam_ID	Interger
Question_Type	varchar(20)
Question_Text	varchar(50)
Correct_Answer	varchar(50)
Points	decimal(10,2)
Tolerance	Decimal(10,2)

- F16G319_Exam
  The table will contain the login details of the user such as User name, Password and if the user if active or not.

Exam_ID	Interger
CRN	integer
Exam	varchar(10)
Start_Date	DateTime
End_Date	DateTime
Duration	integer

- F16G319_Login Info
  This table consists of the information regarding every login session that has occurred on the application. The structure of the table would look as follows -

LoginInfo_ID	Interger
UIN	Varchar(50)
Session_StartTime	Date
Session_EndTime	Date
IP_Address	Varchar(20)

- F16G319_ Exam_log
  The table provides the details on every transaction that has occurred on the application. The attributes of the table are shown here.

Log_ID	Interger
LoginInfo_ID	Interger
Exam_Id	Integer
UIN	Integer
CRN	Integer
ExamAttempt_Startdate	Date
ExamAttempt_EndDate	Date

- F16G319_ Student_Response
  This table contains the scores of the students for combinations of tests and courses. The table can look as follows -

Log_ID	Interger
Question_Id	Interger
Student_Response	Varchar(20)
Correct_Answer	varchar(50)
Points_Scored	Decimal(10,2)

- F16G319_Score_Roster
  This table contains the scores of the students for combinations of tests and courses. The table can look as follows -

CRN	Interger
Last_Name	varchar(50)
First_name	varchar(50)
User_Name	varchar(20)

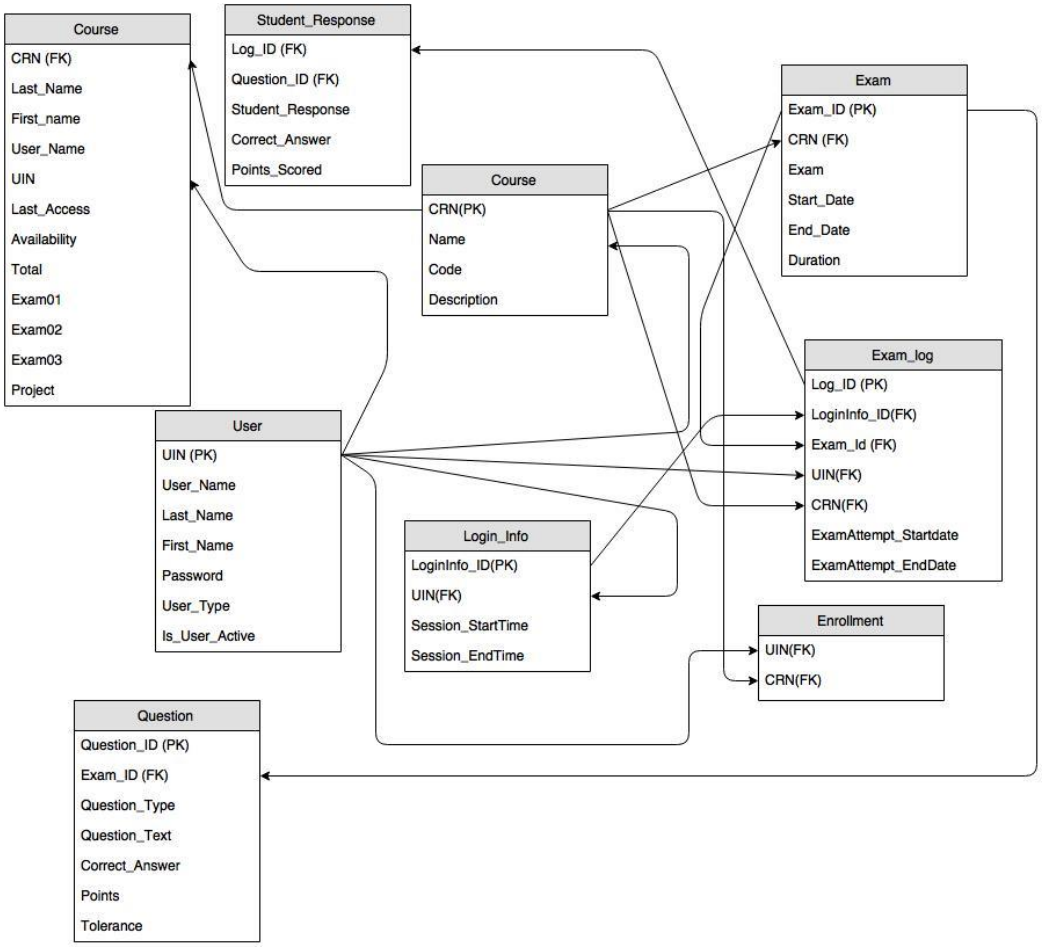
UIN	Integer
Last_Access	DateTime
Availability	boolean
Total	decimal(10,2)
Exam01	decimal(10,2)
Exam02	decimal(10,2)
Exam03	decimal(10,2)
Project	decimal(10,2)

- **SQL Statements to create tables on DB Schema**

  The SQL script attached here provides the SQL statements to create the tables on a schema.

- **Relational Diagram of the tables**

The relational schema diagram of the database is shown here.



- **Flow of the application**

The diagram shows a high level flow of the application.