CSC3003S Capstone Project:
SQL Automatic Marker

Zach Meltzer
MLTZAC001
MLTZAC001@myuct.ac.za

Elvis Sebatane
SBTELV001
SBTELV001@myuct.ac.za

Rian Tshepe
TSHRIA002
TSHRIA002@myuct.ac.za

The aim of this project was to develop a system that creates, marks, and allows students to take Structured Query Language assessments for the University of Cape Town's Department of Computer Science. Due to various factors the scope of the project had to be reduced and two desired functionalities had to be discarded: the ability to upload and parse a datafile containing schema and question-answer sets and the automatic generation of SQL question-answer sets. Aside from this, the resultant project allows a teacher to create assessments, manage the students enrolled, and view marks for assessments while the students can take assessments (and retry them with a different set of questions, depending on the assessment type) and view their results.

Contents

1. Introduction

1.1. Overview

This project aimed to create a system to assign and automatically mark SQL database assignments. This system was created to enhance the University of Cape Town's Computer Science department's assignment marking. There already exists an automatic marker for code (Java, Python, etc.) yet assignments that involve SQL are currently marked manually, which often takes up an unnecessary amount of time and effort.

The purpose of this system is to reduce the need for manual marking of assignments – which are prone to human errors, and to different marking styles of tutors (one might be extremely lenient while another may be overly strict, despite there being a memorandum). Through its automation, the marking process would be able happen in real time, provide feedback to the student as soon as s/he has completed the assignment, and record marks in a manner that is easily accessible to the assessor.

Additionally, with this system, the department would have the option of running assignments as a closed practical assessment or as a homework assignment. Unlike the current method, where the student has vast amounts of time to complete their SQL assignment and is able to cheat or get copious amounts of help, this automated version looks to reduce the ability of a student to cheat and to reflect their real academic capabilities.

The system, by default, will have two pre-installed models for which the lecturer may choose to examine from or alternate from year to year. If the assessor would like to use his/her own dataset, the system has been set up in such a way that s/he would be able to import the datasets directly into MySQL and have it reflect in the system's interface. Furthermore, the decision was taken to create an online MySQL server, hosted by Amazon's Web Service in order to allow for easy access to the raw data and information in order to easily add to/ remove from the necessary tables and databases.

1.2. Goals (Scope)

The scope and goals of this system are simple and conform with SMART criteria:

- Each question in an assignment will indicate its level of difficulty in the form of mark allocation.
- Difficulty levels are split into 7 criteria labelled A through G.
- The number of attempts allowed per assignment will be shown to students.
- The student will be able to view a list of assignments needed to be completed.
- The student will be able to see his/her mark and receive feedback at the end of each assignment or when s/he wishes.
- The lecturer will have the option to set as many assessments (assignments or closed practicals) as s/he wishes.
- Each student will be allocated a unique set of questions (as specified by the staff member) that will be taken at random from the chosen database.
- The staff member can make use of the pre-installed data sets or add new ones into the system through MySQL.
- The lecturer will be able to upload a list of student numbers, which in turn will allow those users access to the assignment(s) as well as record their marks.
- When a lecturer uploads a list of student numbers, each user will be automatically assigned a secure password for the lecturer to provide to the student.
- All users will be able to log out as long as they are not in the middle of a task (i.e. during the assignment).

- Any reattempt at the assessment with result in a new question set being generated for the user.
- Each question/ the assignment as a whole will have a time limit that will be shown to the user. If the user does not finish on time, the application will disallow submissions after the time has elapsed.
- The lecturer will be able to remove all students from the database. The lecturer will be able to remove all student files, results, and assessments directly from MySQL.
- Submission happens in batch rather than one at a time.
- A student will be able to run their query to find out what their answer looks like.

1.3. Inputs, Output, and Performance

The student will be able to input the SQL query into a text area and submit the answer. This input (answer) will be stored in a table named after the student number of the student, along with a key linking the answer to the specific question, and the output produced by the student's SQL statement. At the end of the assignment, all of the student's outputs will be compared with the expected output. If the outputs match, the student will get full marks. If the outputs do not match but their query contains certain key words (dependent on category of question), the user will get a portion of the marks but not full marks. If there is an SQL or compile error, then the student will get 0 for the question. This information is stored in the database under the student's individual file as well as the mark file for the assessment.

After the computation and comparison of outputs, the system will display the student's final mark. In the case of an incorrectly answered statement, the student will be provided feedback - i.e. the expected output (even though there are many ways to execute one query, the program will provide one) while correctly answered questions will just produce the student's answer.

1.4. Software Engineering Methodology

This project made use of a hybridised form of agile development methodology which included some traditional methodology elements, working within a fixed time frame, in order to ensure adaptive planning, evolutionary development, timely delivery, continuous improvement, and a rapid and flexible response to change. The system was developed iteratively which aided in the assessment and discovery of possible problems and bugs that may have arisen, in order to explore design and solution techniques to remedy such issues. The iterative approach complemented the use of evolutionary prototyping which was used to investigate workflow, task designs, screen layouts, and the information display which in turn enabled stakeholders to view, interact, and engage with the prototype of the software more easily as well as the fact that it enabled real-time feedback. This feedback was used to mitigate any uncertainty and address high risk issues surrounding requirements on the part of the client/stakeholder.

2. Requirements Captured

2.1. Use Case Scenarios

The Use Case Diagram in Figure 1 stipulates the scope of the system detailed workings of the system's functional requirements.

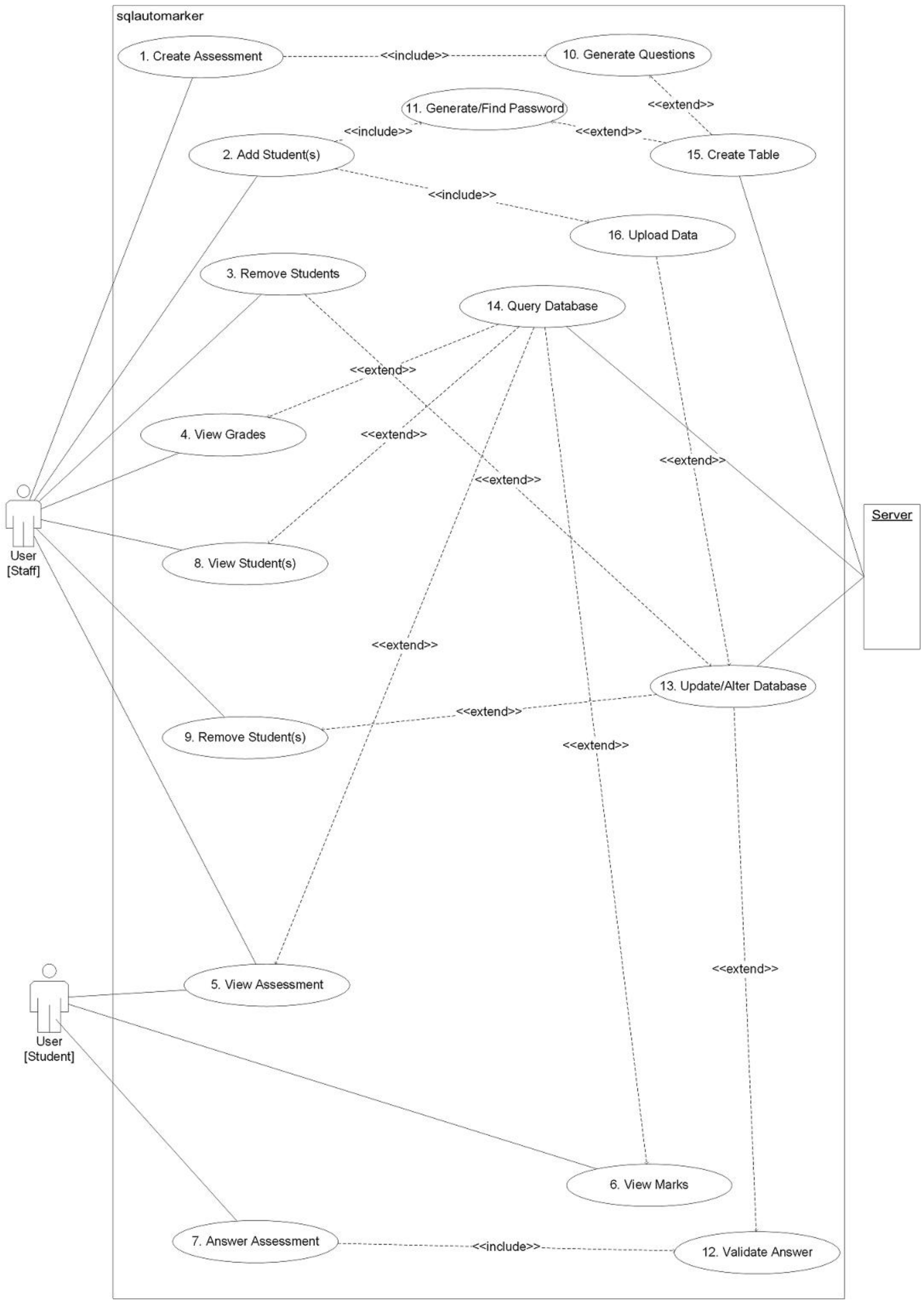
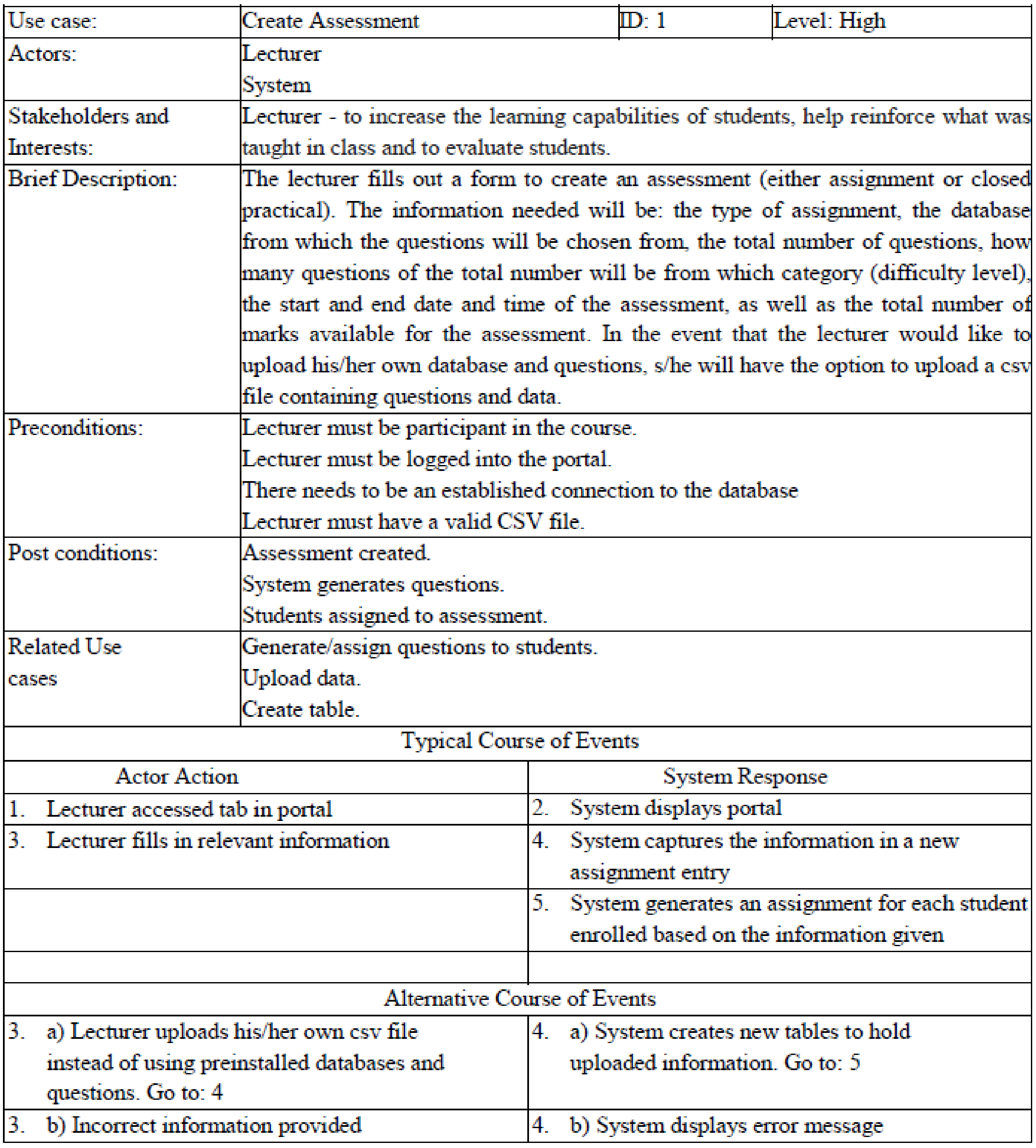


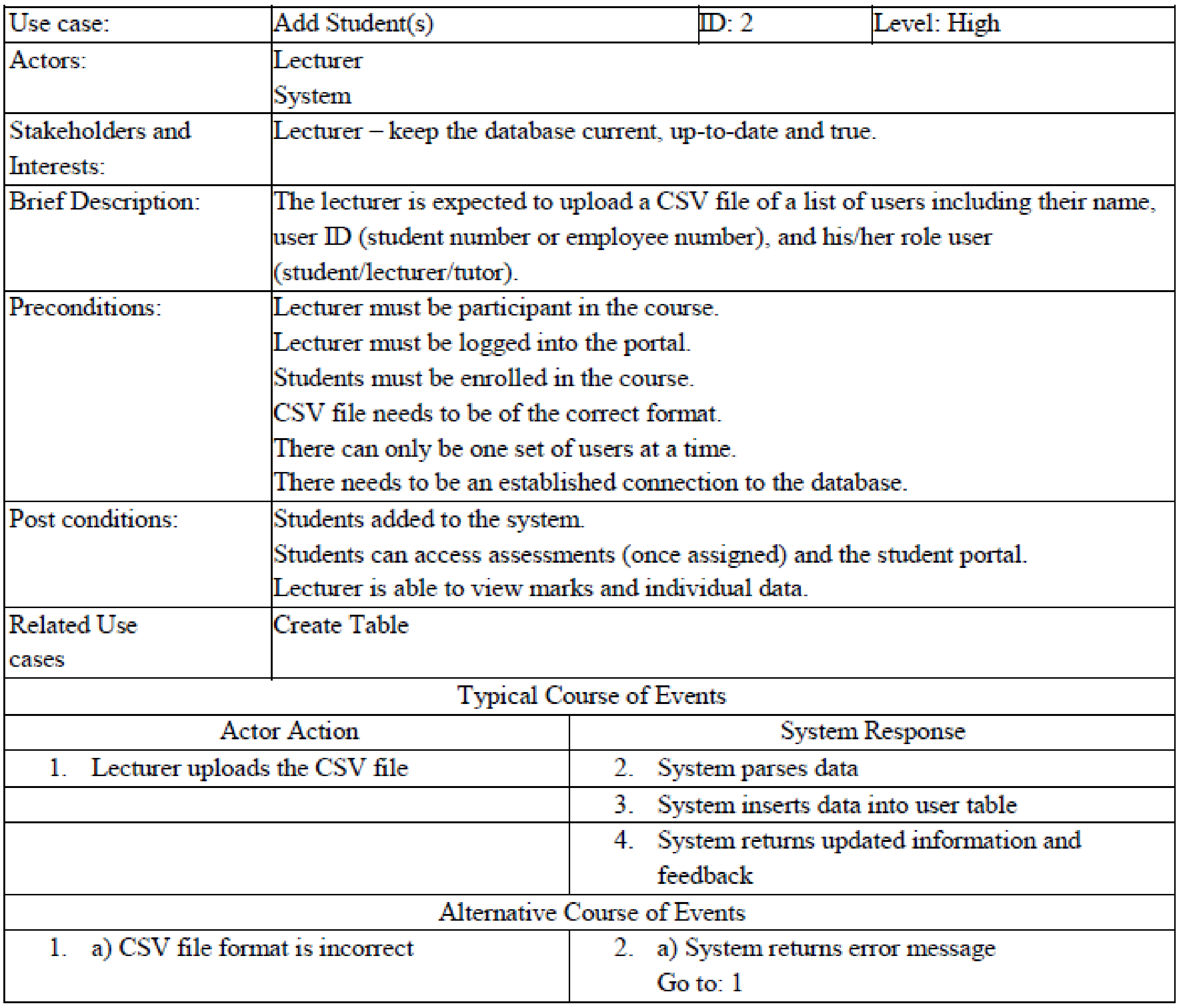
Figure 1: Use Case Diagram for the SQL Automatic Marker



Use case:	Create Assessment	ID: 1	Level: High
Actors:	Lecturer System		
Stakeholders and Interests:	Lecturer - to increase the learning capabilities of students, help reinforce what was taught in class and to evaluate students.		
Brief Description:	The lecturer fills out a form to create an assessment (either assignment or closed practical). The information needed will be: the type of assignment, the database from which the questions will be chosen from, the total number of questions, how many questions of the total number will be from which category (difficulty level), the start and end date and time of the assessment, as well as the total number of marks available for the assessment. In the event that the lecturer would like to upload his/her own database and questions, s/he will have the option to upload a csv file containing questions and data.		
Preconditions:	Lecturer must be participant in the course. Lecturer must be logged into the portal. There needs to be an established connection to the database Lecturer must have a valid CSV file.		
Post conditions:	Assessment created. System generates questions. Students assigned to assessment.		
Related Use cases	Generate/assign questions to students. Upload data. Create table.		

Typical Course of Events	
Actor Action	System Response
1. Lecturer accessed tab in portal	2. System displays portal
3. Lecturer fills in relevant information	4. System captures the information in a new assignment entry
	5. System generates an assignment for each student enrolled based on the information given
Alternative Course of Events	
3. a) Lecturer uploads his/her own csv file instead of using preinstalled databases and questions. Go to: 4	4. a) System creates new tables to hold uploaded information. Go to: 5
3. b) Incorrect information provided	4. b) System displays error message

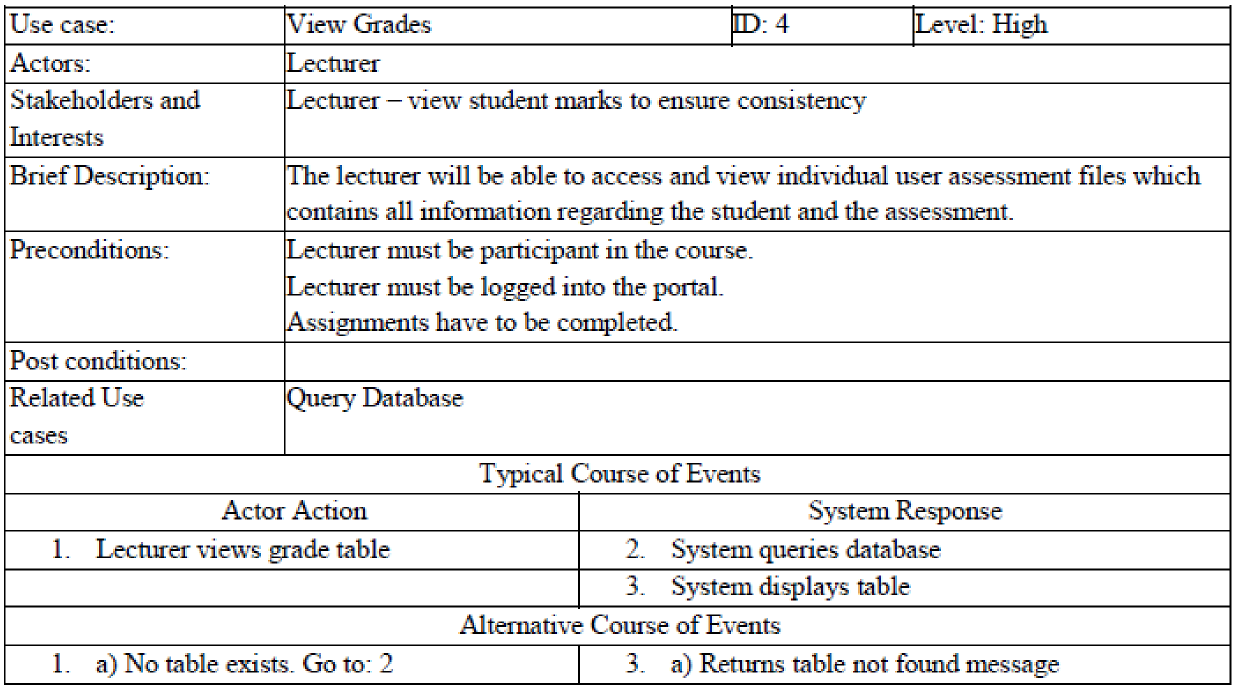
Figure 2: Use Case Narrative for Creating an Assessment



Use case:	Add Student(s)	ID: 2	Level: High
Actors:	Lecturer System		
Stakeholders and Interests:	Lecturer – keep the database current, up-to-date and true.		
Brief Description:	The lecturer is expected to upload a CSV file of a list of users including their name, user ID (student number or employee number), and his/her role user (student/lecturer/tutor).		
Preconditions:	Lecturer must be participant in the course. Lecturer must be logged into the portal. Students must be enrolled in the course. CSV file needs to be of the correct format. There can only be one set of users at a time. There needs to be an established connection to the database.		
Post conditions:	Students added to the system. Students can access assessments (once assigned) and the student portal. Lecturer is able to view marks and individual data.		
Related Use cases	Create Table		

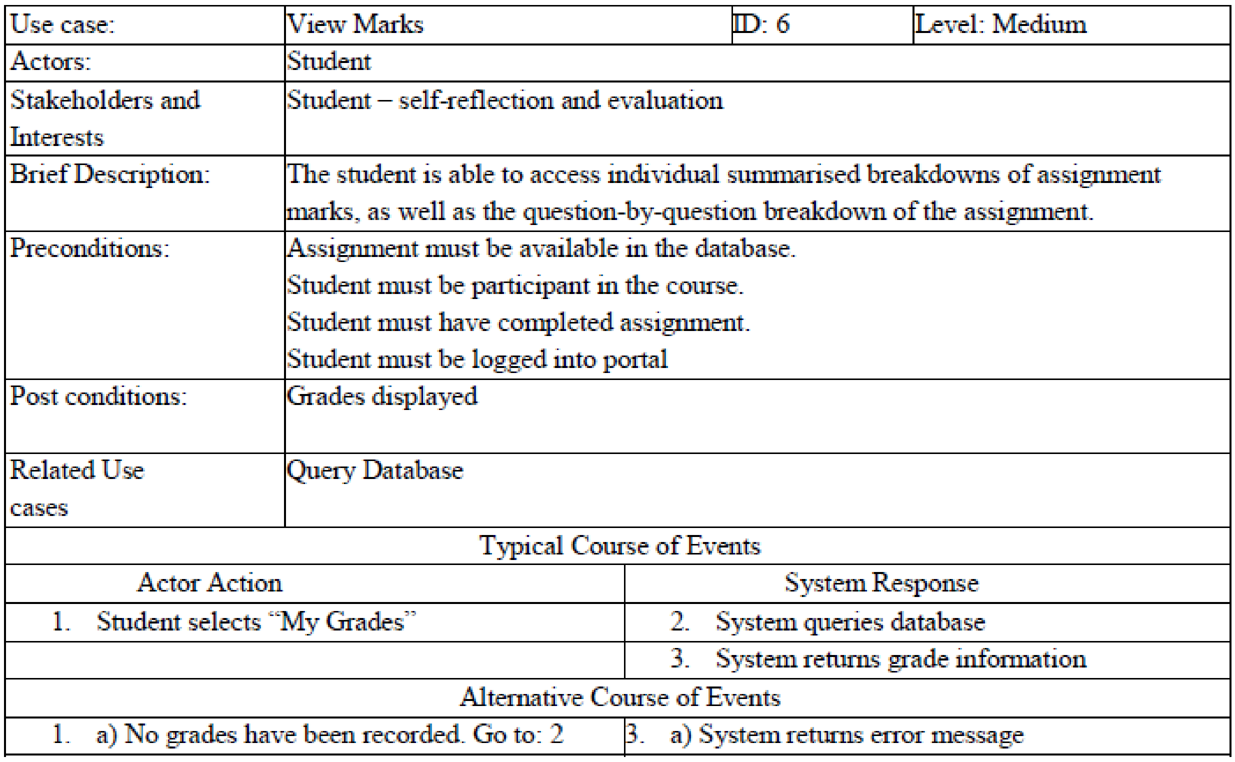
Typical Course of Events	
Actor Action	System Response
1. Lecturer uploads the CSV file	2. System parses data
	3. System inserts data into user table
	4. System returns updated information and feedback
Alternative Course of Events	
1. a) CSV file format is incorrect	2. a) System returns error message Go to: 1

Figure 3: Use Case Narrative for Adding an Assessment



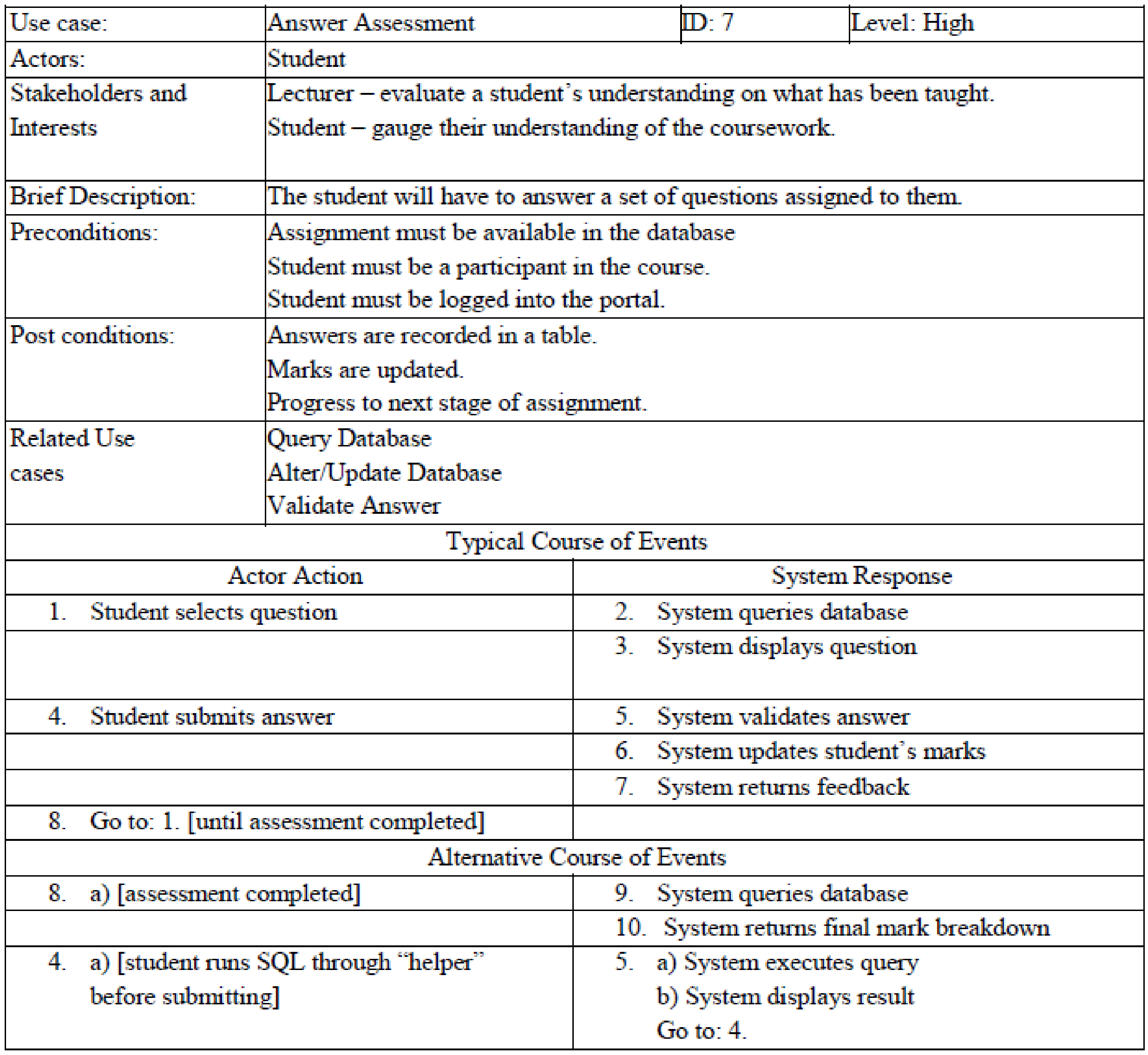
Use case:	View Grades	ID: 4	Level: High
Actors:	Lecturer		
Stakeholders and Interests	Lecturer – view student marks to ensure consistency		
Brief Description:	The lecturer will be able to access and view individual user assessment files which contains all information regarding the student and the assessment.		
Preconditions:	Lecturer must be participant in the course. Lecturer must be logged into the portal. Assignments have to be completed.		
Post conditions:			
Related Use cases	Query Database		
Typical Course of Events			
Actor Action		System Response	
1. Lecturer views grade table		2. System queries database	
		3. System displays table	
Alternative Course of Events			
1. a) No table exists. Go to: 2		3. a) Returns table not found message	

Figure 4: Use Case Narrative for Viewing Grades (Staff)



Use case:	View Marks	ID: 6	Level: Medium
Actors:	Student		
Stakeholders and Interests	Student – self-reflection and evaluation		
Brief Description:	The student is able to access individual summarised breakdowns of assignment marks, as well as the question-by-question breakdown of the assignment.		
Preconditions:	Assignment must be available in the database. Student must be participant in the course. Student must have completed assignment. Student must be logged into portal		
Post conditions:	Grades displayed		
Related Use cases	Query Database		
Typical Course of Events			
Actor Action		System Response	
1. Student selects "My Grades"		2. System queries database	
		3. System returns grade information	
Alternative Course of Events			
1. a) No grades have been recorded. Go to: 2		3. a) System returns error message	

Figure 5: Use Case Narrative for Viewing Assessment Mark (Student)



Use case:	Answer Assessment	ID: 7	Level: High
Actors:	Student		
Stakeholders and Interests	Lecturer – evaluate a student's understanding on what has been taught. Student – gauge their understanding of the coursework.		
Brief Description:	The student will have to answer a set of questions assigned to them.		
Preconditions:	Assignment must be available in the database Student must be a participant in the course. Student must be logged into the portal.		
Post conditions:	Answers are recorded in a table. Marks are updated. Progress to next stage of assignment.		
Related Use cases	Query Database Alter/Update Database Validate Answer		

Typical Course of Events	
Actor Action	System Response
1. Student selects question	2. System queries database
	3. System displays question
4. Student submits answer	5. System validates answer
	6. System updates student's marks
	7. System returns feedback
8. Go to: 1. [until assessment completed]	
Alternative Course of Events	
8. a) [assessment completed]	9. System queries database
	10. System returns final mark breakdown
4. a) [student runs SQL through "helper" before submitting]	5. a) System executes query b) System displays result Go to: 4.

Figure 6: Use Case Narrative for Answering an Assessment

2.2. Non-Functional Requirements

In addition to the functional requirements, there are the following non-functional requirements:

1. Performance

   (a) Speed:

   The internet speed of the user impacts the overall responsiveness of the system. Furthermore, if the server is not fast enough, the more connections that occur with it, the more unresponsive it will become.

   (b) Throughput:

   The system needs to be able to handle at least three hundred users.

   (c) Response Time:

   The response time of the system is directly proportional to the speed of the user's internet along with the number of users accessing the system.

2. Reliability

   (a) Frequency of System Failure:

      The system must not fail.

   (b) Accuracy:

      The data sets produced by the system must be 100% accurate at all times.

   (c) Availability:

      The system must be available for use at any time.

3. Scalability

   (a) Geographic scalability:

      The system should be accessible from any location provided there is an internet connection.

   (b) Functional scalability:

      Features of the system can easily be added, removed, or modified by the administrator of the system.

4. Maintainability

   (a) Adaptive Maintenance: The system should be able to be hosted on multiple platforms without affecting the performance of the system as a whole.
   (b) Perfective Maintenance: The system database should be able to be updated with ease without causing a system failure.
   (c) Corrective Maintenance: Administrators should be able to modify marks and results on the system manually.

2.3. Usability Requirements

In order for the system to be usable, the following conditions have to be met:

1. The user interface should be easy to understand.
2. Interface actions and elements should be consistent throughout the program.
3. They interface layout should be appealing.
4. Students must be added to the system by the system administrator (lecturer) in order to be able to use the system.
5. Operability

   (a) Actions that cannot be undone should ask for confirmation before proceeding.
   (b) Error messages should provide sufficient feedback on the ill-occurrence.

3. Design Overview

3.1. System Architecture

The system design separates different aspects of the application with layers (User Interface, Application, Doman, and Infrastructure), providing a loose coupling between these elements. The design specifies where each kind of logic should be located within the application. The separation of aspects manages the complexity when building the application as it enabled us to focus on one aspect of the implementation at a time. For example, the User Interface was worked on without having to depend on the Application or Domain layers, or how data is handled, manipulated, or stored. This layered client-server approach was chosen because the system being designed deals with multiple ways of viewing and interacting with data. Moreover, future requirements for the system's interaction with and presentation of data are unknown. Thus, by developing a system with different tasks assigned to different individuals, each being assigned a responsibility to develop a functionality, enables the construction of a facility on top of existing functionality, allowing the data to be exchanged independently of its representation.

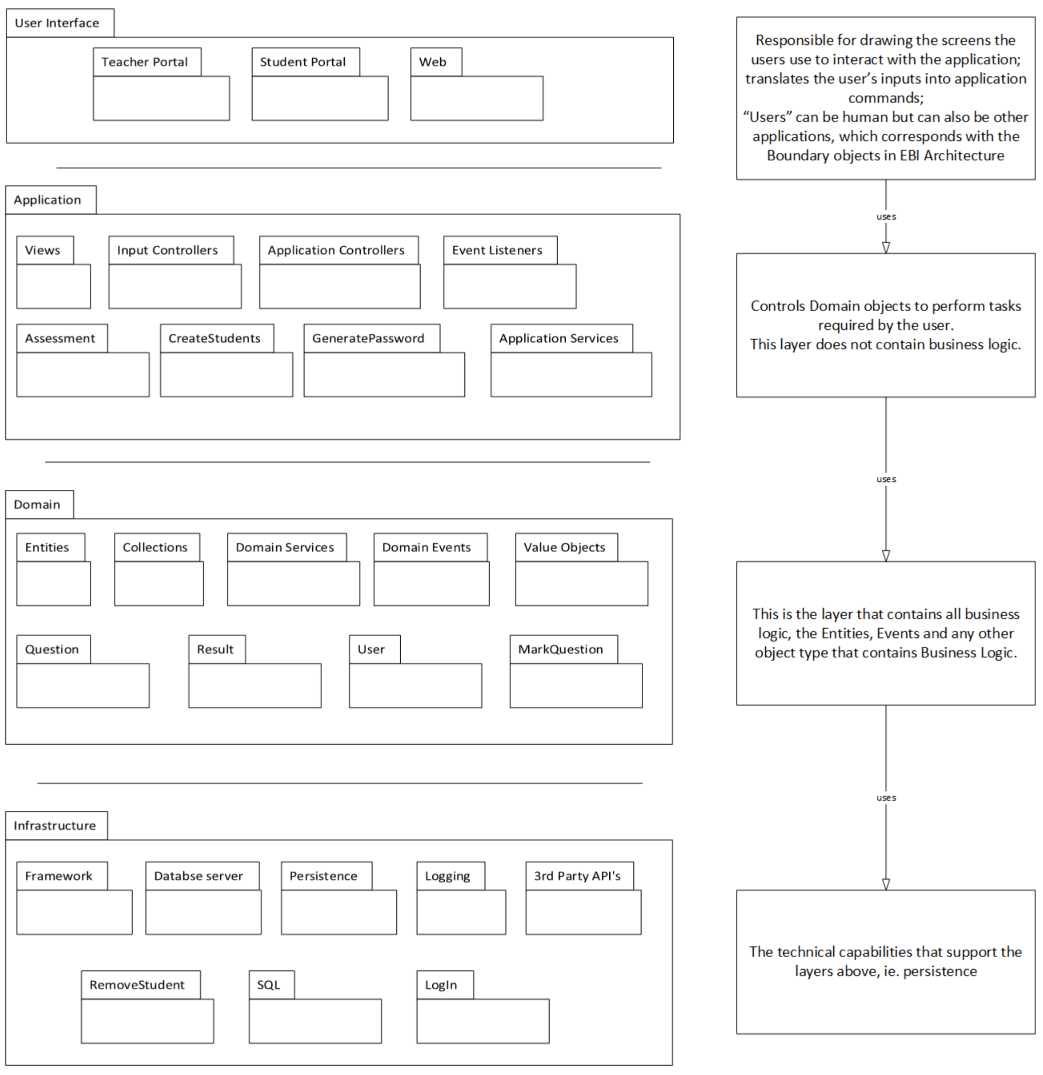

User Interface
Teacher Portal
Student Portal
Web
Application
Views
Input Controllers
Application Controllers
Event Listeners
Assessment
CreateStudents
GeneratePassword
Application Services
Domain
Entities
Collections
Domain Services
Domain Events
Value Objects
Question
Result
User
MarkQuestion
Infrastructure
Framework
Databse server
Persistence
Logging
3rd Party API's
RemoveStudent
SQL
LogIn
Responsible for drawing the screens the users use to interact with the application; translates the user's inputs into application commands; "Users" can be human but can also be other applications, which corresponds with the Boundary objects in EBI Architecture
uses
Controls Domain objects to perform tasks required by the user. This layer does not contain business logic.
uses
This is the layer that contains all business logic, the Entities, Events and any other object type that contains Business Logic.
uses
The technical capabilities that support the layers above, ie. persistence

Figure 7: Architecture Diagram of the SQL Automatic Marker

3.2. Classes

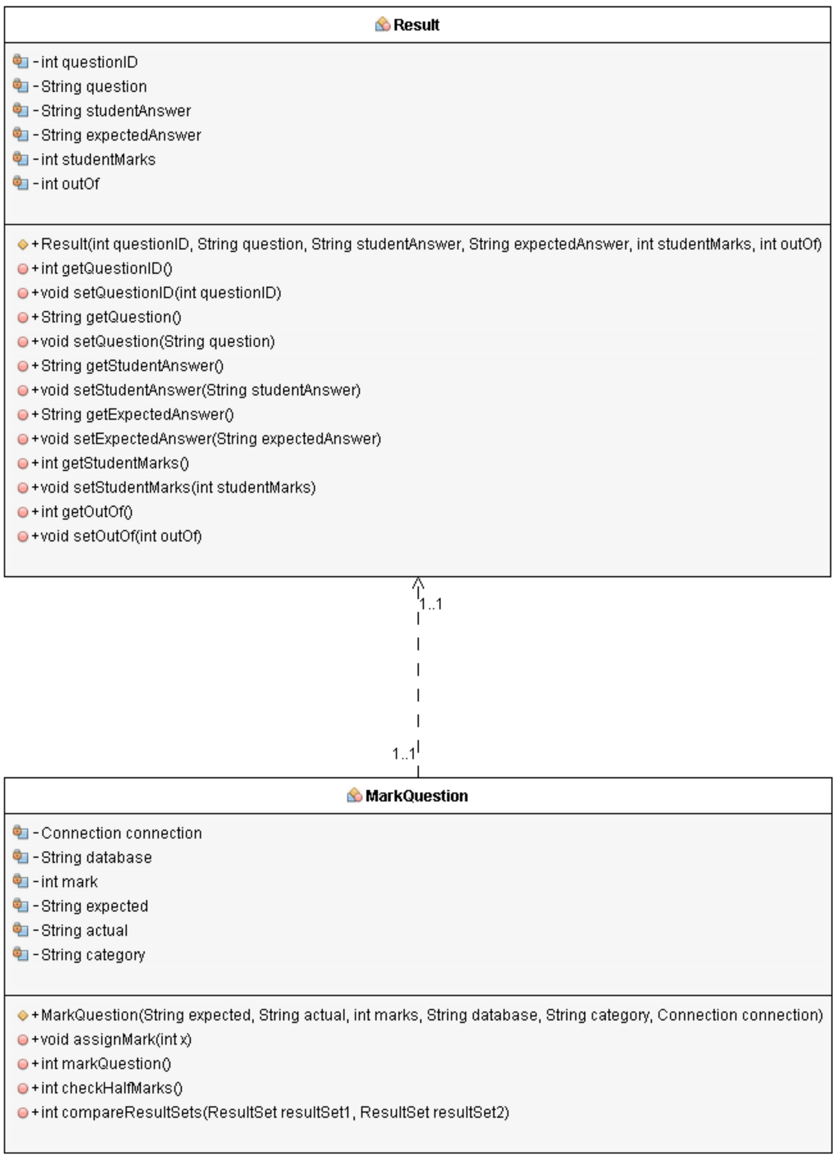

Result
- int questionID
- String question
- String studentAnswer
- String expectedAnswer
- int studentMarks
- int outOf
+Result(int questionID, String question, String studentAnswer, String expectedAnswer, int studentMarks, int outOf)
+int getQuestionID()
+void setQuestionID(int questionID)
+String getQuestion()
+void setQuestion(String question)
+String getStudentAnswer()
+void setStudentAnswer(String studentAnswer)
+String getExpectedAnswer()
+void setExpectedAnswer(String expectedAnswer)
+int getStudentMarks()
+void setStudentMarks(int studentMarks)
+int getOutOf()
+void setOutOf(int outOf)
1..1
1..1
MarkQuestion
- Connection connection
- String database
- int mark
- String expected
- String actual
- String category
+MarkQuestion(String expected, String actual, int marks, String database, String category, Connection connection)
+void assignMark(int x)
+int markQuestion()
+int checkHalfMarks()
+int compareResultSets(ResultSet resultSet1, ResultSet resultSet2)

Figure 8: Detail of the Analysis Class Diagram

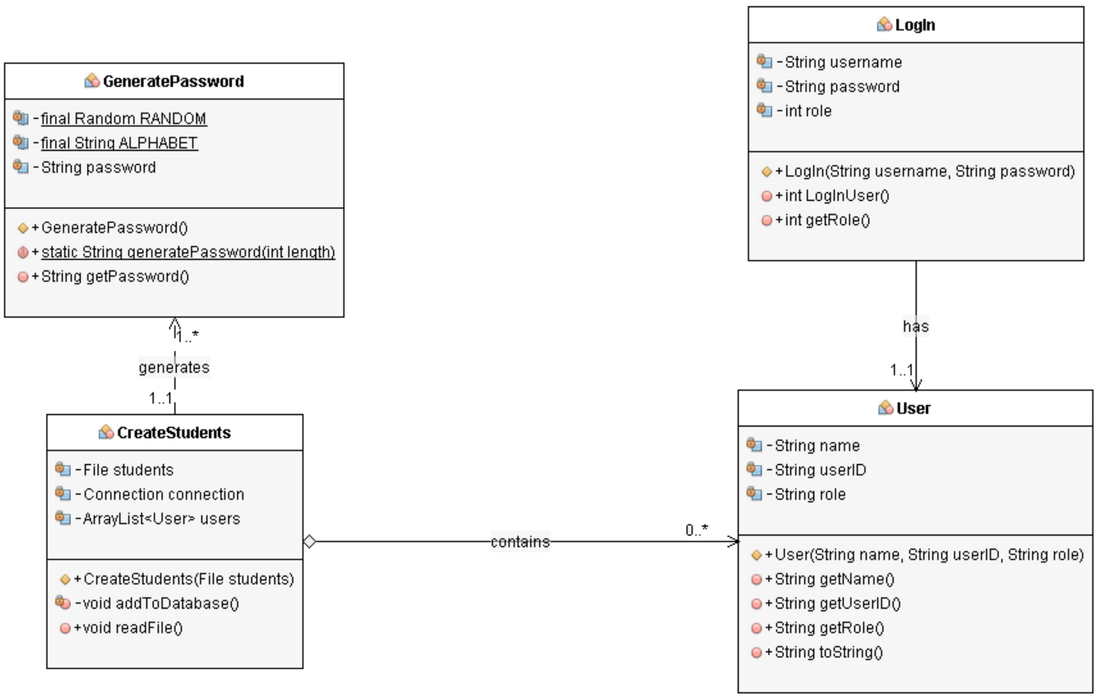


Figure 9: Detail of the Analysis Class Diagram

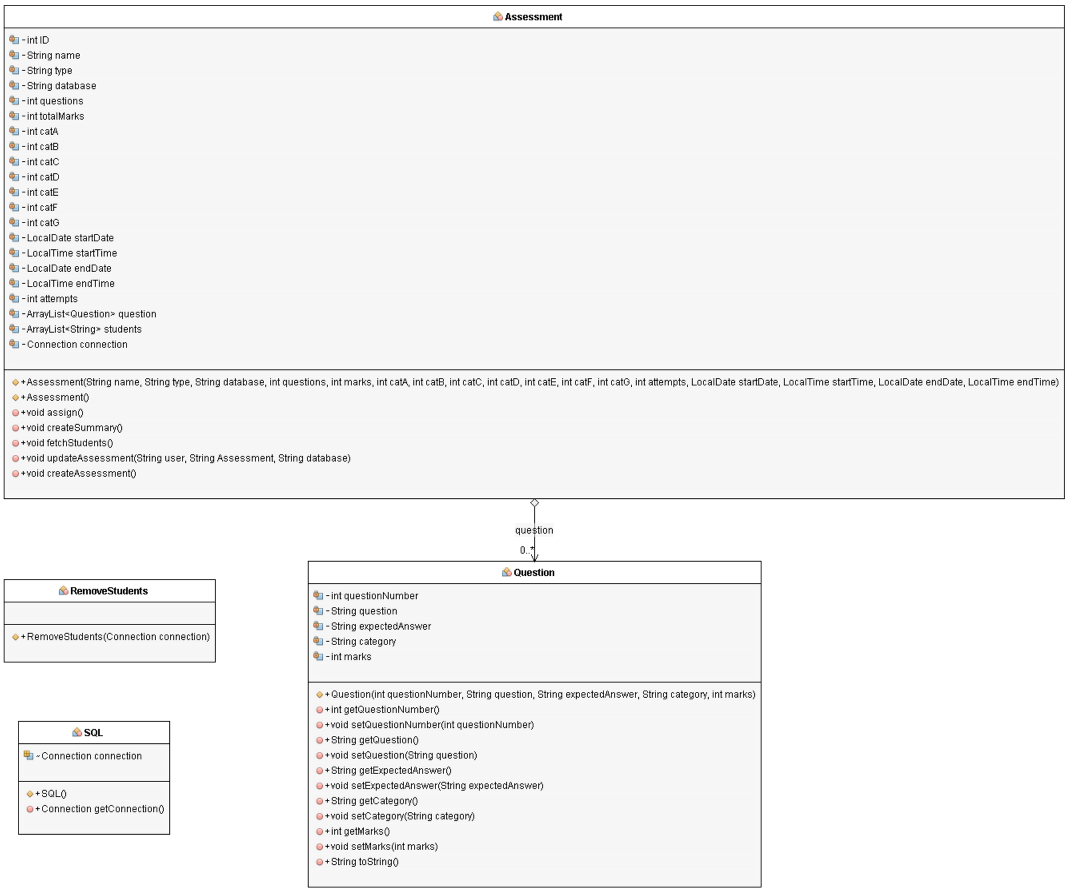

Assessment
-int ID
-String name
-String type
-String database
-int questions
-int totalMarks
-int catA
-int catB
-int catC
-int catD
-int catE
-int catF
-int catG
-LocalDate startDate
-LocalTime startTime
-LocalDate endDate
-LocalTime endTime
-int attempts
-ArrayList<Question> question
-ArrayList<String> students
-Connection connection
+Assessment(String name, String type, String database, int questions, int marks, int catA, int catB, int catC, int catD, int catE, int catF, int catG, int attempts, LocalDate startDate, LocalTime startTime, LocalDate endDate, LocalTime endTime)
+Assessment()
+void assign()
+void createSummary()
+void fetchStudents()
+void updateAssessment(String user, String Assessment, String database)
+void createAssessment()
question
0..*
RemoveStudents
+RemoveStudents(Connection connection)
SQL
~Connection connection
+SQL()
+Connection getConnection()
Question
-int questionNumber
-String question
-String expectedAnswer
-String category
-int marks
+Question(int questionNumber, String question, String expectedAnswer, String category, int marks)
+int getQuestionNumber()
+void setQuestionNumber(int questionNumber)
+String getQuestion()
+void setQuestion(String question)
+String getExpectedAnswer()
+void setExpectedAnswer(String expectedAnswer)
+String getCategory()
+void setCategory(String category)
+int getMarks()
+void setMarks(int marks)
+String toString()

Figure 10: Detail of the Analysis Class Diagram

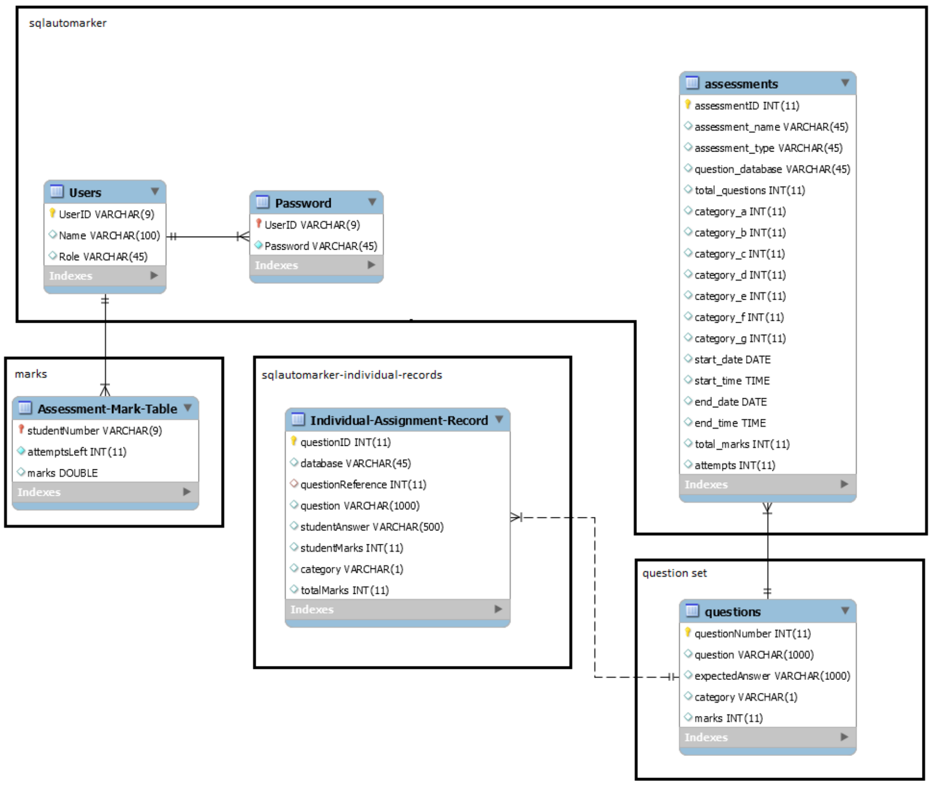


Figure 11: Enhanced Entity-Relationship Model

4. Implementation

4.1. Data Structures Used

The SQL Automatic Marker uses an object-oriented approach to create models based on real-world ideas. These objects consist of a collection of entities as well as existing relationships between and among these objects, using techniques like encapsulation, inheritance, methods, and object identity to aid in providing the system with an interface. The object-relational data model combines features of the object-oriented data model and the relational database model.

4.1.1. MySQL

MySQL is a relational database management system in which data items are organized as a set of formally-described tables from which data can be accessed, manipulated, reassembled, and created in various ways without having to reorganize the database tables. Please see Figure 11 for the types of databases and tables that are used as depicted below. Due to the fact that this system creates new tables in different databases for various reasons, the database management system needs to be able to handle such tasks. Further, MySQL was used because of the user interface it provides. By connecting to the server which is hosted by Amazon Web Services, anyone who connects to it through MySQL will have instant access to all the records. With this, the lecturer or tutor is afforded additional functionalities as opposed to merely using the front-facing interface. Where the front-facing user interface is primarily for viewing, taking, assigning, and marking assessments, with the additional connection to the actual MySQL database, the lecturer or tutor will be able to alter marks in the event of cheating or unnecessary difficulty, add new databases of questions by importing csv files directly into the database, as well as remove redundant individual records at the end of the course or assessment marking/moderation phase.

4.1.2. ArrayLists

ArrayLists were used to store any relevant data that needed to be stored after running a SQL query, be it a simple Integer or String value, or a more complex object such as User, Question, or a Result. An ArrayList was chosen over using traditional Arrays due to the fact that ArrayLists are dynamically resizable while traditional Arrays are of a fixed length as well as the fact that there is little performance difference between traversing through an ArrayList compared to traditional Array traversal.

4.1.3. Arrays

Arrays were used primarily within the Javascript method of sending the student's assessment to be marked. This method used an array to capture all answers by the student and send them to the JSP controller page to begin the marking of the assessment.

4.2. Interface Development

The front-facing user interface was developed by using various java-based libraries and technologies, especially the JavaServer Page (JSP), a technology created to aid software development by creating dynamically generated web pages based on HTML, XML, or other document types while including the ability to perform Java commands and invoke Java methods. The JSP pages were used to run SQL queries in order to populate the relevant pages with information taken from the database (see Figure 11) while Javascript was used in order to handle certain events performed by the user and redirect them to other pages.

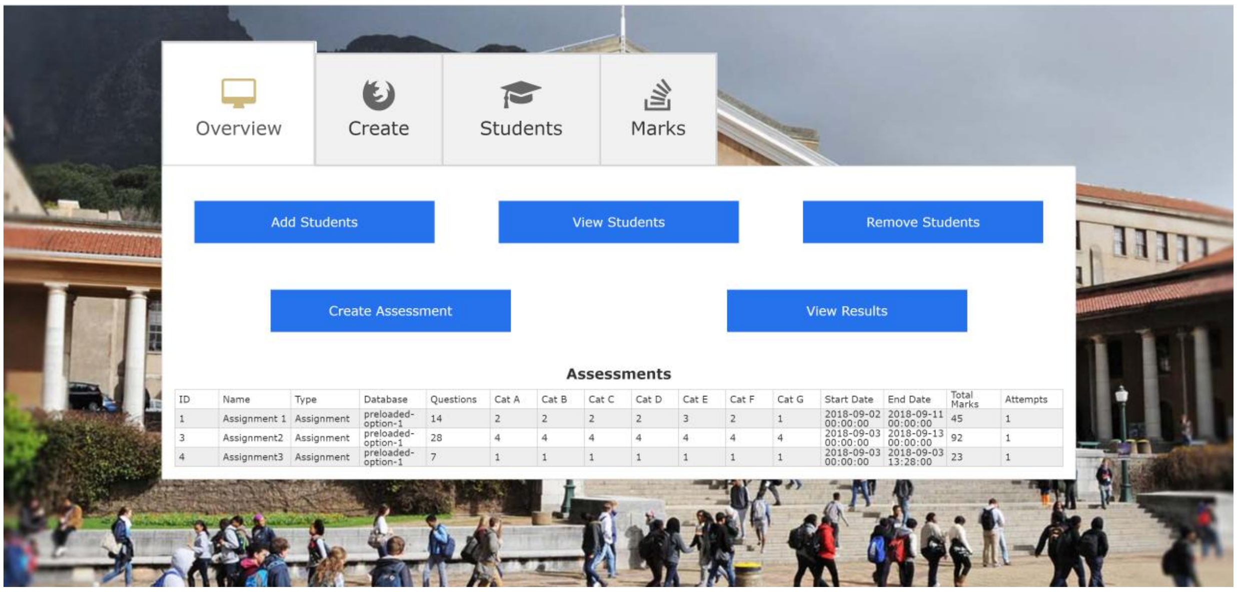
In terms of project build, an Apache Maven Web Application was chosen. Apache Maven is a build automation tool that is primarily used for Java projects and addresses two aspects of building software: it describes how software is built and it describes the project's dependencies. An XML file is used to describe the software project being built, as well as its dependencies on other external modules and components, the build order, directories, and the required plug-ins for the project to run. By using Maven, it drastically simplified the process of developing the physical web application. The user interfaces' visual design was created using Cascading Style Sheets, providing style to the JSP pages which in turn afford the user a more pleasant experience when using the system.

The system strives to adhere to the following principles of web-based design:

- Using familiar and conventional layouts
- Ensuring clarity and simplicity
- Ensuring that user tasks are completed in the most efficient way possible
- Making use of design tools such as GitHub, smart IDEs, build tools, and cloud computing

4.3. Important Relationships

- Most, if not all, Java classes and JavaServer Pages have a relationship with the MySQL database. Some use it simply to fetch information to display to the viewer, while others perform modifications (create, drop, update) to the databases. All connections to the database, however, are handled through the *SQL.java* class.

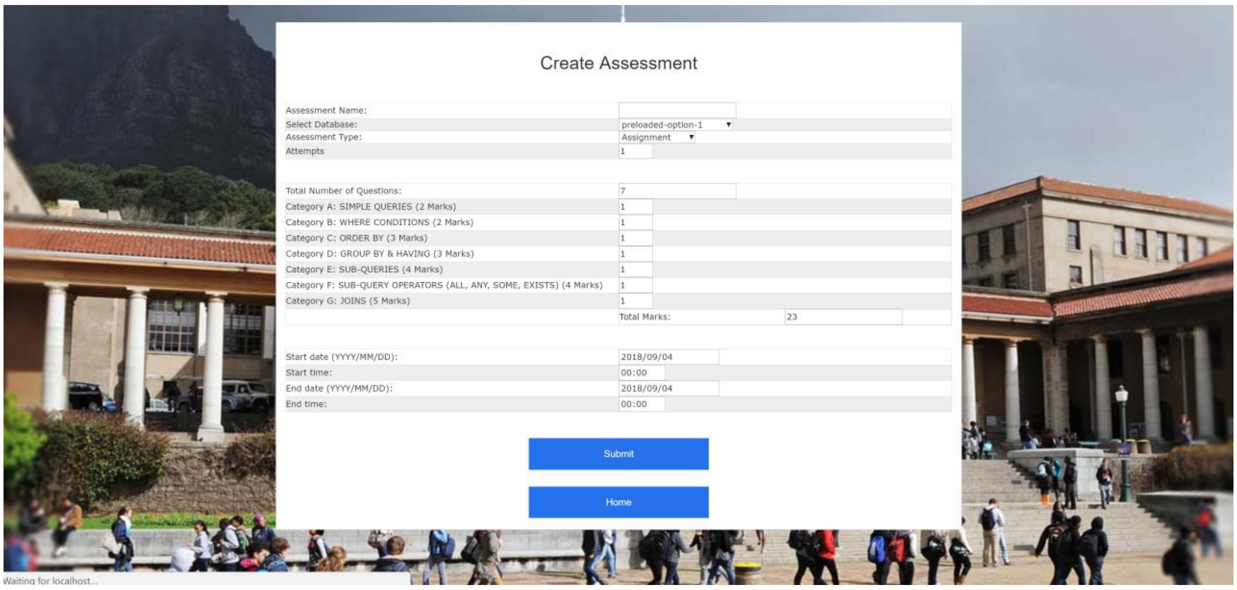


Assessments

ID	Name	Type	Database	Questions	Cat A	Cat B	Cat C	Cat D	Cat E	Cat F	Cat G	Start Date	End Date	Total Marks	Attempts
1	Assignment 1	Assignment	preloaded-option-1	14	2	2	2	2	3	2	1	2018-09-02 00:00:00	2018-09-11 00:00:00	45	1
3	Assignment2	Assignment	preloaded-option-1	28	4	4	4	4	4	4	4	2018-09-03 00:00:00	2018-09-13 00:00:00	92	1
4	Assignment3	Assignment	preloaded-option-1	7	1	1	1	1	1	1	1	2018-09-03 00:00:00	2018-09-03 13:28:00	23	1

Figure 12: Screenshot of the Staff Portal

- The *teacherPortal.jsp* (View) page contains critical relationships which are responsible for the successful use of the system (See Figure 12):
  - The page calls upon the *fileupload.jsp* (Contoller) page which facilitates the uploading of a csv file containing Student details. If the file upload is successful, *CreateStudents.java* class is used to add the students to the Users table in the *'sqlautomarker'* database.
  - The "Create" tab calls the *createAssessment.jsp* (View) page is connected to the *createAssessmentLoading.jsp* (Controller) (See Figure 13) page which then initiates the creation of an assessment through the *Assessment.java* (Model) class.
  - The "Remove Students" button calls the *removeStudents.jsp* (Controller) page, which in turn invokes the *RemoveStudents.java* (Model) class in order to remove all students from the database.

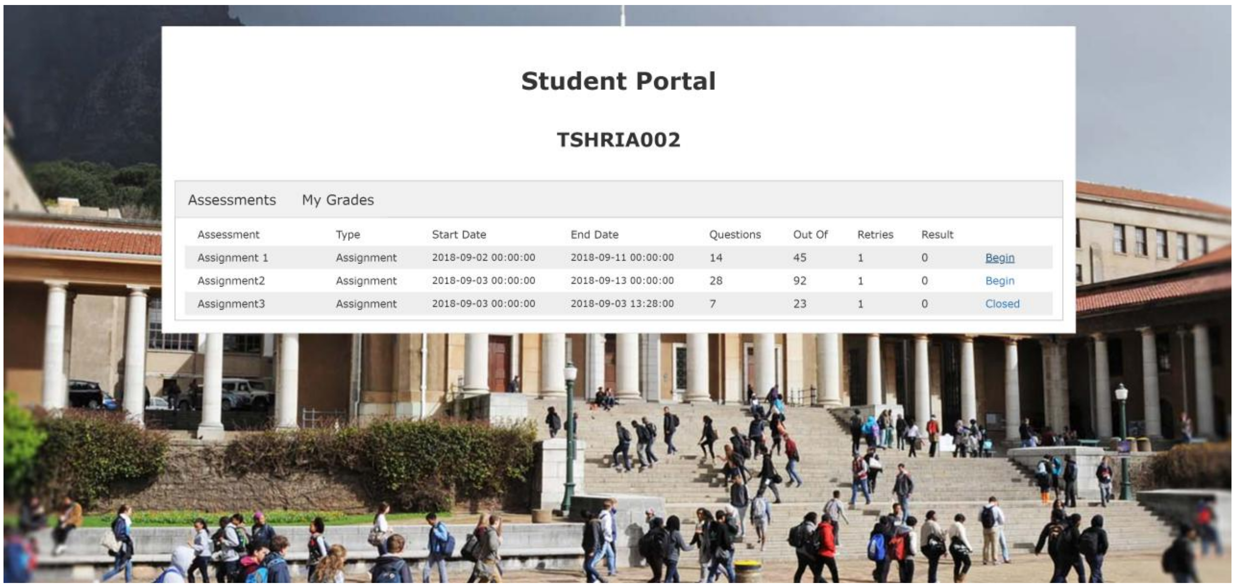


Create Assessment

Assessment Name:	
Select Database:	preloaded-option-1
Assessment Type:	Assignment
Attempts	1
Total Number of Questions:	7
Category A: SIMPLE QUERIES (2 Marks)	1
Category B: WHERE CONDITIONS (2 Marks)	1
Category C: ORDER BY (3 Marks)	1
Category D: GROUP BY & HAVING (3 Marks)	1
Category E: SUB-QUERIES (4 Marks)	1
Category F: SUB-QUERY OPERATORS (ALL, ANY, SOME, EXISTS) (4 Marks)	1
Category G: JOINS (5 Marks)	1
	Total Marks: 23
Start date (YYYY/MM/DD):	2018/09/04
Start time:	00:00
End date (YYYY/MM/DD):	2018/09/04
End time:	00:00

Figure 13: Screenshot of the Create Assessment Page

- The *studentPortal.jsp* (View) page also contains critical relationships which are responsible for the successful use of the system (See Figure 14):
  - The “Begin” link calls the *studentAnswers.jsp* (View) (Figure 15) page, provided that the assessment is open. Upon the submission of the assessment, the *studentAnswers.jsp* (View) page calls upon the *markLoading.jsp* (Controller) page which in turn calls the *markAssessment.jsp* (Conrtoller) page which uses the *MarkQuestion.java* (Model) class, which relies on the *Result.java* class in order to return the marks of the student and successfully mark the assignment.

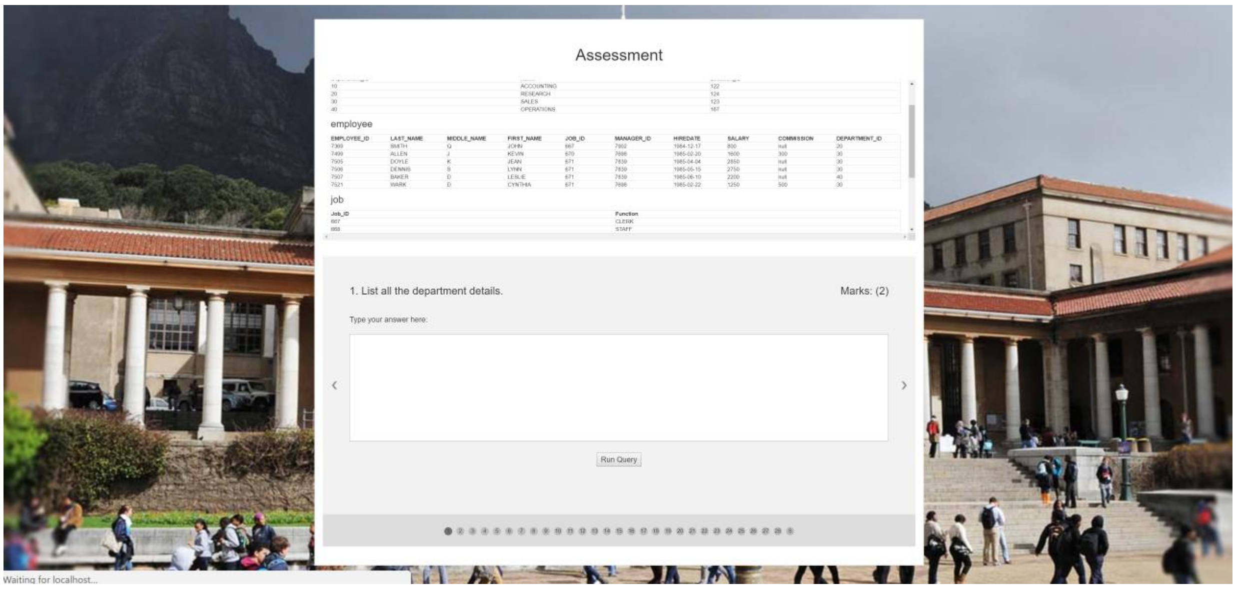

Student Portal

TSHRIA002

Assessments My Grades

Assessment	Type	Start Date	End Date	Questions	Out Of	Retries	Result	
Assignment 1	Assignment	2018-09-02 00:00:00	2018-09-11 00:00:00	14	45	1	0	Begin
Assignment2	Assignment	2018-09-03 00:00:00	2018-09-13 00:00:00	28	92	1	0	Begin
Assignment3	Assignment	2018-09-03 00:00:00	2018-09-03 13:28:00	7	23	1	0	Closed

Figure 14: Screenshot of the Student Portal Page


Assessment

10	ACCOUNTING	122
20	RESEARCH	124
30	SALES	123
40	OPERATIONS	167

employee

EMPLOYEE_ID	LAST_NAME	MIDDLE_NAME	FIRST_NAME	JOB_ID	MANAGER_ID	HIREDATE	SALARY	COMMISSION	DEPARTMENT_ID
7369	SMITH	Q	JOHN	667	7902	1984-12-17	800	null	20
7499	ALLEN	J	KEVIN	670	7698	1985-02-20	1600	300	30
7505	DOYLE	K	JEAN	671	7839	1985-04-04	2850	null	30
7506	DENNIS	S	LYNN	671	7839	1985-05-15	2750	null	30
7507	BAKER	D	LESLIE	671	7839	1985-06-10	2200	null	40
7521	WARK	D	CYNTHIA	671	7698	1985-02-22	1250	500	30

job

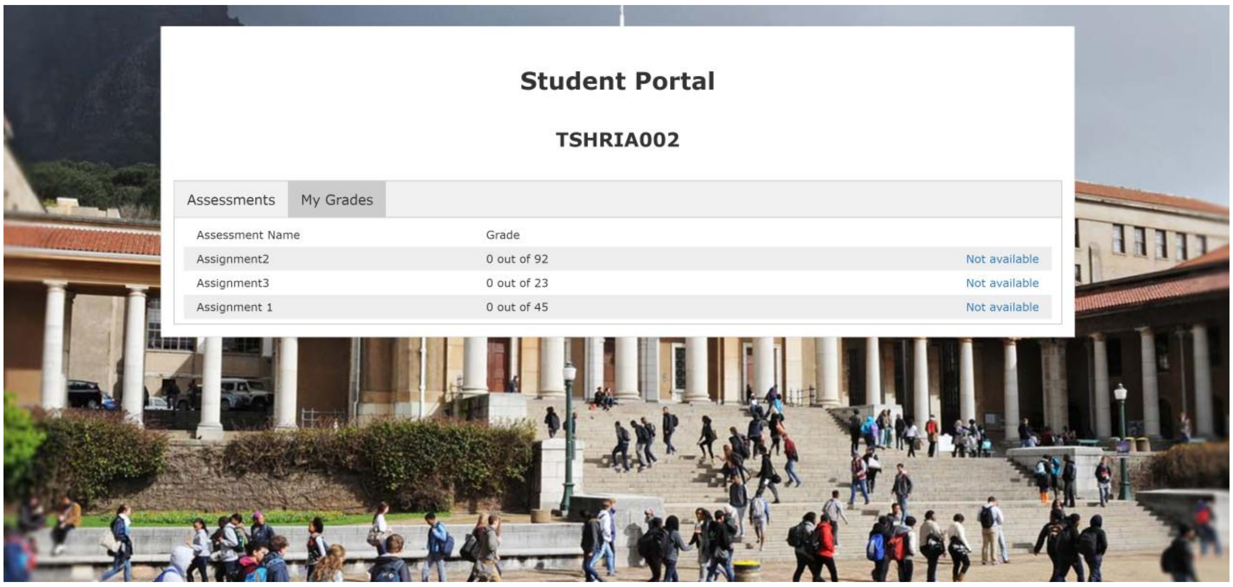
Job_ID	Function
667	CLERK
668	STAFF

1. List all the department details. Marks: (2)

Type your answer here:

Figure 15: Screenshot of the Student Answering Page

  - The “View Results” (See Figure 16) link calls upon the *viewResults.jsp* page in which the student can view the results of his/her assessment.



Assessment Name	Grade	
Assignment2	0 out of 92	Not available
Assignment3	0 out of 23	Not available
Assignment 1	0 out of 45	Not available

Figure 16: Screenshot of the ”My Grades” Tab

4.4. Special Programming Techniques and Libraries

Because the system is a Maven project, there exist dependencies upon which the system relies in order to run correctly and as intended. Maven dynamically downloads Java libraries and Maven plug-ins from repositories and stores them in a local cache. This system relies on the following dependencies:

1. Apache Commons Net – a collection of network utilities and protocol implementations. Supported protocols include: Echo, Finger, FTP, NNTP, NTP, POP3(S), SMTP(S), Telnet, Whois.

2. Apache Commons FileUpload – makes it easy to add robust, high-performance, file upload capabilities to servlets and web applications. If an HTTP request is submitted using the POST method, and with a content type of ”multipart/form-data”, then FileUpload can parse that request, and make the results available in a manner easily used by the caller.

3. Java EE Web Profile Specification APIs – streamlines the platform and enables the creation of lightweight, agile, compelling application servers with a laser focus on web application development.

4. jQuery - makes it much easier to use JavaScript on websites. It takes common tasks that require many lines of JavaScript code to accomplish, and wraps them into methods that can be called by a single line of code.

5. MySQL Connector/J – official JDBC driver for MySQL.

6. Apache Commons IO – library of utilities to assist with developing IO functionality.

7. GlassFish Server 4.1.1 – The server on which the application currently runs.

Much of the parsing of data from the MySQL database occurred through SQL statements being executed in Java. The data is received through ResultSets and when the relevant data has been gathered, the ResultSet closes.

4.5. Important Functions and Methods

- *SQL.java* is used to establish a connection to and return a connection from the MySQL database.

- *MarkQuestion.java*

- The *compareResultSets(ResultSet resultSet1, ResultSet resultSet2)* method takes both the user's and the expected SQL statement, executes them, and compares the two. If either the number of rows or columns do not match, the system checks for half marks. If the number of rows and columns do match, the system will proceed to check the entries of the database. If all entries match, the student gets full marks. If any entries do not match, the student gets no marks.
- The *checkHalfMarks()* method searches the students query for certain key phrases, or a combination of phrases. If the phrases occur, a portion of the marks are assigned to the student. The expected phrases for each category are as follows:
  - Category A: {SELECT, FROM}
  - Category B: {SELECT, FROM, WHERE}
  - Category C: {SELECT, FROM, ORDER BY}
  - Category D: {SELECT, FROM, {GROUP BY or HAVING}}
  - Category E: {SELECT, FROM, (,)}
  - Category F: {SELECT, FROM, {ALL or ANY or SOME or EXISTS}}
  - Category G: {SELECT, FROM, {WHERE or JOIN}}

- *GeneratePassword.java* generates a 10 character secure password from a selection of random alpha-numeric characters and symbols. For security reasons, the student is not allowed to change his/her password but ideally it would be accessible to him/her through Vula or stored in email.
- For additional functions, please refer to project documentation.
- Sequence diagram for Students (See Figure 17)
- Sequence diagram for Staff (Lecturers and Tutors) (See Figure 18)

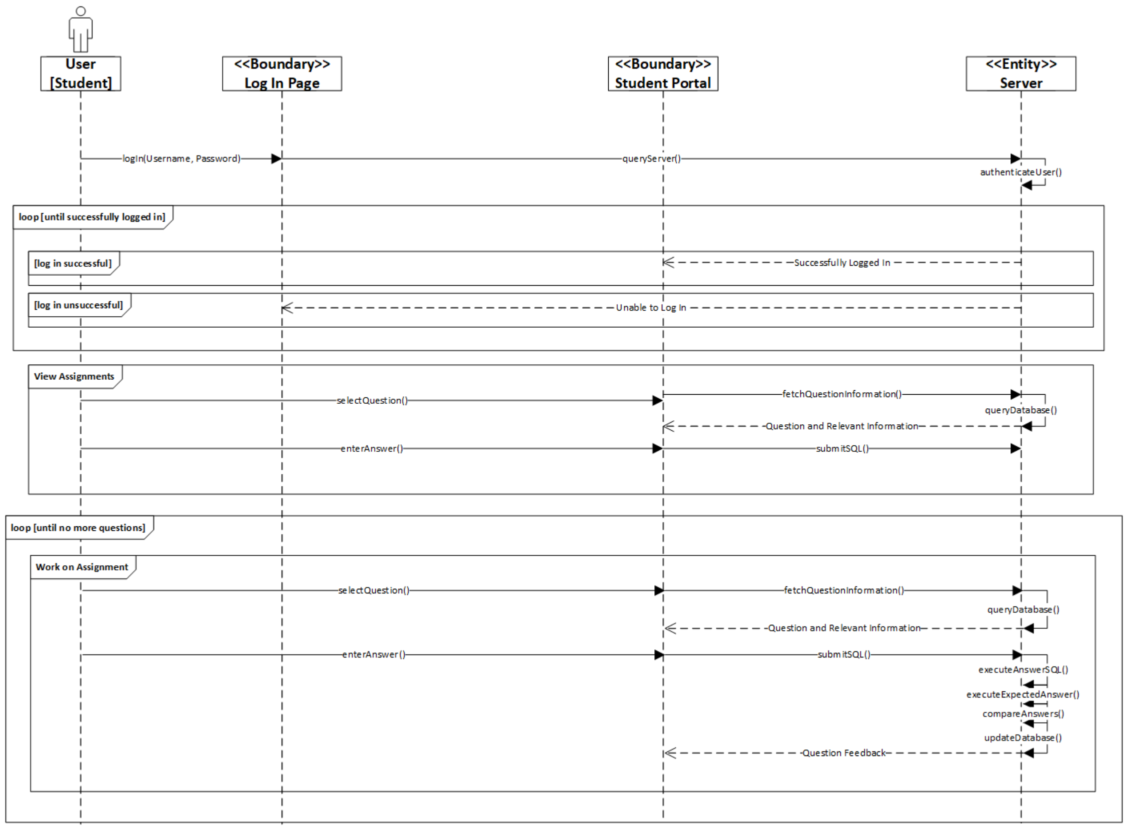


Figure 17: Student Sequence Diagram

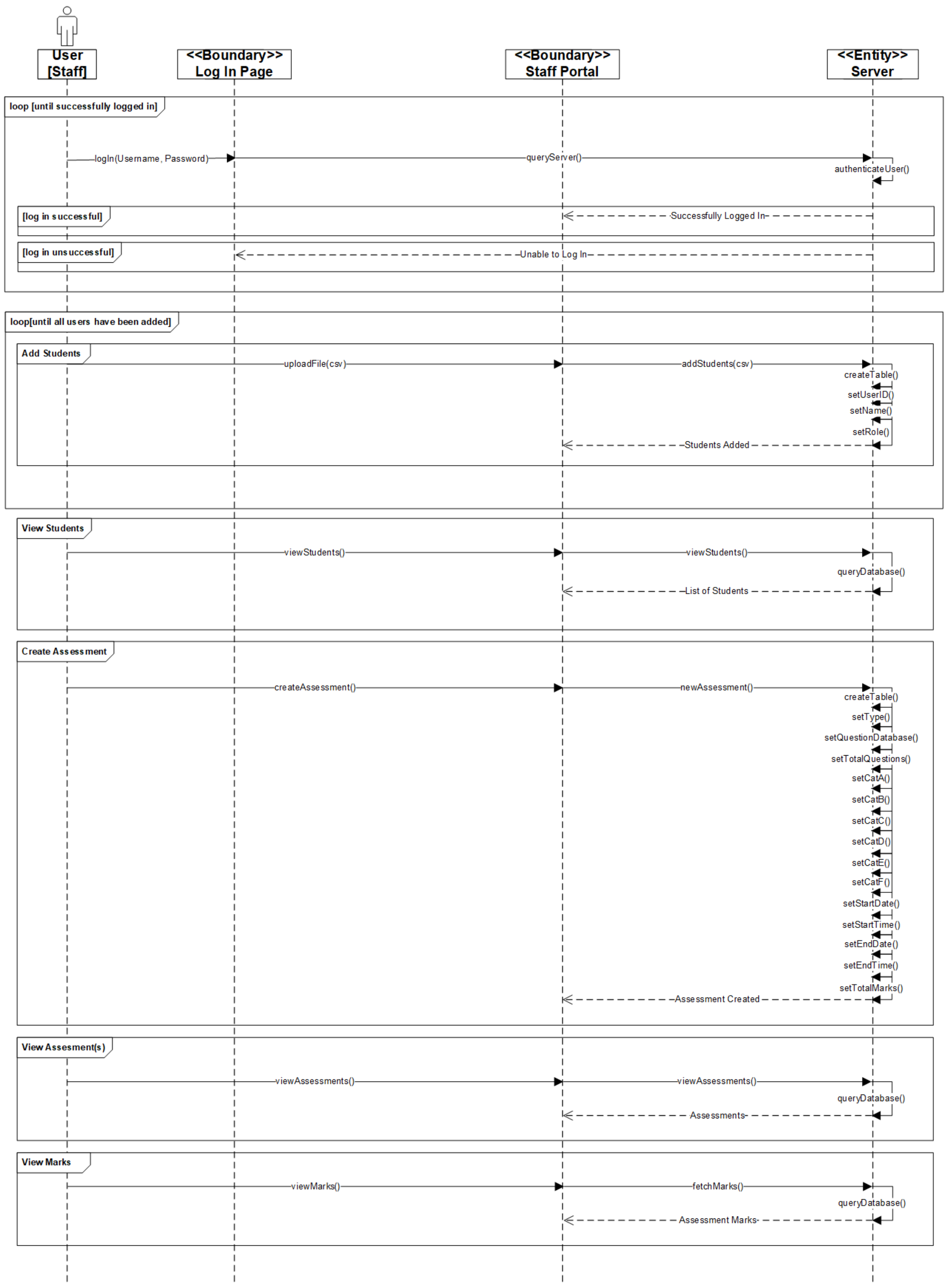


Figure 18: Staff Sequence Diagram

5. Program Validation and Verification

Because much of this system is a front-facing, interactive, user interface, most of the testing was manual and focussed on user input and their interaction with the software. Below (Table 1) is a breakdown of the testing plan used for the system.

Table 1: Summary of the SQL Automatic Marker's Testing Plan.

Process	**Technique**
1. Class Testing: Testing the methods and behaviour of Java classes and JSP files.	Random, Partition and White-Box Tests
2. Integration Testing: Testing if the Database and the Java classes interact in the manner that they should.	Random and Behavioural Testing
3. Validation Testing: Looking at whether the clients requirements and scope have been met.	Use-case based black box and Acceptance tests
4. System Testing: Testing if the the system works on multiple computers and different internet services at different locations.	Recovery, security, stress and performance tests

In addition to the tests mentioned above, additional verification and validation methods took place. Detailed in Table 2 are the test cases, including their descriptions, the reason for the choice of testing, an example of the sort of input (if applicable) for the test case, the expected behaviour of the test, as well as the test's expected output (if applicable).

Table 2: Summary of Tests Carried Out

Test Case	**Description**	**Reason for Choice of Testing**	**Input Example**	**Behaviour**	**Expected Output**
1	Attempt to log in with blank username and password fields	Security reasons. Unauthorised personnel should not be able to view data.	Username: Password:	Error message ("Please fill out all fields")	Log in Failure. Redirection to home page.
2	Attempt to log in with username but no password	Security reasons. Unauthorised personnel should not be able to view data.	Username: MLTZAC001 Password:	Error message ("Please fill out all fields").	Log in Failure. Redirection to home page.
3	Attempt to log in with no username but password		Username: Password: dcnsdcd	Error message ("Please fill out all fields").	Log in Failure. Redirection to home page.

Continued on next page

Table 2 – *Continued from previous page*

Test Case	**Description**	**Reason for Choice of Testing**	**Input Example**	**Behaviour**	**Expected Output**
4	Log in with incorrect username and incorrect password.	Security reasons. Unauthorised personnel should not be able to view data.	Username: Cat Password: 1234567	Error message ("invalid username or password")	Login Failure. Redirection to home page.
5	Login with incorrect username and correct password.	Security reasons. Unauthorised personnel should not be able to view data.	Username: Cat Password :1848@sEt01	Error message ("invalid username or password").	Login Failure. Redirection to home page.
6	Login with correct staff number and correct password.	To determine if the staff member gets redirected to the Staff Portal and gets access to the necessary functions ascribed to the user.	Username: 014578969 Password :1848@sEt01	Redirection to Staff Portal.	Successful log in. Redirection to Staff Portal.
7	Enter a valid student number and valid password.	To check if a student gets redirected to the Student' Portal and gets access to the necessary functions ascribed to the user.	Username: TSHRIA002 Password 12345	Redirection to Student Portal.	Successful log in. Redirection to Student Portal.
8	Incorrectly adding users by uploading an incorrectly formatted .csv file	To ensure that if incorrect data does not get inserted into the database and corrupt the existing data.	file.csv	Error message ("File format error, please read documentation on how file must be formatted").	No students added. Redirect to previous page.
9	Correctly adding users by uploading correctly formatted .csv file	To ensure that the correctly formatted data is inserted successfully and correctly into the database.	file.csv	Alert("Users successfully added").	Students added. Redirect to previous page.

Continued on next page

Table 2 – *Continued from previous page*

Test Case	**Description**	**Reason for Choice of Testing**	**Input Example**	**Behaviour**	**Expected Output**
10	Submitting the "add users" form without selecting a file	To ensure that there is no fall over of the class assigned with adding users to the database		Error message ("No file selected").	
11	Create assessment without a name	To ensure that an assessment without a name identifier cannot be created.	Assessment Name:	Error message ("Please provide a name for the assessment")	
12	Create an assessment with no marks.	To ensure that every assessment created is measurable and that no students are assigned empty assessments.	Category A: 0 Category B: 0 Category C: 0 Category D: 0 Category E: 0 Category F: 0 Category G: 0	Error message (" Assessment has to have at least one question")	
13	Create an assessment with the end date and time occurring before the start date and time.	To ensure that the logic of creating assessment is correct.	Start Date: 2018/09/05 Start Time: 00:00 End Date: 2018/09/04 End Time: 00:00	Error message (" Start date cannot be greater than the end date")	

Continued on next page

Table 2 – *Continued from previous page*

Test Case	**Description**	**Reason for Choice of Testing**	**Input Example**	**Behaviour**	**Expected Output**
14	Correctly creating an assessment.	To ensure that the system behaves as expected, and that the assessment appears on the view assessment list.	Assessment Name: Assignment 2 Select Database: preloaded-option-1 Assessment Type: Assignment Attempts: 1 Category A: 1 Category B: 1 Category C: 1 Category D: 1 Category E: 1 Category F: 1 Category G: 1 Start date (YYYY/MM/DD): 2018/09/04 Start time: 00:00:00 End date (YYYY/MM/DD): 2018/09/05 End time: 00:00:00	Alert ("Assessment has been successfully created").	Assessment created. Redirect to Staff Portal.
15	Answering the assignment questions with wrong answers.	To determine if the grade-allocation algorithm gets allocated correctly.	SELECT * from CAR	SQL error (e.g " No Such Table error") is generated.	If category A: Half marks allocated Else: No marks allocated
16	Answering the assignment questions with wrong answers.	To determine if the grade-allocation algorithm gets allocated correctly.	SELECT type FROM Animal	No SQL error, "Output Not Correct" is shown.	If Category A: Half marks allocated for the question Else: No marks allocated.
17	Student answering the assignment questions with correct answers	To determine if the grade-allocation algorithm gets allocated correctly.	SELECT Name FROM Animal;	"Output Correct" message is shown.	Full marks allocated for the question.

Continued on next page

Table 2 – *Continued from previous page*

Test Case	**Description**	**Reason for Choice of Testing**	**Input Example**	**Behaviour**	**Expected Output**
18	Student abruptly exits assessment .			Student's machine state resets on exit so that if the student accesses it again within the given time frame, (s)he will have to restart the assessment to discourage cheating.	Restarting the assessment .
19	Student attempts to retry the assessment and there are attempts left.	To ensure that the assessment reopens with different questions assigned to the student.		Recreate a new assessment based on the assessment parameters and display it to the student.	New Assessment Answering page
20	Student attempts to retry the assessment but there are no attempts left.	To ensure that the assessment does not get reopened if the student has exhausted his/her tries.		The link to take the assessment is disabled and is labelled "Closed."	Assignment cannot be opened.
21	Assessment created but start date has not occurred yet	To ensure that the assessment opens in the time that it is assigned.	Current date: 2018/09/04 Current time: 16:00:00 Start date: 2018/09/04 Current time: 15:59:00	The link to take the assessment is disabled and is labelled "Closed."	Assignment is cannot be opened.
22	Assessment created but end date has elapsed.	To check that if the deadline has lapsed, the assessment closes.	Current date: 2018/09/04 Current time: 16:00:00 End date: 2018/09/04 End time: 15:59:00	The link to take the assessment is disabled and is labelled "Closed."	Assignment is cannot be opened .

Continued on next page

Table 2 – *Continued from previous page*

Test Case	**Description**	**Reason for Choice of Testing**	**Input Example**	**Behaviour**	**Expected Output**
23	User attempts a test query during assessment with an erroneous SQL statement.	To determine if the query system can handle incorrect input.	Input: ksnx-askjjxk-snxjkasjx;e23222;	Output: SQL Error.	SQL Error.
24	User attempts a test query during assessment with a correct SQL statement.	To determine if the query system can handle correct input.	Input: SELECT * FROM employee;	Output: A table of all employees.	Table of all employees.

After conducting tests of the type mentioned above, the software was adjusted to try and prevent needless errors from occurring. These kinds of prevention tactics came in the form of alert dialogue boxes, confirmation boxes, and other safeguards that were put in place to try and mitigate the possibility of human error. These tests, combined with the iterative approach to risk identification and mitigation resulted in a usable system. This is not to say that it is without error. It merely means that the possibility for error has been significantly reduced. The only errors that may arise are from attempting tasks that deviate significantly from the expected steps that need to be taken as laid out in the user manual. Such tasks can include but are not limited to:

- Uploading duplicates of users. This system was designed for a single class at a time. It is therefore common sense that each user will have a unique username and password.
- Deletion of records and tables from the MySQL Workbench. If, say for instance, the user wishes to remove all individual records from the system, he/she would be required to remove the corresponding assessment entry from the *assessments table* in the *sqlautomarker* database to avoid unwanted errors when students try to access their assessment.

Ultimately, if the users adhere to the expected way by which the system is expected to function, the likelihood of encountering errors is extremely low.

6. Conclusion

The SQL Automatic Marker aimed to create a system which would assign and automatically mark Structured Query Language database assessments for the University of Cape Town's Computer Science Department. The system was created to enhance the process of marking these assessments in comparison to the current method of manual marking. The developed application followed a layered Architectural design and was developed in a hybridised form between iterative methodologies and more traditional approaches. By using evolutionary prototyping, the development of the system was made easier due to the nature of this kind of prototype: it allowed real-time troubleshooting and provided the clients with a view of the current state of the system in order to gauge whether or not their scope was met.

Unfortunately, due to time constraints, two elements were left out of the current model:

1. The ability to upload CSV files containing data for schema creation and question assignment.
2. The automatic generation of SQL questions and answers based on a given set of defined questions.

The uploading of a CSV file proved to be difficult because of the ambiguities which presented themselves during the designing of the method. In order to create a table, the variables needed to be defined (VARCHAR, INT, DOUBLE, etc.) and the layout of the csv file would have to adhere to an unreasonably strict format. Because of this, the decision was made to include the MySQL Workbench as part of the Staff Portal (even though it is accessed through a different program). By integrating the user interface to include the MySQL Workbench, it is arguably easier for the staff member to create a new database and import CSV information directly into the system rather than having to interact with a user interface.

While the aforementioned elements were left out of the final design, there are still some additional features such as the ability to export assessment mark summaries or the individual's assessment summary, the installation of pre-existing datasets that can be reused or alternated over years, the ability for students to run queries to test their answer before submitting, the storing of individual assessment files in order to be able to moderate marks, and a pleasing, easy-to-understand user interface to give the user a more pleasant experience when using the application, just to name a few.

Over all, the system works as intended. Staff are afforded different functionalities from students, which is evident in their interface. They can assign assessments, view grades, add, remove, or view students while students can view their grades and take assessments. Through this application, the hope is that students are able to interact and engage with the Structured Query Language through the system's interface in order to gain a new knowledge and understanding of said language while at the same time making it easier on the tutors and lecturers who already have heavy workloads when it comes to marking what can often be complex and tedious queries.

CSC3003S Capstone Project: SQL Automatic Marker Appendix A: User Manual

This is the user manual for the SQL Automatic Marker. This is used to aid in the user's understanding and navigation of the SQL Automatic Marker system, including the User Interfaces and the MySQL Workbench in order to fully utilize and enjoy the SQL Automatic Marker.

Contents

1. System Overview

The SQL Automatic Marker is an application that is intended to create, assign, take, and mark Structured Query Language assessments. A Lecturer or Tutor creates an assessment and the system assigns all enrolled students an assessment based on the parameters stipulated when creating the assessment. In addition to creating assessments, the Lecturer or Tutor can add, remove, and view students, as well as view assignment mark summaries or individual students' marks. The student can take assessments as well as view their marks, given that they have submitted an assessment. Upon completion of the assignment, the system returns the student's mark along with some indication of how their output differs from expected output as well as the marks that they received. The assessments and the data to being queried is stored in a MySQL database which is accessible to the lecturer or tutors to add to or remove from as they please.

SQL Automaker operates on desktop PC and laptop devices, regardless of operating systems. The application requires a connection to internet in order to be able to access and commit to the database. Users have to be a UCT student/stuff that has been assigned to the system by the Computer Science Department.

2. SQL Automatic Marker

2.1. Logging In

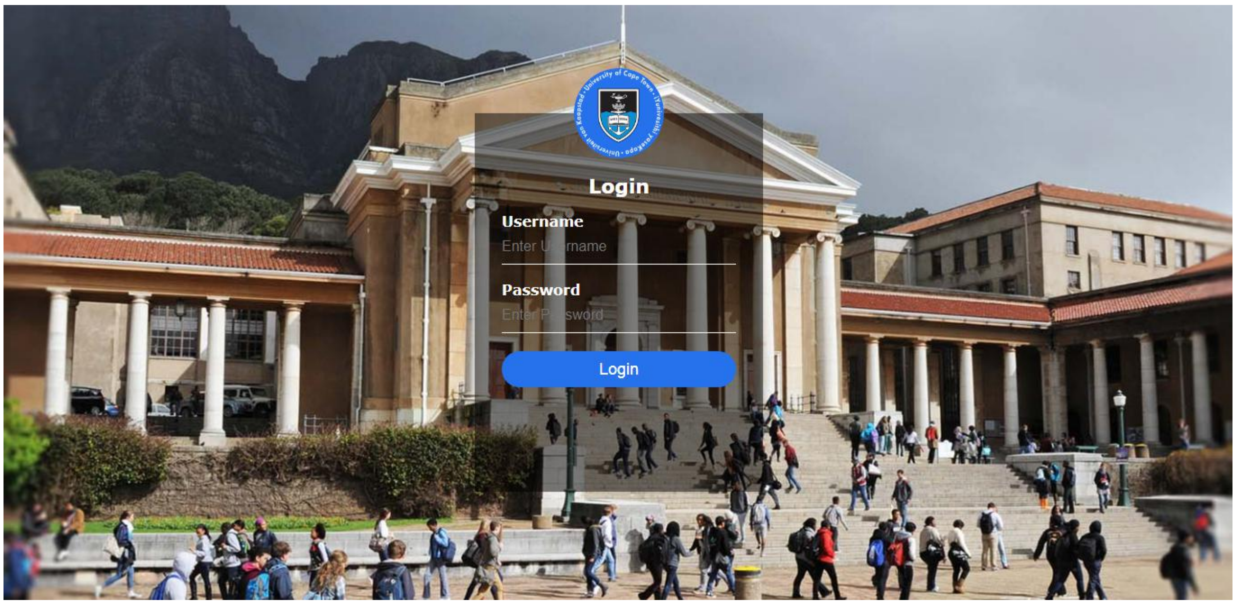


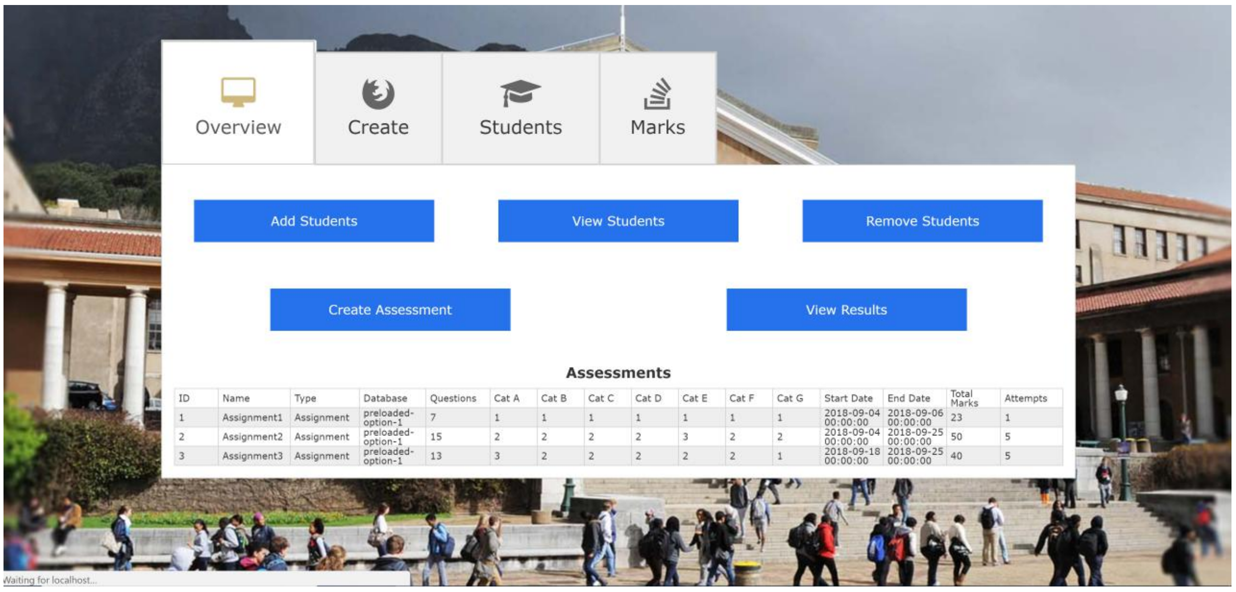
Figure 1: The Log In Portal

To log in, enter Student Number or Employee Number under “Username”. Under “Password,” enter the secure password that has been assigned to you by the system.

After entering user credentials, press the “Login” button. If the username and log in credentials match those on file, the user will be redirected to either the Student Portal (Figure 12) or the Staff Portal (Figure 2), dependent on their role.

2.2. Staff Portal

The Staff Portal is set out in tabular form with its landing page being the “Overview” tab (See Figure 2). Other tabs relate to other functionalities provided by the system.



Assessments

ID	Name	Type	Database	Questions	Cat A	Cat B	Cat C	Cat D	Cat E	Cat F	Cat G	Start Date	End Date	Total Marks	Attempts
1	Assignment1	Assignment	preloaded-option-1	7	1	1	1	1	1	1	1	2018-09-04 00:00:00	2018-09-06 00:00:00	23	1
2	Assignment2	Assignment	preloaded-option-1	15	2	2	2	2	3	2	2	2018-09-04 00:00:00	2018-09-25 00:00:00	50	5
3	Assignment3	Assignment	preloaded-option-1	13	3	2	2	2	2	2	1	2018-09-18 00:00:00	2018-09-25 00:00:00	40	5

Figure 2: The ”Overview” Tab in the Staff Portal

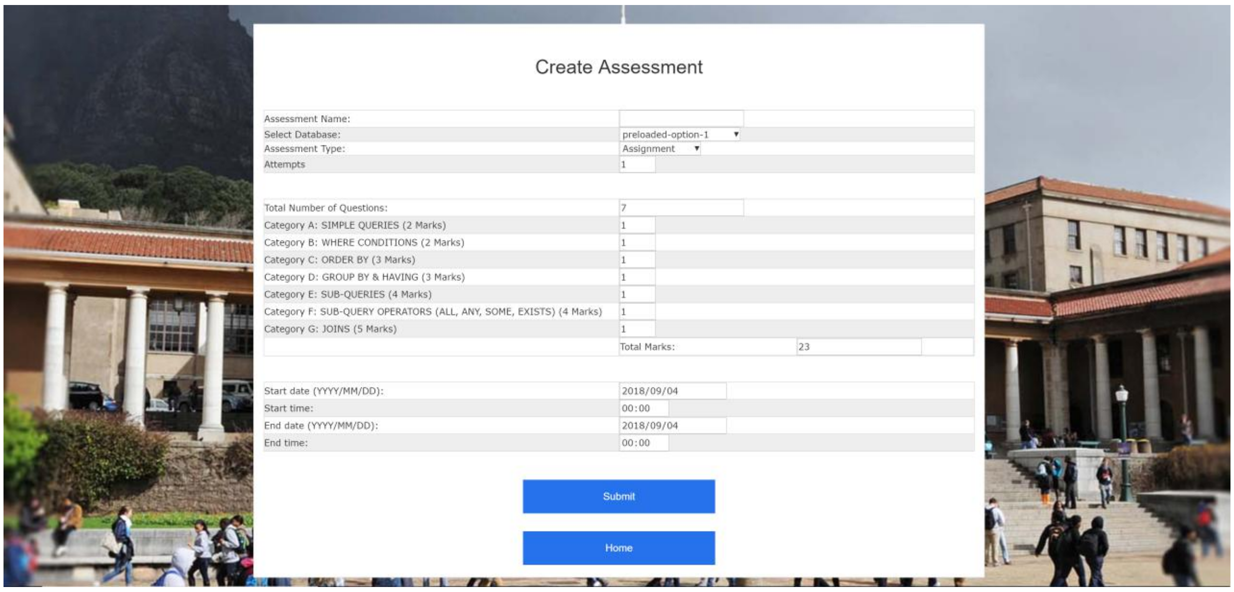
2.2.1. Overview Tab

The overview tab (Figure 2) aims to provide the lecturer or tutor an easy interface by offering shortcuts to other functions of the Staff Portal:

- Add Students directs the staff member to the "Add Students" tab.
- View Students opens a pop up window that displays all of the users in the system.
- Remove students directs the staff member to the "Remove Students" tab.
- Create Assessment redirects the staff member to the "Create Assessment" page.
- View Results directs the staff member to the "Marks" tab.

Underneath the shortcuts, an overview of all assignments is available for the staff member's reference.

2.2.2. Create Assessment Tab



Create Assessment

Assessment Name:		
Select Database:	preloaded-option-1	
Assessment Type:	Assignment	
Attempts	1	
Total Number of Questions:	7	
Category A: SIMPLE QUERIES (2 Marks)	1	
Category B: WHERE CONDITIONS (2 Marks)	1	
Category C: ORDER BY (3 Marks)	1	
Category D: GROUP BY & HAVING (3 Marks)	1	
Category E: SUB-QUERIES (4 Marks)	1	
Category F: SUB-QUERY OPERATORS (ALL, ANY, SOME, EXISTS) (4 Marks)	1	
Category G: JOINS (5 Marks)	1	
	Total Marks:	23
Start date (YYYY/MM/DD):	2018/09/04	
Start time:	00:00	
End date (YYYY/MM/DD):	2018/09/04	
End time:	00:00	

Figure 3: The "Create Assessment" Tab in the Staff Portal

2.2.2.1. Creating an Assessment

In the Create Assessment page (Figure 3), you are given a form to complete. Once the form is completed and you have clicked "Submit," an assessment will be created and assigned to each user in the system. If the "Home" button is pressed, the staff member will be redirected back to the Staff Portal Overview page.

Expected inputs are defined in the table below:

Table 1: Expected Input When Creating an Assessment

Field	Description
Asessment Name:	The name of the assessment. E.g. Assignment 1
Select Database:	The database from which the questions will be coming from. E.g. preloaded-option-1
Assessment Type:	The type of assessment. It can either be an "Assignment" or a "Closed Practical"

Continued on next page

Table 1 – *Continued from previous page*

Test Case	**Description**
Attempts:	The number of attempts the student has to complete the assessment. The default and minimum value is 1.
Total Number of Questions:	This is an read-only field that updates the total number of questions when a value is changed.
Category A: SIMPLE QUERIES (2 Marks)	The number of questions to be assigned from Category A.
Category B: WHERE CONDITIONS (2 Marks)	The number of questions to be assigned from Category B.
Category C: ORDER BY (3 Marks)	The number of questions to be assigned from Category C.
Category D: GROUP BY & HAVING (3 Marks)	The number of questions to be assigned from Category D.
Category E: SUB-QUERIES (4 Marks)	The number of questions to be assigned from Category E.
Category F: SUB-QUERY OPERATORS (ALL, ANY, SOME, EXISTS) (4 Marks)	The number of questions to be assigned from Category F.
Category G: JOINS (5 Marks)	The number of questions to be assigned from Category G.
Total Marks:	A read-only field that automatically updates the total mark value to correspond with the number of questions and the categories that they are from.
Start date (YYYY/MM/DD):	The start date of the assessment.
Start time:	The start time of the assessment.
End date (YYYY/MM/DD):	The end date of the assessment.
End time:	The end time of the assessment.

2.2.2.2. Category Types

SQL questions are stored in 7 categories; A-G. Each category is afforded a different mark weighting and increase in difficulty with A being the easiest and G being the most difficult.

Listed below are some examples of these categories taken from preloaded-option-1:

- Category A: SIMPLE QUERIES (2 Marks)
```
SELECT *
FROM employee;
```
- Category B: WHERE CONDITIONS (2 Marks)
```
SELECT
    *
FROM
    employee
WHERE
    last_name = 'SMITH';
```
- Category C: ORDER BY (3 Marks)
```
SELECT
    employee_id, last_name
FROM
```

```
        employee
ORDER BY employee_id;
```

- Category D: GROUP BY & HAVING (3 Marks)

```
SELECT
        DATE_FORMAT(hiredate, '%Y') Year, COUNT(*)
FROM
        employee
GROUP BY DATE_FORMAT(hiredate, '%Y')
HAVING Year = 1985;
```

- Category E: SUB-QUERIES (4 Marks)

```
SELECT
        *
FROM
        employee
WHERE
        department_id = (SELECT
                department_id
        FROM
                department
        WHERE
                location_id = (SELECT
                        location_id
                FROM
                        location
                WHERE
                        regional_group = 'Dallas'));
```

- Category F: SUB-QUERY OPERATORS (ALL, ANY, SOME, EXISTS) (4 Marks)

```
SELECT
        employee_id, last_name, department_id
FROM
        employee e
WHERE
        NOT EXISTS( SELECT
                department_id
        FROM
                department d
        WHERE
                d.department_id = e.department_id);
```

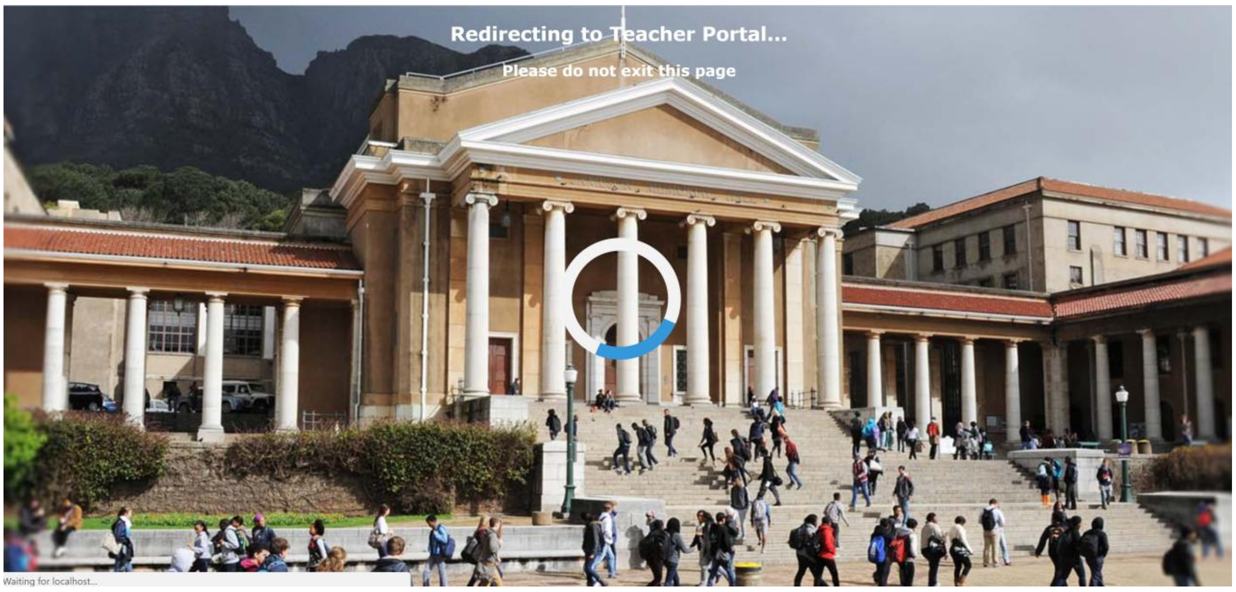
- Category G: JOINS (5 Marks)

```
SELECT
        employee_id, last_name, name, regional_group
FROM
        employee e,
department d,
        location l
WHERE
        e.department_id = d.department_id
                AND d.location_id = l.location_id;
```

2.2.2.3. Notes When Creating Assessments

1. Depending on the number of students in the class, the creation of assessments can take time if the computer does not have enough processing power to do so quickly. As a result, please wait for the redirection page (Figure 4) to display before performing any actions such as going back or exiting the page. If you do exit the page, the assessment will still be created but you will not be notified of its completion unless the redirection page is displayed.

2. If the assessment name is left blank, you will be alerted that you will not be able to create the assessment.
3. If all categories are set to 0, you will be alerted that you will not be able to create the assessment.
4. If the start date of the assessment occurs after the end date of the assessment, you will be alerted that you will not be able to create the assessment.
5. If there is already an assignment with the name that is the same as the new one entered, you will be alerted that you will not be able to create the assessment.


Redirecting to Teacher Portal...

Please do not exit this page

Waiting for localhost...

Figure 4: The ”Create Assessment” Redirection Page in the Staff Portal

2.2.3. Student Tab

Clicking on this tab directs the user to the “Student Overview” (Figure 5) page whereby the staff member can view, add, or remove students.

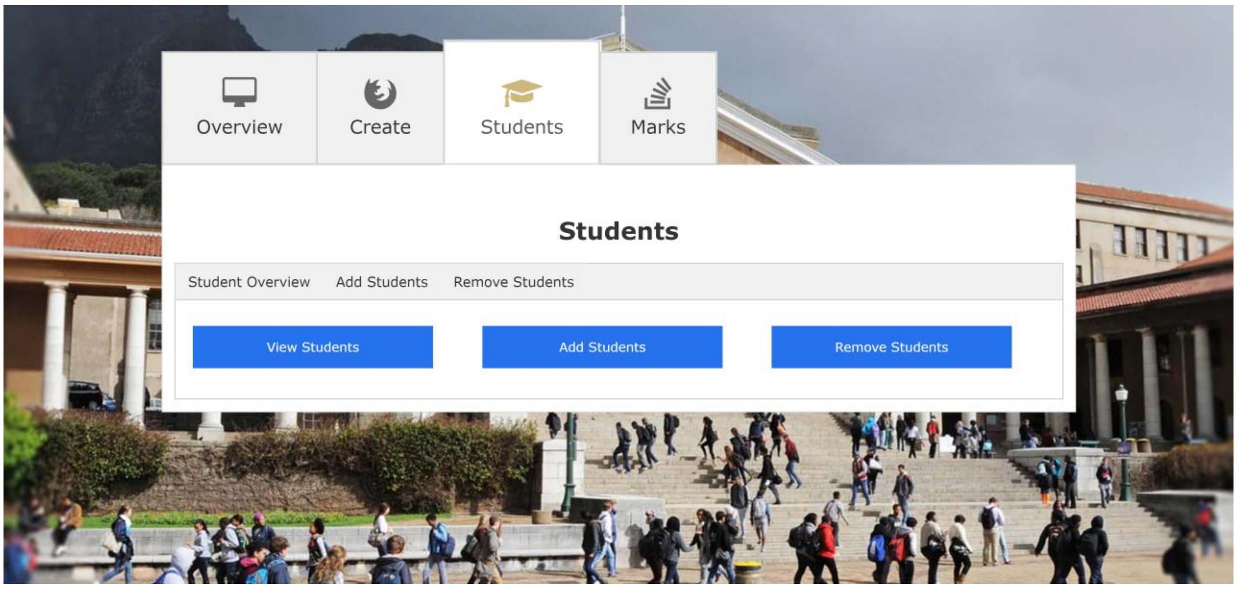
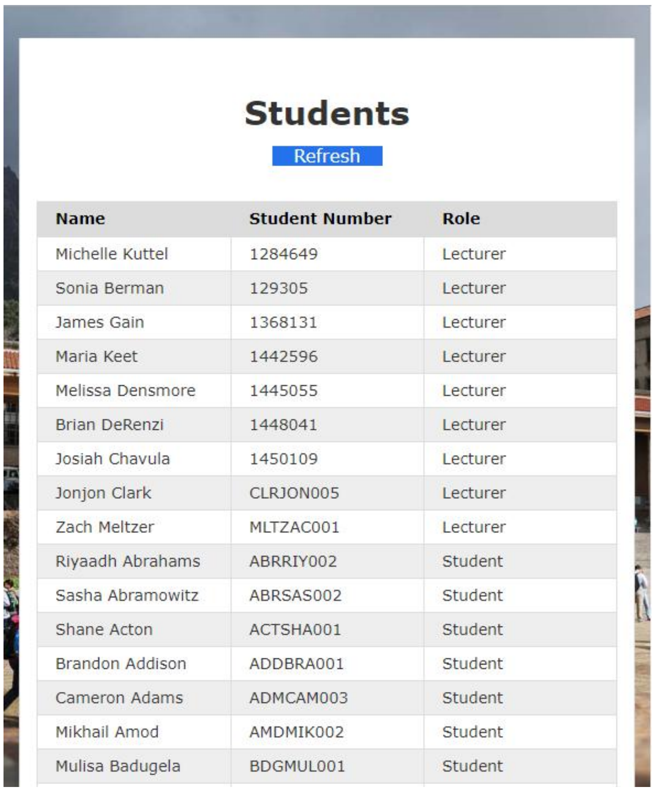


Figure 5: The ”Student Overview” Tab in the Staff Portal

2.2.3.1. Viewing Students

A summary of users (Figure 6) will display in a popup window when the “View Students” button or tab is selected.



Name	Student Number	Role
Michelle Kuttel	1284649	Lecturer
Sonia Berman	129305	Lecturer
James Gain	1368131	Lecturer
Maria Keet	1442596	Lecturer
Melissa Densmore	1445055	Lecturer
Brian DeRenzi	1448041	Lecturer
Josiah Chavula	1450109	Lecturer
Jonjon Clark	CLRJON005	Lecturer
Zach Meltzer	MLTZAC001	Lecturer
Riyaadh Abrahams	ABRRIY002	Student
Sasha Abramowitz	ABRSAS002	Student
Shane Acton	ACTSHA001	Student
Brandon Addison	ADDBRA001	Student
Cameron Adams	ADMCAM003	Student
Mikhail Amod	AMDMIK002	Student
Mulisa Badugela	BDGMUL001	Student

Figure 6: The ”View Student” Page

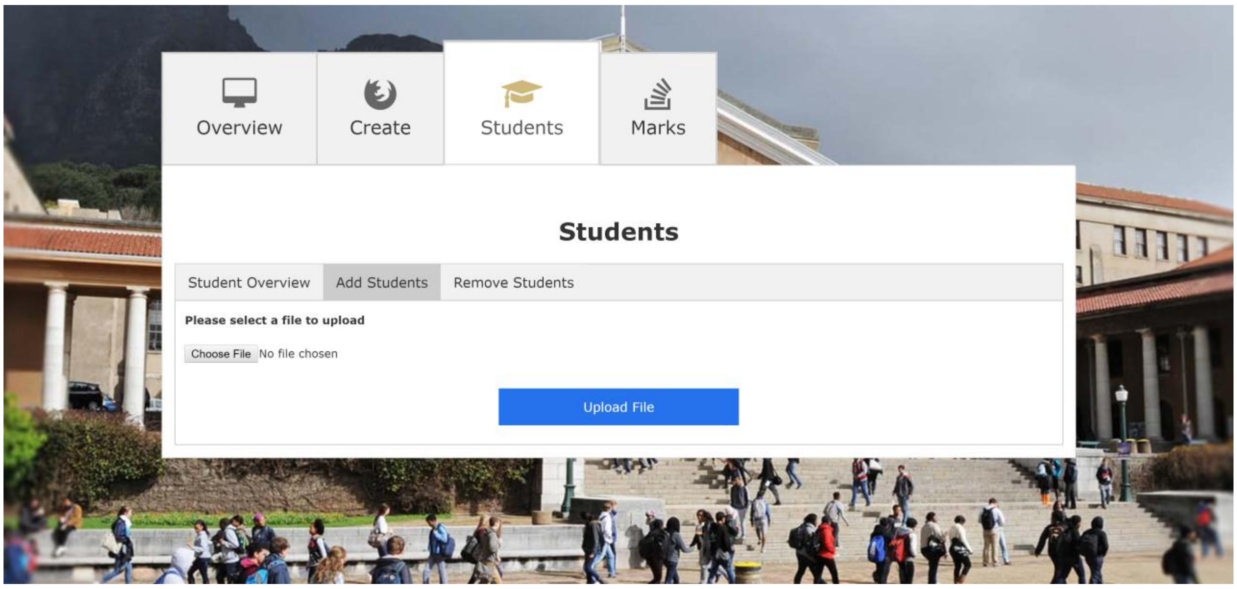


Figure 7: The "Add Student" Tab

2.2.3.2. Adding Students

To add students to the system, select the "Add Students" tab or button (Figure 7). You will be prompted to select a file for upload. Please ensure that this file is a text file or csv file which adheres to the following format:

Surname, Name; User ID (Student or Staff Number); Role

E.g.

```
Surname, Name; User ID (Student or Staff Number); Role
Berman, Sonia;129305;Lecturer
Chavula, Josiah;1450109;Lecturer
Clark, Jonjon;clrjon005;Lecturer
Densmore, Melissa;1445055;Lecturer
DeRenzi, Brian;1448041;Lecturer
Safla, Aslam;1457918;Lecturer
Stewart, Gary;1360406;Lecturer
Suleman, Hussein;1375526;Lecturer
Abrahams, Riyaadh;abrriy002;Student
Abramowitz, Sasha;abrsas002;Student
Acton, Shane;actsha001;Student
Adams, Cameron;admcam003;Student
Addison, Brandon;addbra001;Student
Amod, Mikhail;amdmik002;Student
Badenhorst, Alec;bdnale004;Student
Badugela, Mulisa;bdgmul001;Student
Baijnath, Alka;bjnalk001;Student
Beatham, James;bthjam006;Student
Behm, Dawid;bhmdaw001;Student
Bossi, Dino;bssdin001;Student
Bourn, Jess;brnjes018;Student
Brodie, Sean;brdsea003;Student
Changfoot, Jakon;chnjak001;Student
```

Please note that the separator is a semi-colon (;), not a comma (,).

2.2.3.2.1. Notes for Adding Students

1. If no file is selected and the "Upload File" is selected, the user will be alerted that there is no file selected and the upload will not take place.

2. If the file is of the incorrect format, the user will be alerted of an error and the upload will not take place.
3. During the upload of users, a secure password is generated for and assigned to each user. Please note that the only way to change these will be through MySQL itself and will be discussed later on. The idea is that users would be able to use their Vula credentials to sign into their respective portals.
4. At the completion of the upload, the user will be alerted that it is complete. If the user closes the tab, they will not be notified.
5. This system was designed to cater to single classes and therefore does not anticipate duplicate records.

2.2.3.3. Removing Students

This function removes all students from the User's database. If you would like to remove individual members or staff members, it will have to be done through the MySQL Workbench.

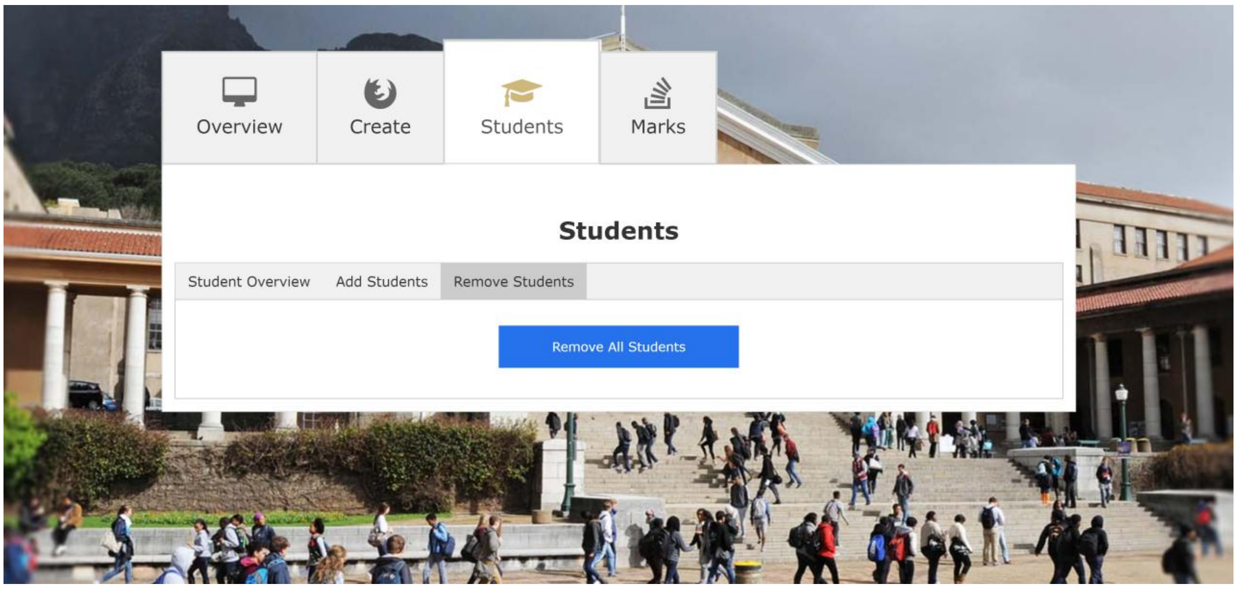
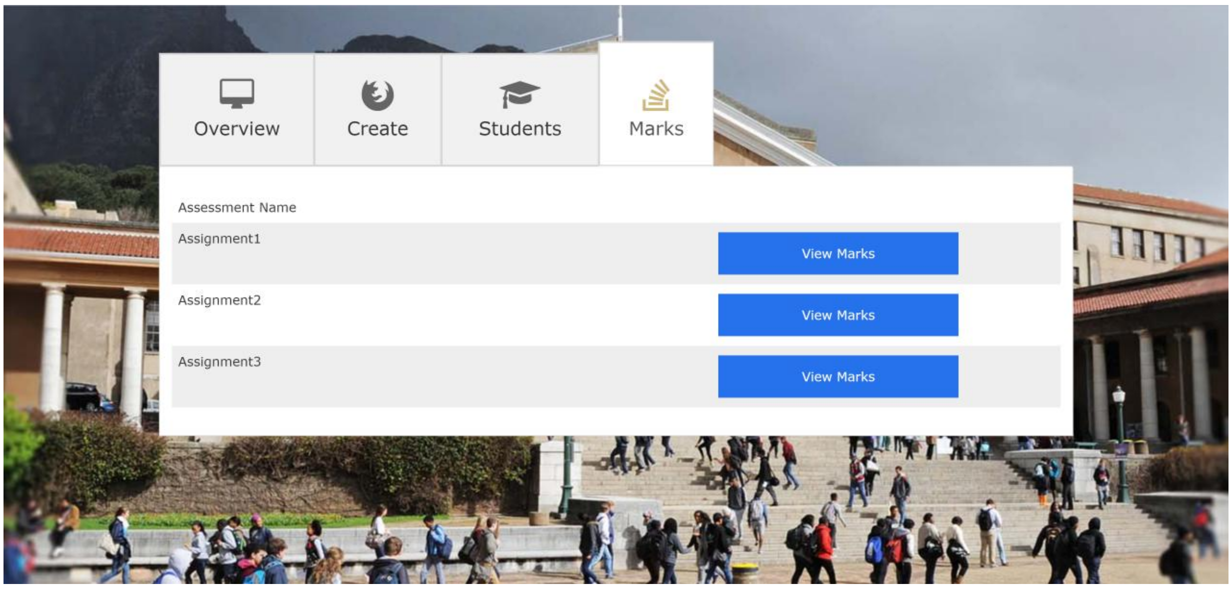


Figure 8: The ”Remove Student” Tab

2.2.4. Marks Tab

Clicking on this tab directs the user to the “Marks” tab (Figure 9) whereby the staff member can view a mark summary of an assessment or the individual student's answers.

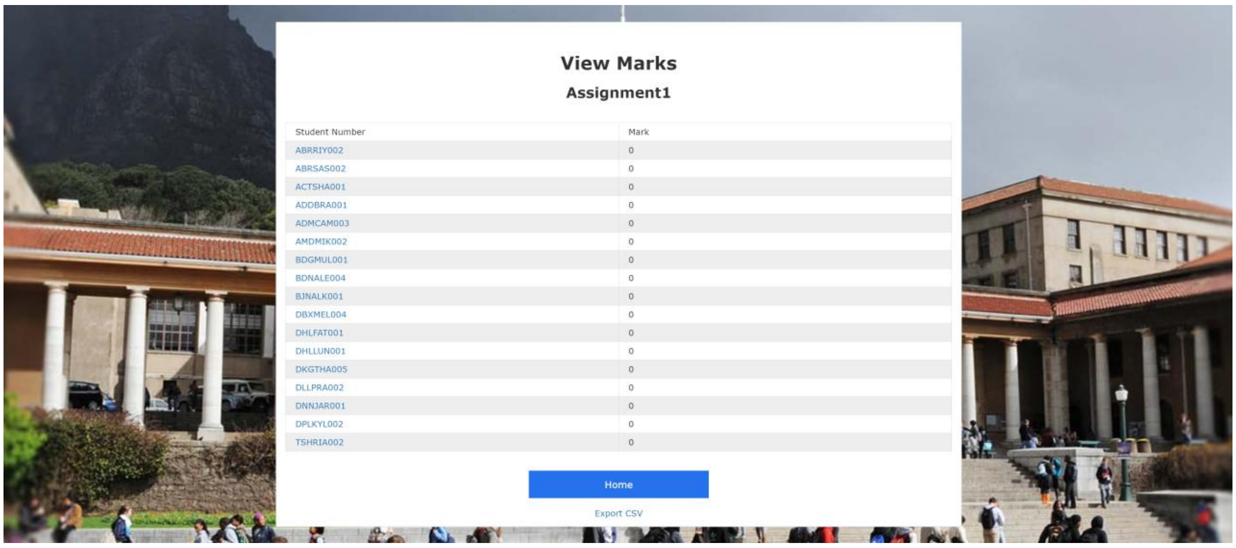


Assessment Name	
Assignment1	View Marks
Assignment2	View Marks
Assignment3	View Marks

Figure 9: The ”Marks” Tab

2.2.4.1. Assessment Mark Summary

To view the mark summary for an assessment, click on the “View Marks” button next to the assessment. You should be redirected to the “View Marks Summary” page (Figure 10) where all students are listed along with their marks. This breakdown is downloadable as a CSV file when the “Export CSV” link is clicked.


View Marks

Assignment1

Student Number	Mark
ABRRIY002	0
ABRSAS002	0
ACTSHA001	0
ADDBRA001	0
ADMCAM003	0
AMDMIK002	0
BDGMUL001	0
BDNALE004	0
BJNALK001	0
DBXMEL004	0
DHLFAT001	0
DHLLUN001	0
DKGTHA005	0
DLLPRA002	0
DNNJAR001	0
DPLKYL002	0
TSHRIA002	0

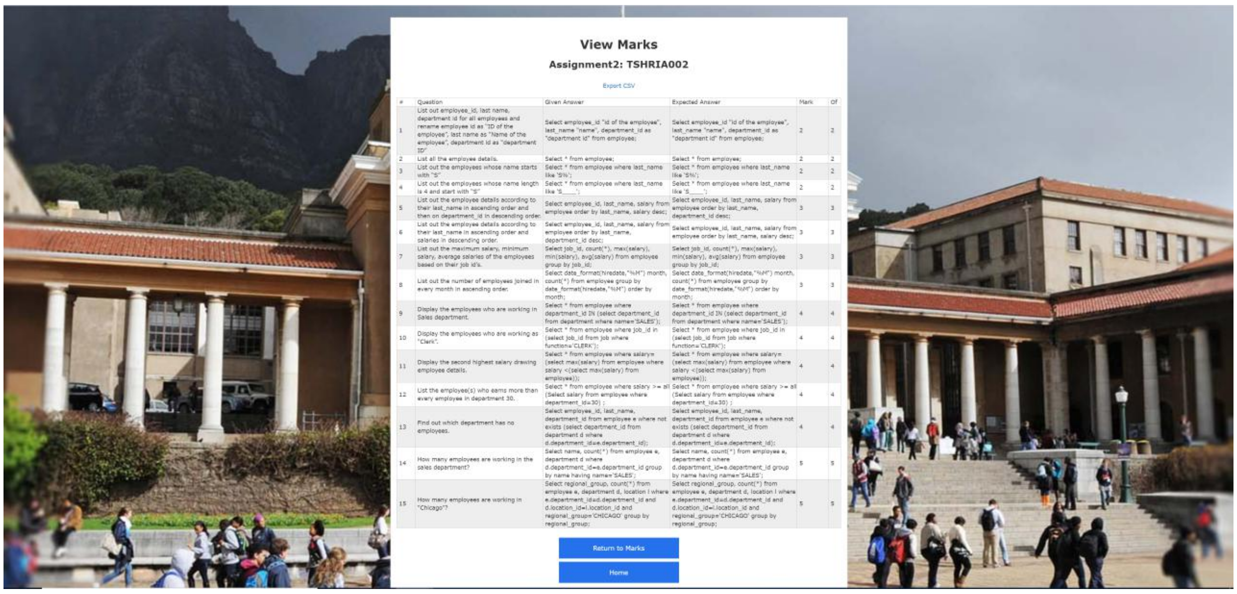
Figure 10: The ”Assessment Summary” Page

If the staff member would like to moderate an assessment, or check why a mark it is as it appears, clicking on the student number of the student will redirect them to the individual mark breakdown of the student (See Figure 11).

2.2.4.2. Individual Assessment Breakdown

The viewing of individual assessment mark breakdowns provides a view for the staff member to moderate or ensure that students have not cheated on their tests, or that they received the mark they deserve. Again, this table is downloadable as a CSV file.

Please note: if a user has retaken the assessment and their mark on the assessment is lower than the one on record, it means that a previous attempt by the student garnered them higher marks than their most recent attempt.

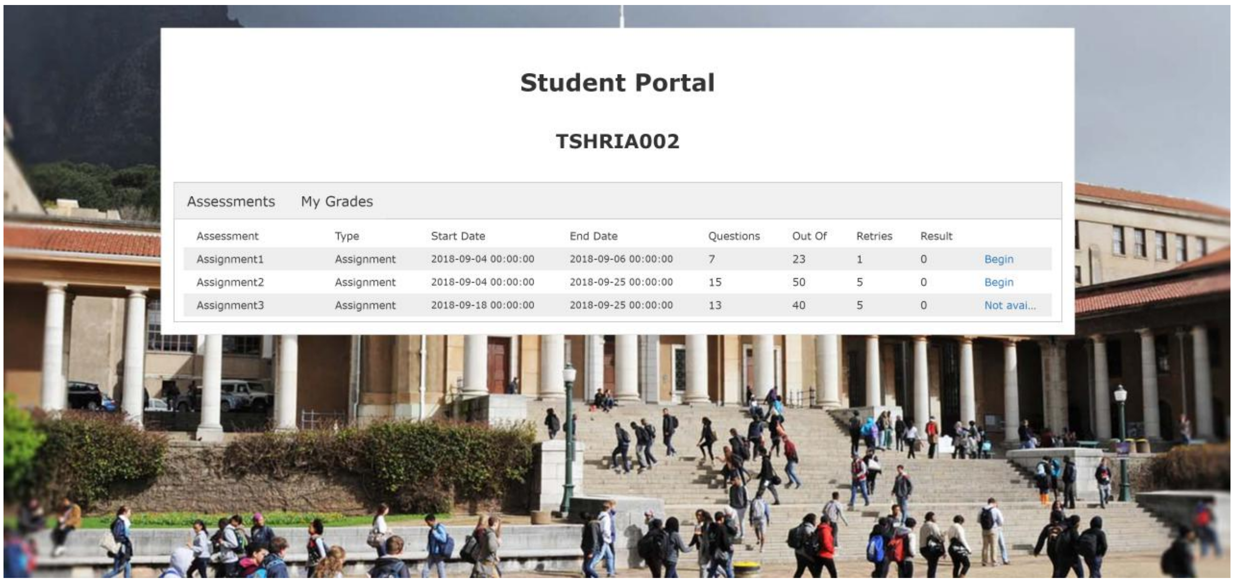


#	Question	Given Answer	Expected Answer	Mark	Of
1	List out employee_id, last name, department id for all employees and rename employee id as "ID of the employee", last name as "Name of the employee", department id as "department ID"	Select employee_id "id of the employee", last_name "name", department_id as "department id" from employee;	Select employee_id "id of the employee", last_name "name", department_id as "department id" from employee;	2	2
2	List all the employee details.	Select * from employee;	Select * from employee;	2	2
3	List out the employees whose name starts with "S"	Select * from employee where last_name like 'S%';	Select * from employee where last_name like 'S%';	2	2
4	List out the employees whose name length is 4 and start with "S"	Select * from employee where last_name like 'S____';	Select * from employee where last_name like 'S____';	2	2
5	List out the employee details according to their last_name in ascending order and then on department_id in descending order.	Select employee_id, last_name, salary from employee order by last_name, salary desc;	Select employee_id, last_name, salary from employee order by last_name, department_id desc;	3	3
6	List out the employee details according to their last_name in ascending order and salaries in descending order.	Select employee_id, last_name, salary from employee order by last_name, department_id desc;	Select employee_id, last_name, salary from employee order by last_name, salary desc;	3	3
7	List out the maximum salary, minimum salary, average salaries of the employees based on their job id's.	Select job_id, count(*), max(salary), min(salary), avg(salary) from employee group by job_id;	Select job_id, count(*), max(salary), min(salary), avg(salary) from employee group by job_id;	3	3
8	List out the number of employees joined in every month in ascending order.	Select date_format(hiredate,"%M") month, count(*) from employee group by date_format(hiredate,"%M") order by month;	Select date_format(hiredate,"%M") month, count(*) from employee group by date_format(hiredate,"%M") order by month;	3	3
9	Display the employees who are working in Sales department.	Select * from employee where department_id IN (select department_id from department where name='SALES');	Select * from employee where department_id IN (select department_id from department where name='SALES');	4	4
10	Display the employees who are working as "Clerk".	Select * from employee where job_id in (select job_id from job where function='CLERK');	Select * from employee where job_id in (select job_id from job where function='CLERK');	4	4
11	Display the second highest salary drawing employee details.	Select * from employee where salary= (select max(salary) from employee where salary <(select max(salary) from employee));	Select * from employee where salary= (select max(salary) from employee where salary <(select max(salary) from employee));	4	4
12	List the employee(s) who earns more than every employee in department 30.	Select * from employee where salary >= all (Select salary from employee where department_id=30) ;	Select * from employee where salary >= all (Select salary from employee where department_id=30) ;	4	4
13	Find out which department has no employees.	Select employee_id, last_name, department_id from employee e where not exists (select department_id from department d where d.department_id=e.department_id);	Select employee_id, last_name, department_id from employee e where not exists (select department_id from department d where d.department_id=e.department_id);	4	4
14	How many employees are working in the sales department?	Select name, count(*) from employee e, department d where d.department_id=e.department_id group by name having name='SALES';	Select name, count(*) from employee e, department d where d.department_id=e.department_id group by name having name='SALES';	5	5
15	How many employees are working in "Chicago"?	Select regional_group, count(*) from employee e, department d, location l where e.department_id=d.department_id and d.location_id=l.location_id and regional_group='CHICAGO' group by regional_group;	Select regional_group, count(*) from employee e, department d, location l where e.department_id=d.department_id and d.location_id=l.location_id and regional_group='CHICAGO' group by regional_group;	5	5

Figure 11: The ”Idividual Assessment Mark” Page

2.3. Student Portal

The Student Portal is set out in a more minimal tabular form compared to that of the Staff Porta with its landing page being the “Assessments” tab (Figure 12)Other tabs relate to other functionalities provided by the system. Because the student’s only role on the system is to attempt assessments, their view is restricted to taking assessments and viewing their marks.



Assessment	Type	Start Date	End Date	Questions	Out Of	Retries	Result	
Assignment1	Assignment	2018-09-04 00:00:00	2018-09-06 00:00:00	7	23	1	0	Begin
Assignment2	Assignment	2018-09-04 00:00:00	2018-09-25 00:00:00	15	50	5	0	Begin
Assignment3	Assignment	2018-09-18 00:00:00	2018-09-25 00:00:00	13	40	5	0	Not avai...

Figure 12: The ”Assessments” Tab in the Student Portal

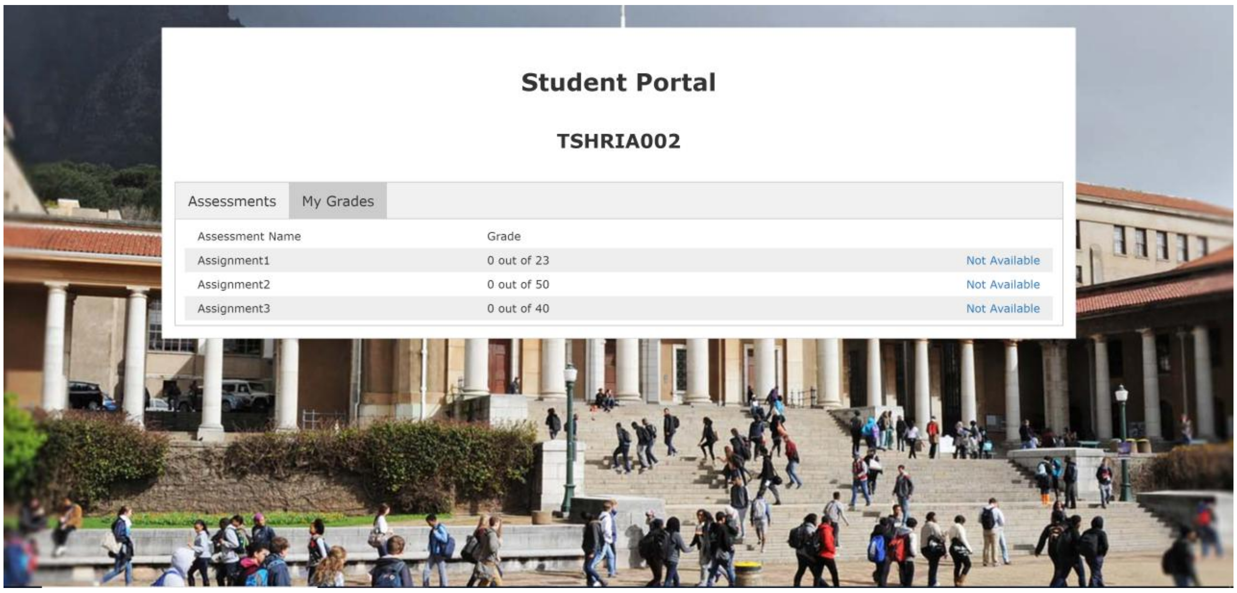
2.3.1. Assessments Tab

The assessments tab (Figure 12) provides the student with an overview of all assessments; the name, type, start and end date, number of questions, the total available marks, the number of retries, and the result achieved of each assessment. There are three assessment states: Begin, Closed, and Not available. If the assessment is open (falls within the start and end dates and times and has more than 0 attempts left) the student will be able to attempt the assessment and the link will be displayed as “Begin”. If the assessment has not opened yet (start date has not passed), the link will be disabled and “not available” will be displayed. If the student has used up all of his/her reattempts, the assignment status will be set to “closed” and the link will be disabled. This will also happen if the end date has elapsed.

If the student attempts the assessment more than once, the highest mark is stored in the “result” column, even if the result is not the most recent attempt.

2.3.2. Grade Tab

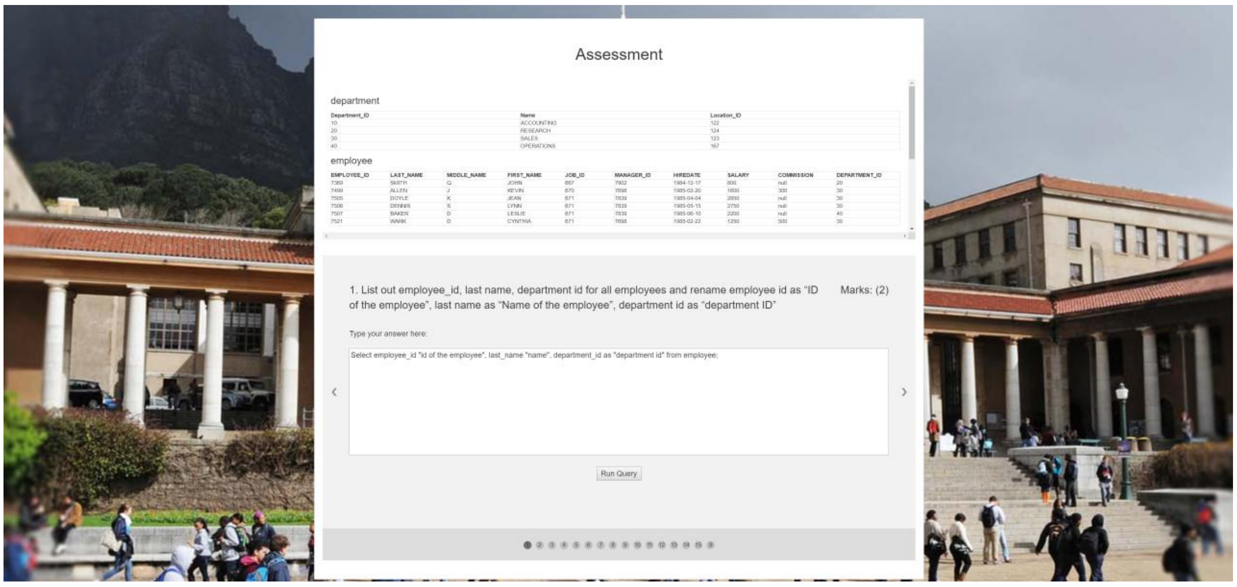
The ”Grade” tab (Figure 13) displays all results of the assessments taken by the students. They (the students) are able to view their results and feedback by clicking the link of the assessment on the right. This is the same feedback page that is displayed upon the submission of an assessment. If the assessment has not been attempted, “Not Available” message is show.



Assessment Name	Grade	
Assignment1	0 out of 23	Not Available
Assignment2	0 out of 50	Not Available
Assignment3	0 out of 40	Not Available

Figure 13: The ”Grades” Tab in the Student Portal

2.3.3. Taking an Assessment



department

Department_ID	Name	Location_ID
10	ACCOUNTING	122
20	RESEARCH	124
30	SALES	123
40	OPERATIONS	167

employee

EMPLOYEE_ID	LAST_NAME	MIDDLE_NAME	FIRST_NAME	JOB_ID	MANAGER_ID	HIREDATE	SALARY	COMMISSION	DEPARTMENT_ID
7369	SMITH	Q	JOHN	667	7902	1984-12-17	800	null	20
7499	ALLEN	J	KEVIN	670	7698	1985-02-20	1600	300	30
7505	DOYLE	K	JEAN	671	7839	1985-04-04	2850	null	30
7506	DENNIS	S	LYNN	671	7839	1985-05-15	2750	null	30
7507	BAKER	D	LESLIE	671	7839	1985-06-10	2200	null	40
7521	WARK	D	CYNTHIA	671	7698	1985-02-22	1250	500	30

1. List out employee_id, last name, department id for all employees and rename employee id as “ID of the employee”, last name as “Name of the employee”, department id as “department ID” Marks: (2)

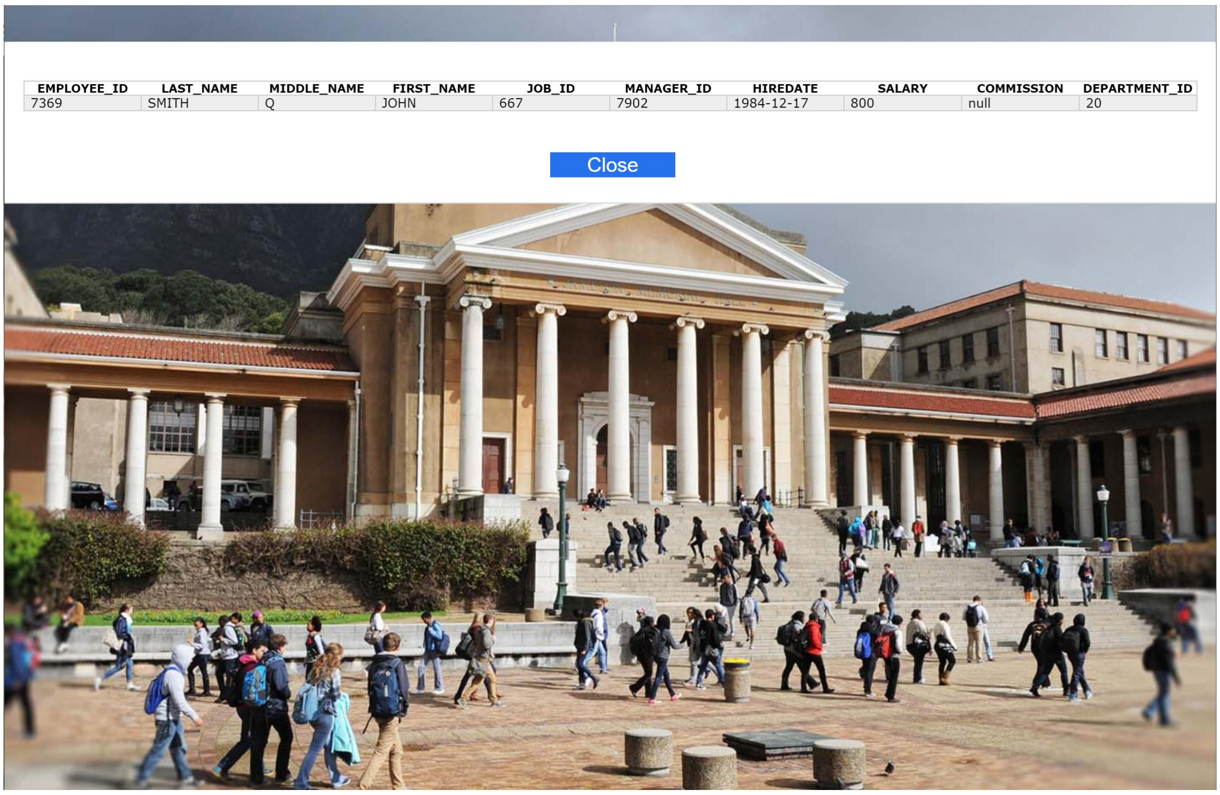
Type your answer here:

```
Select employee_id "id of the employee", last_name "name", department_id as "department id" from employee;
```

Figure 14: The Assessment Page

- A tabular representation of the necessary data tables needed to answer the questions appears at the top part of the page in a scrollable area.

- Below the data section there exists a slide section consisting of all of the questions assigned to the user. Under every question is a text area where the answer is entered.
- A query button exists under every text area to run test queries (Figure 15).

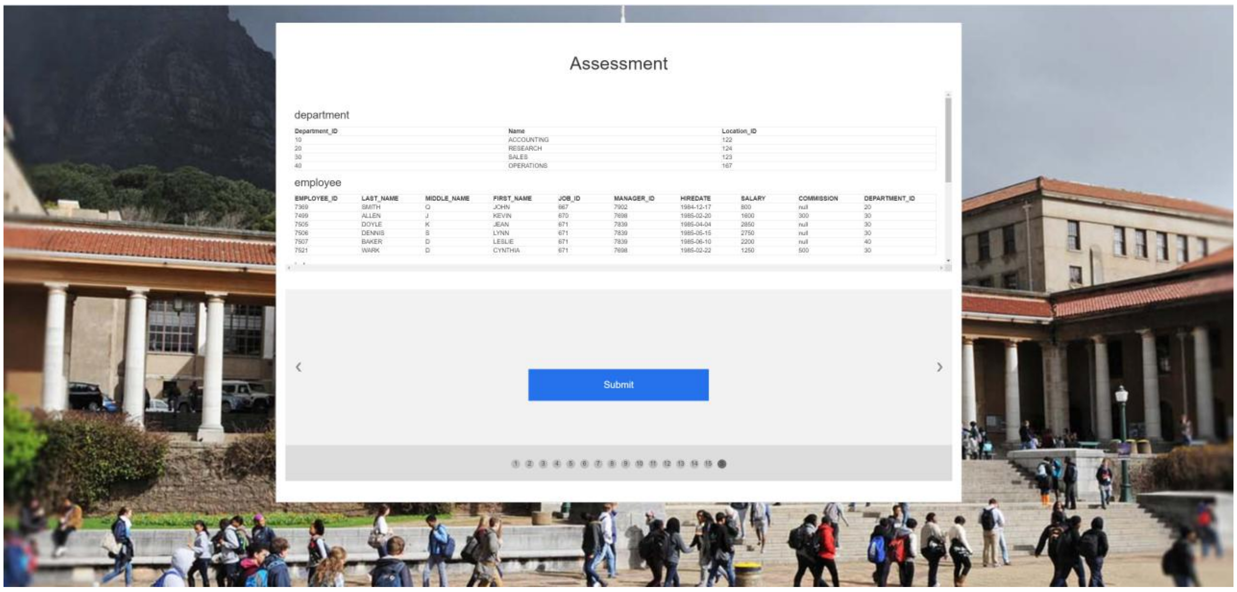


EMPLOYEE_ID	LAST_NAME	MIDDLE_NAME	FIRST_NAME	JOB_ID	MANAGER_ID	HIREDATE	SALARY	COMMISSION	DEPARTMENT_ID
7369	SMITH	Q	JOHN	667	7902	1984-12-17	800	null	20

Figure 15: Query Page for an SQL Statement

- Next and previous arrows control the navigation between questions
- Alternatively, there are numbered circles at the bottom of the page corresponding to each question in the assessment. These can be used to navigate through questions as well.
- The final slide, and the last circle (labelled S) is used to submit the assessment (Figure 16).

Upon submission, the user will be redirected to the loading page (Figure 17) which is where the marking of the assessment will take place. Marking can take time and if s/he wishes to view feedback and his/her result, s/he must wait until the page is redirected to the results page.



department

Department_ID	Name	Location_ID
10	ACCOUNTING	122
20	RESEARCH	124
30	SALES	123
40	OPERATIONS	167

employee

EMPLOYEE_ID	LAST_NAME	MIDDLE_NAME	FIRST_NAME	JOB_ID	MANAGER_ID	HIREDATE	SALARY	COMMISSION	DEPARTMENT_ID
7369	SMITH	Q	JOHN	667	7902	1984-12-17	800	null	20
7499	ALLEN	J	KEVIN	670	7698	1985-02-20	1600	300	30
7505	DOYLE	K	JEAN	671	7839	1985-04-04	2850	null	30
7506	DENNIS	S	LYNN	671	7839	1985-05-15	2750	null	30
7507	BAKER	D	LESLIE	671	7839	1985-06-10	2200	null	40
7521	WARK	D	CYNTHIA	671	7698	1985-02-22	1250	500	30

Figure 16: The Submit Assessment Page

Figure 17: The Marking Assessment Page

2.3.4. Results Page

This page displays marking feedback to the student in order for him/her to learn from his/her mistake(s) or see where s/he went wrong. This is also accessible through the "Grades" tab as mentioned above.

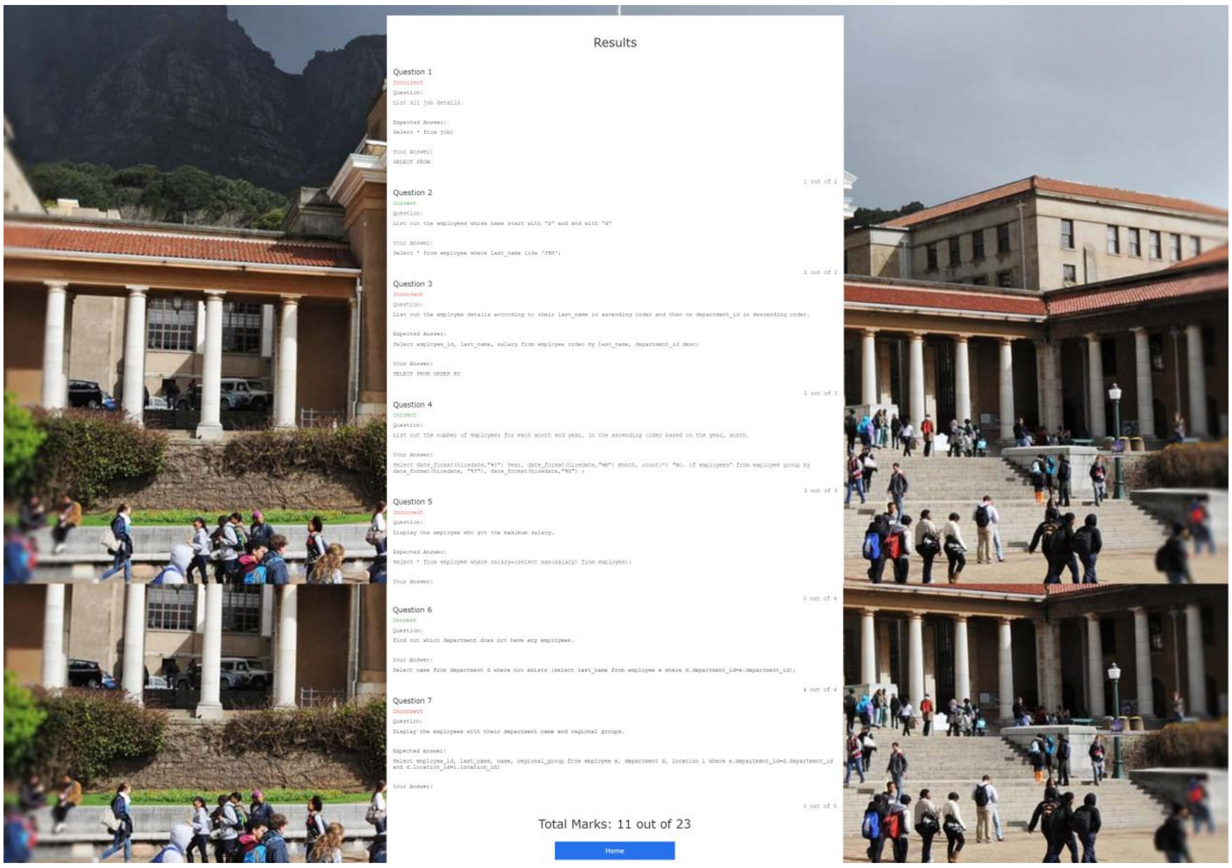
2.3.4.1. How are Marks Assigned?

The system executes the SQL statement given by the student. If there is an error (compile, SQL, etc), it will check the statement for key words, depending on the category of the question, to determine if it should get a portion of the marks. Similarly, if the number of rows or columns produced by the student's statement does not match that of the expected output, it will again determine if the statement deserves a portion of the mark, if not, the student will get 0. If all data items match, the user will get full marks.

The assignment of portioned marks occur if and only if the statement contains the following key words in their respective categories:

Table 2: Keywords Considered when Assigning Partial Marks

	Keywords
Category A:	SELECT, FROM
Category B:	SELECT, FROM, WHERE
Category C:	SELECT, FROM, ORDER BY
Category D:	SELECT, FROM, GROUP BY or HAVING
Category E:	SELECT, FROM, (,)
Category F:	SELECT, FROM, ALL or ANY or SOME or EXISTS
Category G:	SELECT, FROM, WHERE or JOIN


Results

Question 1
Incorrect
Question:
List all job details.

Expected Answer:
Select * from job;

Your Answer:
SELECT FROM

1 out of 2

Question 2
Correct
Question:
List out the employees whose name start with "S" and end with "H"

Your Answer:
Select * from employee where last_name like 'S%H';

2 out of 2

Question 3
Incorrect
Question:
List out the employee details according to their last_name in ascending order and then on department_id in descending order.

Expected Answer:
Select employee_id, last_name, salary from employee order by last_name, department_id desc;

Your Answer:
SELECT FROM ORDER BY

1 out of 3

Question 4
Correct
Question:
List out the number of employees for each month and year, in the ascending order based on the year, month.

Your Answer:
Select date_format(hiredate,"%Y") Year, date_format(hiredate,"%M") Month, count(*) "No. of employees" from employee group by date_format(hiredate, "%Y"), date_format(hiredate,"%M") ;

3 out of 3

Question 5
Incorrect
Question:
Display the employee who got the maximum salary.

Expected Answer:
Select * from employee where salary=(select max(salary) from employee);

Your Answer:

0 out of 4

Question 6
Correct
Question:
Find out which department does not have any employees.

Your Answer:
Select name from department d where not exists (select last_name from employee e where d.department_id=e.department_id);

4 out of 4

Question 7
Incorrect
Question:
Display the employees with their department name and regional groups.

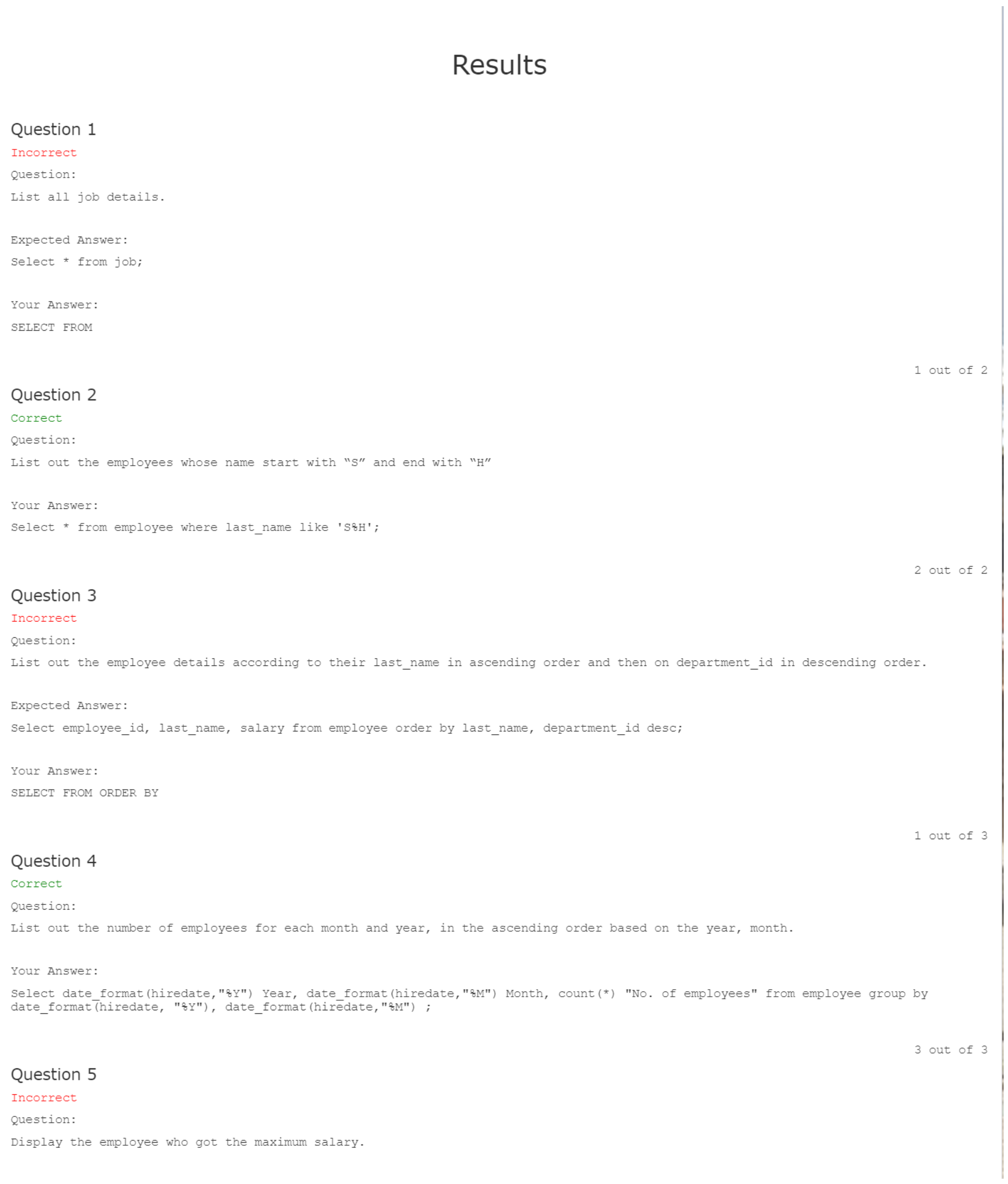
Expected Answer:
Select employee_id, last_name, name, regional_group from employee e, department d, location l where e.department_id=d.department_id and d.location_id=l.location_id;

Your Answer:

0 out of 5

Total Marks: 11 out of 23

Figure 18: Example of the Results Page



Results

Question 1

Incorrect

Question:

List all job details.

Expected Answer:

```
Select * from job;
```

Your Answer:

```
SELECT FROM
```

1 out of 2

Question 2

Correct

Question:

List out the employees whose name start with “S” and end with “H”

Your Answer:

```
Select * from employee where last_name like 'S%H';
```

2 out of 2

Question 3

Incorrect

Question:

List out the employee details according to their last_name in ascending order and then on department_id in descending order.

Expected Answer:

```
Select employee_id, last_name, salary from employee order by last_name, department_id desc;
```

Your Answer:

```
SELECT FROM ORDER BY
```

1 out of 3

Question 4

Correct

Question:

List out the number of employees for each month and year, in the ascending order based on the year, month.

Your Answer:

```
Select date_format(hiredate,"%Y") Year, date_format(hiredate,"%M") Month, count(*) "No. of employees" from employee group by
date_format(hiredate, "%Y"), date_format(hiredate,"%M") ;
```

3 out of 3

Question 5

Incorrect

Question:

Display the employee who got the maximum salary.

Figure 19: Detail of the Results Page

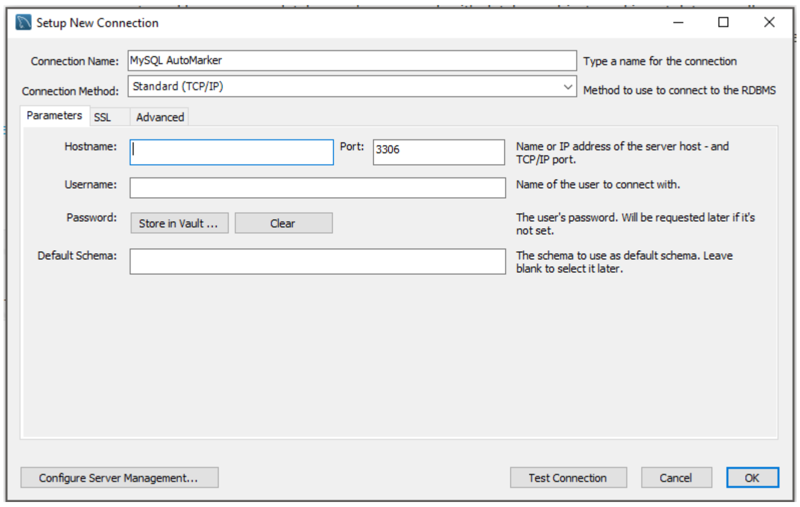
3. Using and Navigating the MySQL Component of the Automatic Marker

In addition to the front-facing user interface, the lecturer or tutor has access to an additional interface – MySQL. It is through this that the staff member will be able to import new question datasets, alter marks, alter assessment details, and export data if necessary. The following section covers basic functionalities of the MySQL database server for the SQL Automatic Marker.

3.1. Connecting to the MySQL Automatic Marker Database

In order to access the MySQL cloud database, follow the steps set out below:

1. In the MySQL Workbench, select "Setup New Connection." You should be shown a screen as displayed in Figure 20.



Setup New Connection

Connection Name:	MySQL AutoMarker	Type a name for the connection
Connection Method:	Standard (TCP/IP)	Method to use to connect to the RDBMS

Parameters | SSL | Advanced

Hostname:	Port: 3306	Name or IP address of the server host - and TCP/IP port.
Username:		Name of the user to connect with.
Password:	Store in Vault ... / Clear	The user's password. Will be requested later if it's not set.
Default Schema:		The schema to use as default schema. Leave blank to select it later.

Figure 20: Set Up Window

2. Enter the following details into the given fields:

   (a) Connection Name: Name of the database (recommended: MySQL AutoMarker)

   (b) Connection Method: Standard (TCP/IP)

   (c) Hostname: sqlautomarker.cdtrwjzldbjr.us-east-1.rds.amazonaws.com

   (d) Port: 3306

   (e) Username: mltzac001

   (f) Password: SQLauto2018

   (g) Default Schema: leave empty

   The display should look like it does in Figure 21.

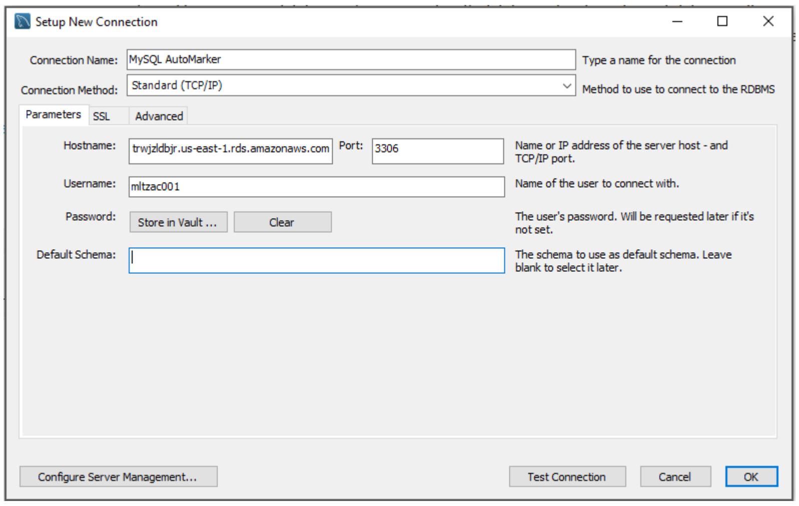
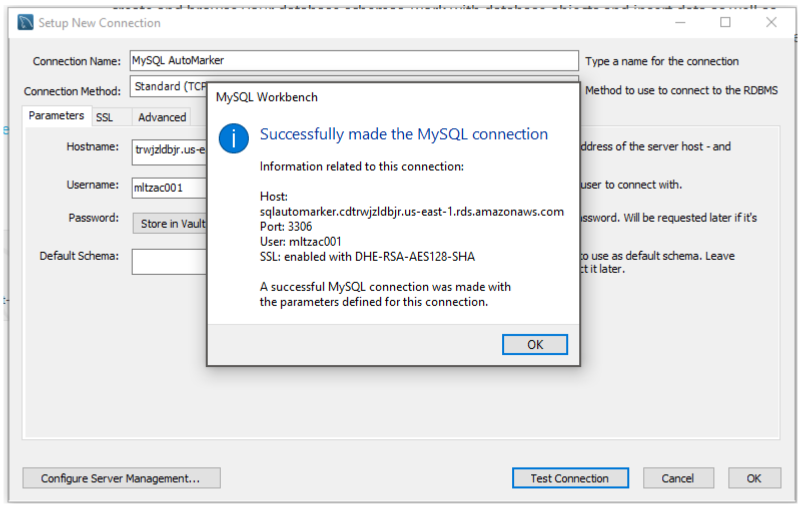

Setup New Connection
Connection Name: MySQL AutoMarker
Type a name for the connection
Connection Method: Standard (TCP/IP)
Method to use to connect to the RDBMS
Parameters
SSL
Advanced
Hostname: trwjzldbjr.us-east-1.rds.amazonaws.com
Port: 3306
Name or IP address of the server host - and TCP/IP port.
Username: mltzac001
Name of the user to connect with.
Password:
Store in Vault ...
Clear
The user's password. Will be requested later if it's not set.
Default Schema:
The schema to use as default schema. Leave blank to select it later.
Configure Server Management...
Test Connection
Cancel
OK

Figure 21: Set Up Window

3. Selectt “Test Connection.” The result should be as seen in Figure 22.


MySQL Workbench

Successfully made the MySQL connection

Information related to this connection:

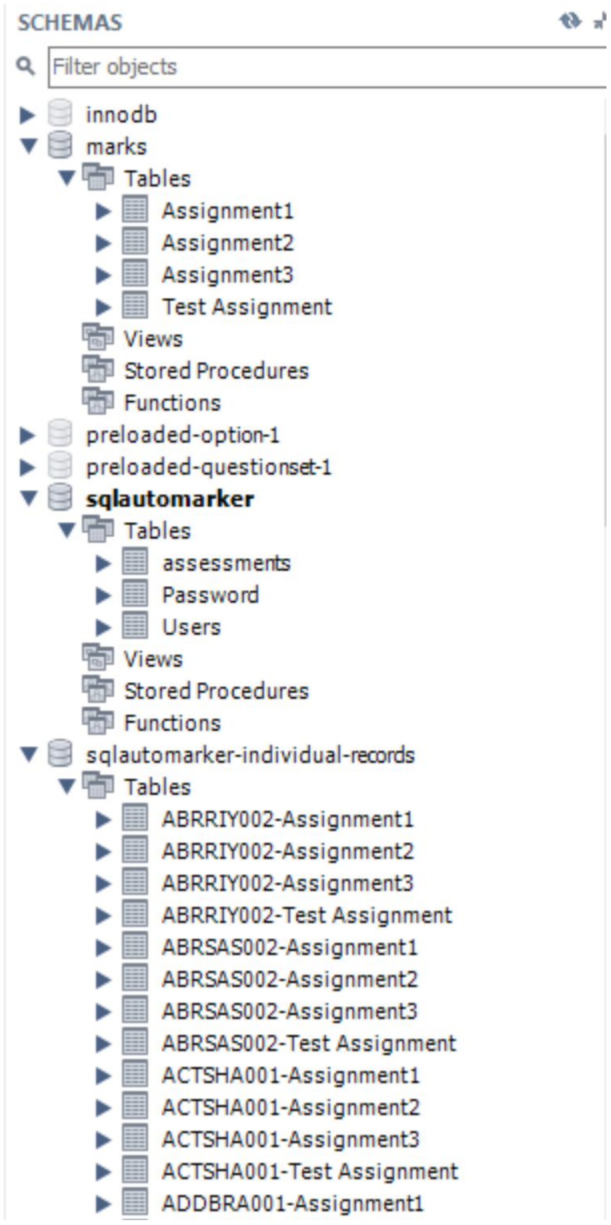
Host:
sqlautomarker.cdtrwjzldbjr.us-east-1.rds.amazonaws.com
Port: 3306
User: mltzac001
SSL: enabled with DHE-RSA-AES128-SHA

A successful MySQL connection was made with
the parameters defined for this connection.

Figure 22: Set Up Window

4. If the test is successful, select “OK” on the “Setup New Connection” window.

3.2. Using the MySQL Automatic Marker Database

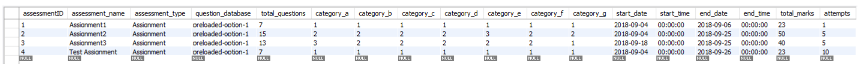


```
SCHEMAS
Filter objects
▶ innodb
▼ marks
    ▼ Tables
        ▶ Assignment1
        ▶ Assignment2
        ▶ Assignment3
        ▶ Test Assignment
      Views
      Stored Procedures
      Functions
▶ preloaded-option-1
▶ preloaded-questionset-1
▼ sqlautomarker
    ▼ Tables
        ▶ assessments
        ▶ Password
        ▶ Users
      Views
      Stored Procedures
      Functions
▼ sqlautomarker-individual-records
    ▼ Tables
        ▶ ABRRIY002-Assignment1
        ▶ ABRRIY002-Assignment2
        ▶ ABRRIY002-Assignment3
        ▶ ABRRIY002-Test Assignment
        ▶ ABRSAS002-Assignment1
        ▶ ABRSAS002-Assignment2
        ▶ ABRSAS002-Assignment3
        ▶ ABRSAS002-Test Assignment
        ▶ ACTSHA001-Assignment1
        ▶ ACTSHA001-Assignment2
        ▶ ACTSHA001-Assignment3
        ▶ ACTSHA001-Test Assignment
        ▶ ADDBRA001-Assignment1
```

Figure 23: Overview of the General MySQL Database Layout

3.2.1. Accessing Assessments

Information about assessments are stored in the ‘sqlautomarker‘.‘assessments‘ table. This information is viewable upon signing into the Staff Portal and so there is no direct need to use this table unless the staff member wishes to remove an assessment.



assessmentID	assessment_name	assessment_type	question_database	total_questions	category_a	category_b	category_c	category_d	category_e	category_f	category_g	start_date	start_time	end_date	end_time	total_marks	attempts
1	Assignment1	Assignment	preloaded-option-1	7	1	1	1	1	1	1	1	2018-09-04	00:00:00	2018-09-06	00:00:00	23	1
2	Assignment2	Assignment	preloaded-option-1	15	2	2	2	2	3	2	2	2018-09-04	00:00:00	2018-09-25	00:00:00	50	5
3	Assignment3	Assignment	preloaded-option-1	13	3	2	2	2	2	2	1	2018-09-18	00:00:00	2018-09-25	00:00:00	40	5
4	Test Assignment	Assignment	preloaded-option-1	7	1	1	1	1	1	1	1	2018-09-04	00:00:00	2018-09-26	00:00:00	23	10
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

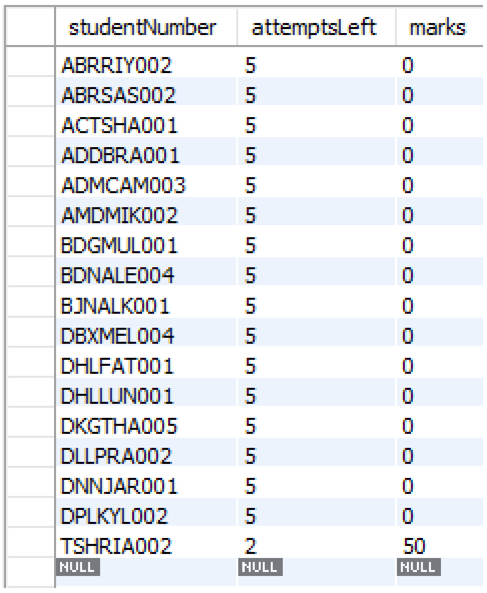
Figure 24: Example of the Assessments Table

3.2.2. Accessing Assessment Marks

Assessment marks can be found in the ‘marks‘ database (Figure 25). Each table in the database reflects an assessment. This information is viewable through the staff portal but in the event that a staff member wishes to update a mark, s/he will need to use these tables to do so. Additionally, what is not shown on the Staff Portal when viewing assessment summaries are the attempts left by each student to retry the assessment. This is included in the event that the assessor wants to gauge the difficulty level of the assessment. This table only stores the highest mark for the user so if s/he retook the assessment but got a lower mark, while her individual record will reflect the lower mark, this record will reflect the higher of the two. By default, the marks are set to 0.

3.2.3. Updating Assessment Marks

In the event of moderation, an assessor can change the mark of an individuals assessment by selecting on the 'marks' cell in MySQL, changing the value, and the clicking "Apply" to commit the changes.



studentNumber	attemptsLeft	marks
ABRRIY002	5	0
ABRSAS002	5	0
ACTSHA001	5	0
ADDBRA001	5	0
ADMCAM003	5	0
AMDMIK002	5	0
BDGMUL001	5	0
BDNALE004	5	0
BJNALK001	5	0
DBXMEL004	5	0
DHLFAT001	5	0
DHLLUN001	5	0
DKGTHA005	5	0
DLLPRA002	5	0
DNNJAR001	5	0
DPLKYL002	5	0
TSHRIA002	2	50
NULL	NULL	NULL

Figure 25: Example of the Marks Table

3.2.4. Accessing Individual Marks

This can be done by going to the 'sqlautomarker-individual-records' database. This database stores tables of all assessments assigned to students. In these tables, the staff member can view all assessment details for the particular student, such as the questions assigned to him/her, the student's answers as well as the marks that they achieved. This view is provided in the Staff Portal and can be exported from there as a CSV file.

3.3. Question Databases

Question databases is what is used when assigning assessments to students and is also used in the marking of assessments. This was not included in the Staff Portal as it was decided that it is much simpler for the staff member to import his/her own CSV datafiles into MySQL directly because MySQL already accommodates this functionality.

3.3.1. Properties of a Question Database

The properties of a question database are as follows:

1. Naming
   A question database has to contain one of the following phrases: 'questionset' or 'preloaded'. If the database name does not contain these words, it will not be picked up by the system when the staff member wishes to create an assessment.
2. Tables
   All schema used in the assessment are to be stored individually in the database according to the name used in the SQL statements (e.g. department, employee, location, etc.). All questions are to be stored in a table named 'questions'.
3. The 'questions' table has to take on the format as seen in Figure 27

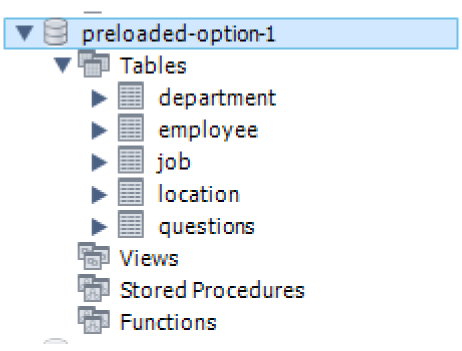
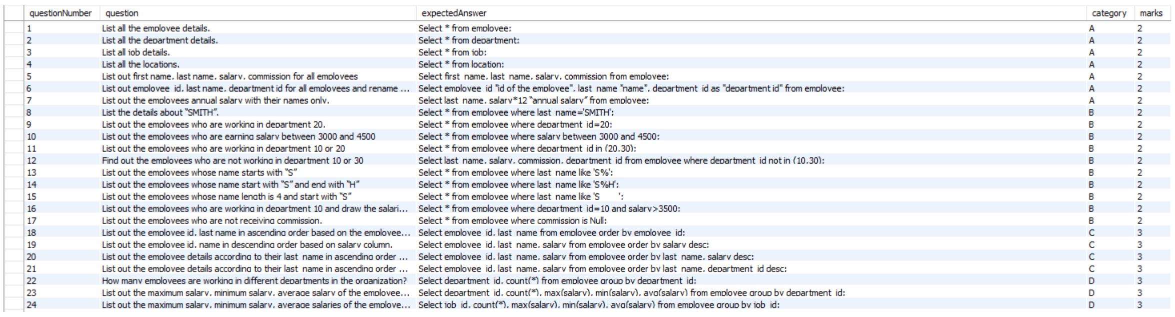


Figure 26: Example of a Preloaded Question Database

- questionNumber – the number of the question. It is recommended that these are in order and incremented by 1.
- question – the actual question to be asked to the user.
- expectedAnswer – the *working* SQL statement that will produce the correct answer set for the question. If this SQL statement does not work, the student will be marked incorrectly.
- category – the category of the question (A-G, as mentioned above).
- marks – the marks that the question is worth. Please stick to the recommended mark allocation for each category as stipulated in 2.2.2.1.



questionNumber	question	expectedAnswer	category	marks
1	List all the employee details.	Select * from employee;	A	2
2	List all the department details.	Select * from department;	A	2
3	List all job details.	Select * from job;	A	2
4	List all the locations.	Select * from location;	A	2
5	List out first name, last name, salary, commission for all employees	Select first_name, last_name, salary, commission from employee;	A	2
6	List out employee_id, last name, department id for all employees and rename …	Select employee_id "id of the employee", last_name "name", department_id as "department id" from employee;	A	2
7	List out the employees annual salary with their names only.	Select last_name, salary*12 "annual salary" from employee;	A	2
8	List the details about "SMITH".	Select * from employee where last_name='SMITH';	B	2
9	List out the employees who are working in department 20.	Select * from employee where department_id=20;	B	2
10	List out the employees who are earning salary between 3000 and 4500	Select * from employee where salary between 3000 and 4500;	B	2
11	List out the employees who are working in department 10 or 20	Select * from employee where department_id in (20,30);	B	2
12	Find out the employees who are not working in department 10 or 30	Select last_name, salary, commission, department_id from employee where department_id not in (10,30);	B	2
13	List out the employees whose name starts with "S"	Select * from employee where last_name like 'S%';	B	2
14	List out the employees whose name start with "S" and end with "H"	Select * from employee where last_name like 'S%H';	B	2
15	List out the employees whose name length is 4 and start with "S"	Select * from employee where last_name like 'S ';	B	2
16	List out the employees who are working in department 10 and draw the salari…	Select * from employee where department_id=10 and salary>3500;	B	2
17	List out the employees who are not receiving commission.	Select * from employee where commission is Null;	B	2
18	List out the employee id, last name in ascending order based on the employee…	Select employee_id, last_name from employee order by employee_id;	C	3
19	List out the employee id, name in descending order based on salary column.	Select employee_id, last_name, salary from employee order by salary desc;	C	3
20	List out the employee details according to their last_name in ascending order …	Select employee_id, last_name, salary from employee order by last_name, salary desc;	C	3
21	List out the employee details according to their last_name in ascending order …	Select employee_id, last_name, salary from employee order by last_name, department_id desc;	C	3
22	How many employees are working in different departments in the organization?	Select department_id, count(*) from employee group by department_id;	D	3
23	List out the maximum salary, minimum salary, average salary of the employee…	Select department_id, count(*), max(salary), min(salary), avg(salary) from employee group by department_id;	D	3
24	List out the maximum salary, minimum salary, average salaries of the employe…	Select job_id, count(*), max(salary), min(salary), avg(salary) from employee group by job_id;	D	3

Figure 27: Example of a Preloaded Question Set

3.3.2. Format of a Question CSV file

The format of a CSV file containing the questions should take the following format where the first line defines the column names and every subsequent line follows the format:

”questionNumber”;”question”;”expectedAnswer”;”category”;”marks”

E.g.
1;”List all the employee details.”;”Select * from employee;”;”A”;2
2;”List all the department details.”;”Select * from department;”;”A”;2
3;”List all job details.”;”Select * from job;”;”A”;2
4;”List all the locations.”;”Select * from location;”;”A”;2
5;”List out first name, last name, salary, commission for all employees”;”Select first_name, last_name, salary, commission from employee;”;”A”;2
6;”List out employee_id, last name, department id for all employees and rename employee id as “ID of the employee”, last name as “Name of the employee”, department id as “department ID””;”Select employee_id ””id of the employee””, last_name ””name””, department_id as ””department id”” from employee;”;”A”;2
7;”List out the employees annual salary with their names only.”;”Select last_name, salary*12 “annual salary” from employee;”;”A”;2
8;”List the details about “SMITH”.”;”Select * from employee where last_name=’SMITH’;”;”B”;2
9;”List out the employees who are working in department 20.”;”Select * from employee where department_id=20;”;”B”;2
10;”List out the employees who are earning salary between 3000 and 4500”;”Select * from employee where salary between 3000 and 4500;”;”B”;2

11;"List out the employees who are working in department 10 or 20";"Select * from employee where department_id in (20,30);";"B";2
12;"Find out the employees who are not working in department 10 or 30";"Select last_name, salary, commission, department_id from employee where department_id not in (10,30);";"B";2

3.3.3. Example of a Format for a Question Data Table CSV File

An example of the format for a CSV file containing a data table that the questions rely upon is as follows, where the first line defines the column names and every subsequent line follows the format:

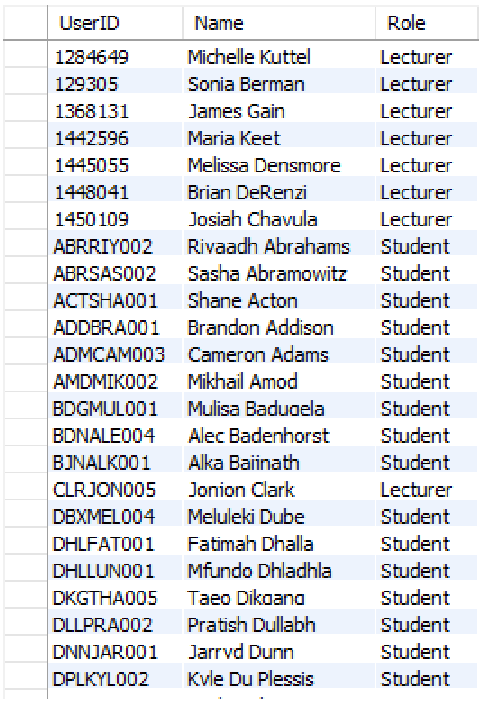
    "Department_ID";"Name";"Location_ID"

E.g.

    "Department_ID";"Name";"Location_ID"
    10;"ACCOUNTING";122
    20;"RESEARCH";124
    30;"SALES";123
    40;"OPERATIONS";167

3.4. Users and Passwords

Users of the SQL Automatic Marker are assigned to the 'sqlautomarker'.'Users' table (Figure 28). The addition of users can be done through the Staff Portal (See Section 2.2), as can the removal of all students. While the front-facing interface can remove students from the database, in order to remove staff members or individual students, it will have to be done through the MySQL Workbench as this action is considered rare and thus should only be accessed if needed.

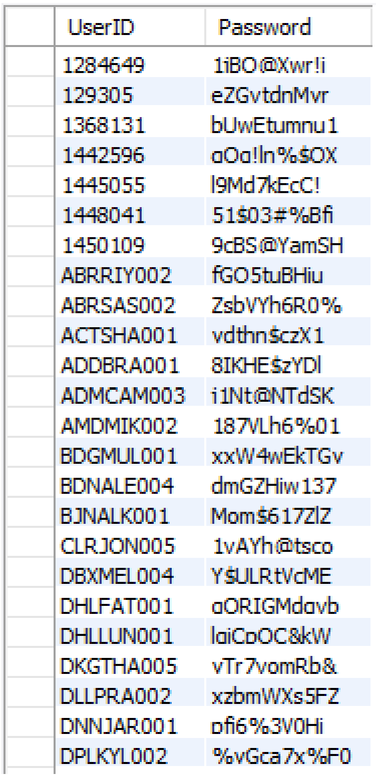


UserID	Name	Role
1284649	Michelle Kuttel	Lecturer
129305	Sonia Berman	Lecturer
1368131	James Gain	Lecturer
1442596	Maria Keet	Lecturer
1445055	Melissa Densmore	Lecturer
1448041	Brian DeRenzi	Lecturer
1450109	Josiah Chavula	Lecturer
ABRRIY002	Rivaadh Abrahams	Student
ABRSAS002	Sasha Abramowitz	Student
ACTSHA001	Shane Acton	Student
ADDBRA001	Brandon Addison	Student
ADMCAM003	Cameron Adams	Student
AMDMIK002	Mikhail Amod	Student
BDGMUL001	Mulisa Badugela	Student
BDNALE004	Alec Badenhorst	Student
BJNALK001	Alka Bajinath	Student
CLRJON005	Jonjon Clark	Lecturer
DBXMEL004	Meluleki Dube	Student
DHLFAT001	Fatimah Dhalla	Student
DHLLUN001	Mfundo Dhladhla	Student
DKGTHA005	Taeo Dikgang	Student
DLLPRA002	Pratish Dullabh	Student
DNNJAR001	Jarryd Dunn	Student
DPLKYL002	Kyle Du Plessis	Student

Figure 28: Example of the Users Table

3.4.1. Removing Individuals

To remove individuals, find their Name or User ID in the table, right click on it, and select "Delete Row(s)." After the row is removed, click the "Apply" button.
Note: if you are removing a user, you will have to complete the similar action in the 'sqlautomarker'.'Password' table (Figure 29).



UserID	Password
1284649	1iBO@Xwr!i
129305	eZGvtdnMvr
1368131	bUwEtumnu1
1442596	aOa!ln%$OX
1445055	l9Md7kEcC!
1448041	51$03#%Bfi
1450109	9cBS@YamSH
ABRRIY002	fGO5tuBHiu
ABRSAS002	ZsbVYh6R0%
ACTSHA001	vdthn$czX1
ADDBRA001	8IKHE$zYDl
ADMCAM003	i1Nt@NTdSK
AMDMIK002	187VLh6%01
BDGMUL001	xxW4wEkTGv
BDNALE004	dmGZHiw137
BJNALK001	Mom$617ZlZ
CLRJON005	1vAYh@tsco
DBXMEL004	Y$ULRtVcME
DHLFAT001	aORIGMdavb
DHLLUN001	laiCpOC&kW
DKGTHA005	vTr7vomRb&
DLLPRA002	xzbmWXs5FZ
DNNJAR001	pfi6%3V0Hi
DPLKYL002	%vGca7x%F0

Figure 29: Example of the Passwords Table

3.4.2. Changing Passwords

Because of the nature of this system, the ideal situation would be to integrate it with UCT's internal system so that a user can log in with his/her UCT credentials. However, because this is not currently possible, the system generates a random, secure password for each user upon the user's creation through the Staff Portal. The user is not able to change his/her password due to the fact that this system is designed for the completion of closed practical tests and homework assessments and nothing more, the password remains fixed until it can be integrated with Vula. The staff member can change passwords if s/he wishes by simply selecting on the "Password" of the User, updating the field, and clicking the "Apply" button in MySQL.