Code Runner Extension

Semester Report

by

Dian LIN

ID:6916490

UPI:dlin368

BTech 451
Information Technology
in
Bachelor of Technology

University of Auckland

Supervisor: Dr. Wannes van der Mark

ABSTRACT

Code Runner provides an on-line platform for students to complete their programming assignments. It supports different programming languages. It is widely used at the University of the Auckland for courses from Stage One to Stage Three. To enable each student to complete their assignments individually, this study aims to research different ways of Code Runner assignments cheating. The goal is to extend Code Runner to have parameterized exercises so that each student is presented with unique exercise. The outcome ideas from research results will be implemented into the platform in order to make cheating more difficult in Code Runner.

Contents

Chapter 1

Introduction

Moodle is an open-source learning management system(LMS) [1]. And is an open-source platform. At the moment, Moodle is extensively used in schools and tertiary institutions in Australia and New Zealand. It is also widely used in Europe, especially in Spain and the U.K. The core of Moodle is courses that contain activities and resources, there are about 20 different types of activities available including assignments, quizzes, choices etc. [2]

At the Department of Computer Science of the University of Auckland, it is used for programming exercises. Programming assignments are usually required environments installation such as Java. Moodle provides a way for students to easily complete their assignments without the installation of programming software and environment path setting but browsers only. Hence, Moodle as one of the most important tools in the Department of Computer Science to test students learning abilities. It can be used to set up coding assignments, multiple-choice questions and simple quizzes. Moodle is able to run the input source code at the background and immediately gives the feedback to participants about their submission. For teachers, this activity-based model combines the activities into sequences and groups. It helps teachers to guide participants. Also, Moodle contains auto-grading and group courses functionality, which saves the efforts of the manual. So it is widely accepted by people through the department.

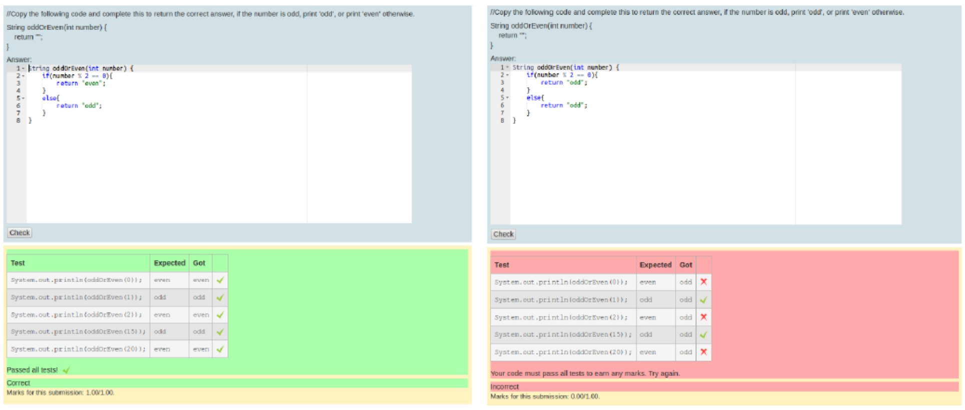
1.1 Code Runner

Code Runner [3] is a Moodle question type that requests students to submit program code to some given specification such as Java or Python function. At the University of Auckland, Code Runner is the main reason for using Moodle, as lecturers or administrators can simply post questions by defining question contents and test cases under specific course list as the assignment build-up. The Sandbox is used to run the series of test cases for input source code under limited time which prevents infinite loops or deadlocks from blocking the system. The outputs are compared with expected answers and used to automatically mark the assignment works upon given a marking guide. The Sandbox using keeps Code Runner safe since it could effectively prevent malicious code injection which could be used to guard the system.

Every participant should be able to pass all test cases that are pre-defined by question creators in order to gain the full marks, Code Runner becomes a convenient and useful tool. Because it helps students getting more and more familiar with testing topics which introduced in lectures. And encouraging students to gain more marks in terms of splitting a large assignment into several simple questions.

Moodle provides a large database for Code Runner to store data. Each question creation will create a new record inserted into Moodle database(MySql Database, will be mentioned in the following section), instead of searching created questions in Code Runner website, authorised people could simply look up and modify every question details by SQL. To connect Code Runner with Database Management System(DBMS), Code Runner becomes more feasible to handle its functionality.

The output of different feedback of Code Runner is presented in Figure 1.1.



```
//Copy the following code and complete this to return the correct answer, if the number is odd, print 'odd', or print 'even' otherwise.
String oddOrEven(int number) {
  return "";
}
Answer:
String oddOrEven(int number) {
    if(number % 2 == 0){
        return "even";
    }
    else{
        return "odd";
    }
}
```

Test	Expected	Got	
System.out.println(oddOrEven(0));	even	even	✓
System.out.println(oddOrEven(1));	odd	odd	✓
System.out.println(oddOrEven(2));	even	even	✓
System.out.println(oddOrEven(15));	odd	odd	✓
System.out.println(oddOrEven(20));	even	even	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

```
//Copy the following code and complete this to return the correct answer, if the number is odd, print 'odd', or print 'even' otherwise.
String oddOrEven(int number) {
  return "";
}
Answer:
String oddOrEven(int number) {
    if(number % 2 == 0){
        return "odd";
    }
    else{
        return "odd";
    }
}
```

Test	Expected	Got	
System.out.println(oddOrEven(0));	even	odd	✗
System.out.println(oddOrEven(1));	odd	odd	✓
System.out.println(oddOrEven(2));	even	odd	✗
System.out.println(oddOrEven(15));	odd	odd	✓
System.out.println(oddOrEven(20));	even	odd	✗

Your code must pass all tests to earn any marks. Try again.

Incorrect

Marks for this submission: 0.00/1.00.

Figure 1.1: Code Runner output feedback of correct and wrong answers

1.2 MySQL

MySql is an open-source relational database management system(RDBMS), and written in C and C++. [6] As the advantage of MySql, it is allowed users to create relationships between tables by primary keys and foreign keys. This feature helps database become more consistency, and users are able to look for particular data quickly. In Moodle, lots of data there need to be recorded and tracked, MySql database becomes a good tool to manage the platform.

1.3 Vulnerability to Cheating

Code Runner is an online tool that automatically tests and marks students' assignments by comparing answers to predefined outputs. This is a significant weakness of Code Runner because it is not able to detect cheating. Students could simply copy-paste the source code from someone else who hold the same question and already passed all test cases. Those students would be able to gain the marks without putting any efforts. Assignments cheating is not allowed by the University. Some damaging effects would be caused by Code Runner assignments cheating:

- Unfair to hard working students and lecturers:

  Most of students in the University put lots of efforts on their assignments, they usually spend time on studying and preparing assignments. It is quite unfair for those students who hard working are getting the same grade as those who cheating on assignments. Also, lots of efforts are put by lecturers to prepare teaching materials and create assignments. If cheating becomes possible, it would be unfair to lecturers as well.

- Lost faith in Code Runner:

  This is a significant effect impact the University. Once cheating becomes a possible way to solve the assignments, the quality of students is not able to be evaluable. Reputation of the University will be reduced due to students lost their faith in Code Runner.

- More students are possible to cheat on assignments in Code Runner:

  If cheating is allowed in Code Runner, more students would try it to save time for fun. As students could get the same grade even they cheat. Then the learning ability of students is poor,even though cheating students could pass the course, it does not mean that those students have obtained the knowledge that required by the course.

The basic principle of whatever assignments or Code Runner tests at the University of Auckland is that it encourages students to do individual works in order to obtain the skills and ideas that required by the courses. Therefore, if Code Runner is going to be treated as one of the main assignment parts, it is necessary to push up barriers against cheating in Code Runner.

1.4 Study Outline

In this project, we will first research on Code Runner program before idea implementation. In research, useful information will be obtained, and becomes the guideline to implement ideas properly, so that we will not waste our time on unnecessary coding. Based on research results, the following ideas implementation would be much simpler and easier because of the corrected guide direction. The workflow indicates what the steps of the project will be:

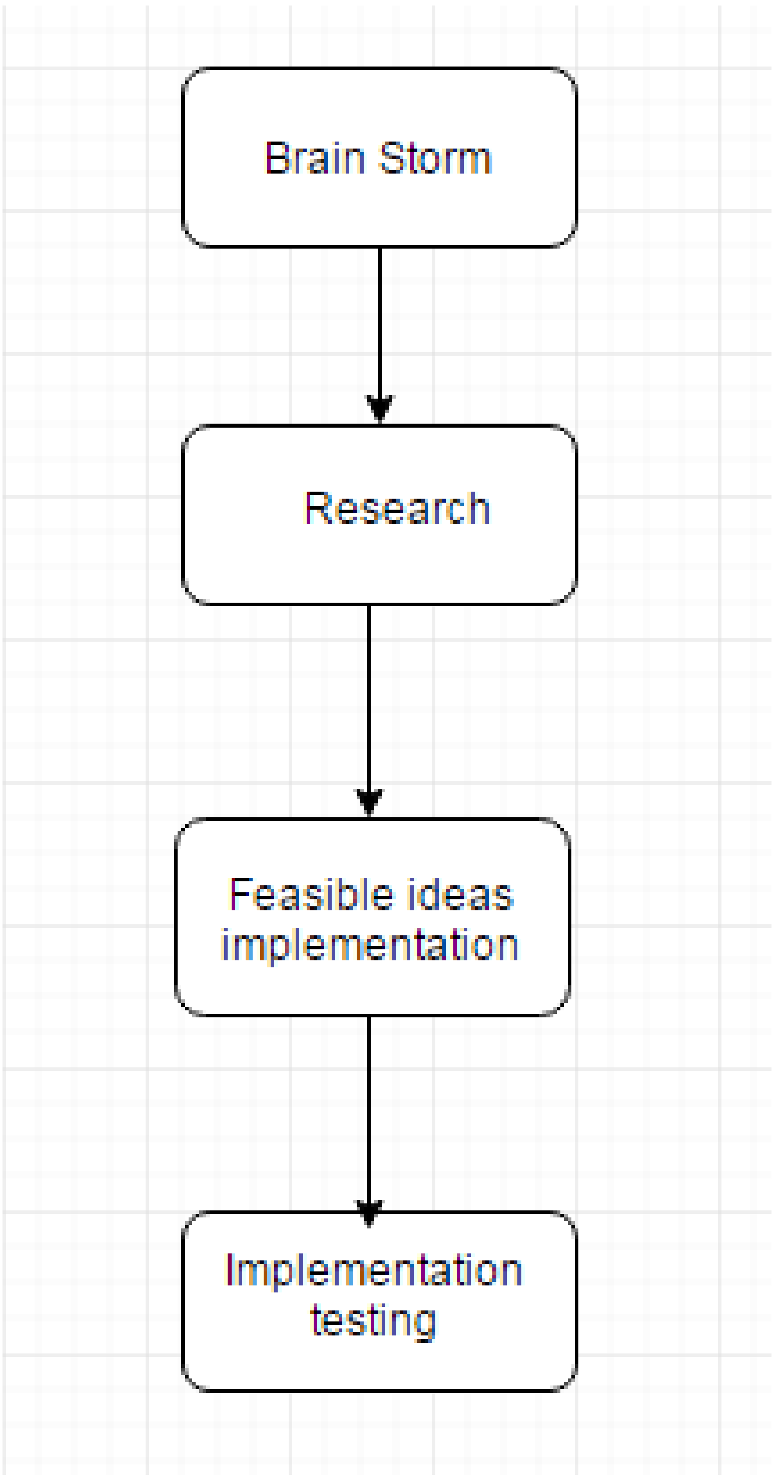


Figure 1.2: Study flow chart

Chapter 2

Brain Storm

Before starting research and ideas implementation, some of the new ideas should generate to provide a direction to build up barriers against cheatings.

2.1 Similarity Checking

The most straightforward idea that appeared into our mind is to check the source code for similarity. Because different courses would have different assignments, then the assignment difficult may differ to each other. Base on course requirements, setting up similarity tolerance between source codes as one barrier to against cheating.

2.2 Programming Variation

For the same programming language, source code could be possibly found from websites. To determine that if the answers to Code Runner questions could be easily found on the website, some research have to be done.

2.3 Functionality Addition

Additional functionality could be possibly added into Code Runner platform (code implementation) so that Code Runner is able to have several options for one question, and each option has similar difficulty to each other. Assign each option to every student, in this case, most of the students would have different questions(options), which reduces the possibility of assignment cheating.
The tables in next page gives the layout of ideas so far.

Table 2.1: **Same Assignment**

Cheating ways	Difficulty of Cheat	Prevention	Changes	Feasibility
Submission Copy and paste	Easy	Similarity checking	Sandbox to check codes similarity	Common way against cheating. Research required
Others who have finished assignment coding for students	Easy	Assignment interview	Extra time required	Further discussion

Table 2.2: **Different Assignments**

Cheating ways	Difficulty of Cheat	Prevention	Changes	Feasibility
Looking for similar solutions	Intermediate	Everyone has unique assignment	Create more variants in one question	Unique assignment reduces cheating probability
Online solutions or languages translation	Hard	Pre-defined questions preferred	define class used in Code Runner	Research required to see if it's feasible

Chapter 3

Research

Research for this chapter is done in ground lab room, Department of Computer Science, the University of Auckland.

3.1 Similarity Checking Research

3.1.1 Research Process

This study will focus on the coding part in Code Runner. Most of Code Runner users are Stage One students, and programming coding is the main part for those Code Runner students. In order to reduce the research error, the obtained dataset should follow some basic rules.

- People from different backgrounds, did not know each other.
- Source code obtained should be comparable, Compsci105 was selected.
- Source code with different lengths should be selected including longest and shortest ones.

3.1.2 Analysis

In order to compare the similarity between different source codes, one online tool,called CopyScape [4], is used to determine the similarity in percentage.

The shortest code length which is 3 coding lines is firstly being checked, the similarity is 100%, which is normally acceptable due to short program length. Then different source codes with different lengths are compared, the average similarity percentage reaches 77%, while the longest program gives the similarity with around 56.5%. There is a high probability that Code Runner submissions are similar even though student are not cheating. The summarized statistic table is presented below:

Table 3.1: Summarized Table

Program Length	Similarity Percentage
3 (shortest)	100%
3 lines - 67 lines	68% - 77%
67 lines	56.5%
Median length: 30 lines	Median similarity: 76.7%

Therefore, students who use Code Runner for assignments normally write very short programs. We determined that the median coding length in Code Runner is around 25 to 30 lines. This implies that high percentage similar of the submissions would occur. Furthermore, all courses are using the same question for all participants. Since students in each course were lectured by same lecturers, reading the same slides, attend same tutorials and see same examples, it is highly likely to have the high similarity coding results if the median program length is relatively short. In this case, even though students are not cheating each other, a high similarity is still possible.

3.1.3 Conclusion

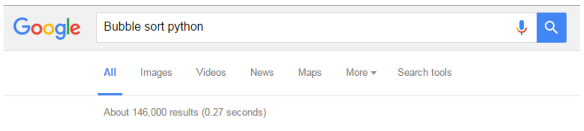
Similarity checking in Code Runner server is not a feasible anti-cheating approach. A high similarity between submissions due to short program length, questions are same and the solution is mostly based on lecture topics.

3.2 Programming Variation Research

3.2.1 Research Process

Three example types of questions were selected for our research. Instead of random selection, we selected questions from the typical topics, such as the recursive question, pre-defined question. Also, three common but popular sorting ways, Bubble-sort, Insertion-sort and Merge-sort, which is an important topic, were selected as well.

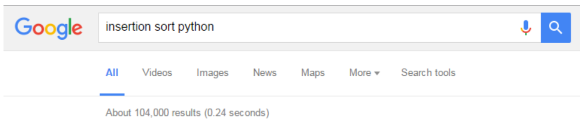
After question selection, we started searching the solutions or available sources on the website. Although Python is used for Compsi105 coding, different solutions of programming languages were still searched. First of all, three sorting algorithms were searched, and the answers(source code) that presented and would be able to pass the Code Runner questions as these codes have passed all test cases that we used to test the code accuracy. Three search terms and quantitative data of hits to pages are presented in Figure 3.1, Figure 3.2, Figure 3.3.Three searched source codes are



Bubble sort python

About 146,000 results (0.27 seconds)

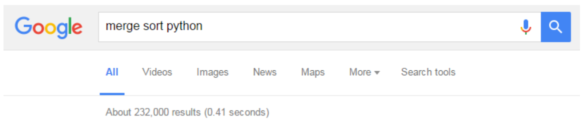
Figure 3.1: Search term and hits to pages for bubble sort



insertion sort python

About 104,000 results (0.24 seconds)

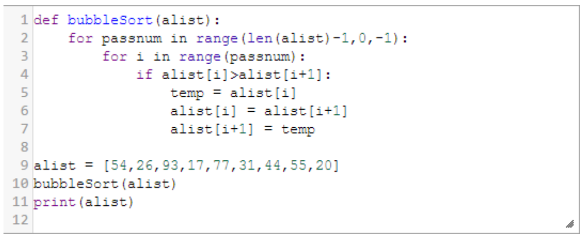
Figure 3.2: Search term and hits to pages for insertion sort



merge sort python

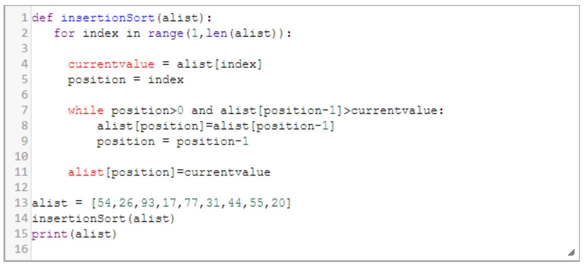
About 232,000 results (0.41 seconds)

Figure 3.3: Search term and hits to pages for merge sort



```
def bubbleSort(alist):
    for passnum in range(len(alist)-1,0,-1):
        for i in range(passnum):
            if alist[i]>alist[i+1]:
                temp = alist[i]
                alist[i] = alist[i+1]
                alist[i+1] = temp

alist = [54,26,93,17,77,31,44,55,20]
bubbleSort(alist)
print(alist)

```

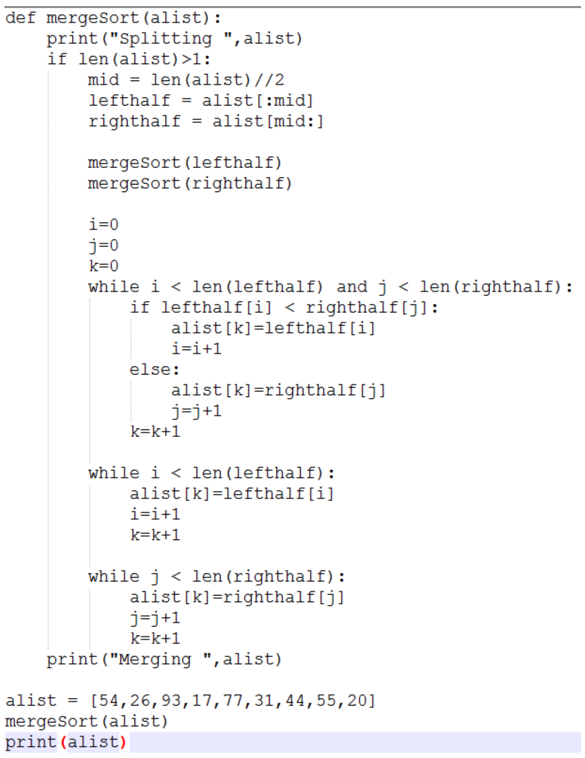
Figure 3.4: Searched source code for bubble sort



```
def insertionSort(alist):
   for index in range(1,len(alist)):

     currentvalue = alist[index]
     position = index

     while position>0 and alist[position-1]>currentvalue:
         alist[position]=alist[position-1]
         position = position-1

     alist[position]=currentvalue

alist = [54,26,93,17,77,31,44,55,20]
insertionSort(alist)
print(alist)

```

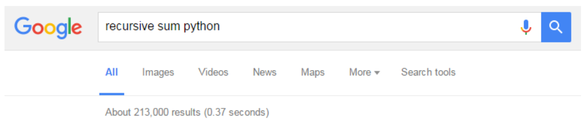
Figure 3.5: Searched source code for insertion sort

presented in Figure 3.4, Figure 3.5, Figure 3.6. [5]
Secondly, solutions of the recursive algorithm for a sum-up question has been searched from the website. More than one programming language gives the solution and explanation for the recursive algorithm of sum-up including Python, Java,etc. Research term and hits to pages are presented in Figure 3.7. Source code is presented in Figure 3.8. [5]
However, pre-defined questions were not easily to search the solution codes as each



```
def mergeSort(alist):
    print("Splitting ",alist)
    if len(alist)>1:
        mid = len(alist)//2
        lefthalf = alist[:mid]
        righthalf = alist[mid:]

        mergeSort(lefthalf)
        mergeSort(righthalf)

        i=0
        j=0
        k=0
        while i < len(lefthalf) and j < len(righthalf):
            if lefthalf[i] < righthalf[j]:
                alist[k]=lefthalf[i]
                i=i+1
            else:
                alist[k]=righthalf[j]
                j=j+1
            k=k+1

        while i < len(lefthalf):
            alist[k]=lefthalf[i]
            i=i+1
            k=k+1

        while j < len(righthalf):
            alist[k]=righthalf[j]
            j=j+1
            k=k+1
    print("Merging ",alist)

alist = [54,26,93,17,77,31,44,55,20]
mergeSort(alist)
print(alist)
```

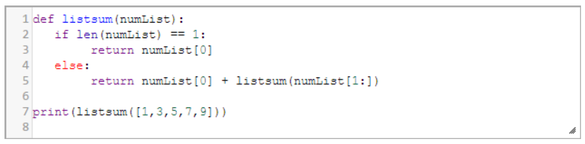
Figure 3.6: Searched source code for merge sort



recursive sum python

About 213,000 results (0.37 seconds)

Figure 3.7: Search term and hits to pages for sum recursive



```
def listsum(numList):
   if len(numList) == 1:
        return numList[0]
   else:
        return numList[0] + listsum(numList[1:])

print(listsum([1,3,5,7,9]))
```

Figure 3.8: Searched source code for recursive

pre-defined question requires using the same method and variable names from class which is already defined by question creators.

3.2.2 Analysis

Based on the result of research and source obtained, students were asked to solve the questions based on lecture topics in Code Runner. Questions could be popular or rare, and because of this reason, popular topics are easy to find available source while rare ones could only find the general description. By considering three sorting ways in this research, the full source code can be found on the websites. it is easy to do programming languages translation by translating known code to other languages. It reveals that programming languages translation is an unavoided personal skill which we are not able to detect. One the other hand, if we look at the research result of pre-defined questions. Code Runner actually defined its own class for students to solve the question based on the variables and method names which were already provided by questions. In this case, it is highly unlikely to get the full source code but conception from the Internet, which is helpful for Code Runner participants as it somehow encourages students to solve the problems independently by reviewing the lecture slides, recordings, and try to understand the provided class.

Conclusion

Answers to Code Runner questions are possibly being obtained on the websites. The programming languages translation skill cannot be detected by Code Runner. Pre-defined questions become a preferred question creation way in Code Runner, it prevents students from searching source code online, instead, encourages participants to do more revision and learn themselves (individual works). This implies that pre-defined questions is another barrier to against cheating, and should be widely used in Code Runner. However, it can't prevent copy-paste cheating, hence, some functionality should be implemented into Code Runner platform.

Chapter 4

Anti-cheat Idea Description

4.1 Idea 1

The idea here is to create more than one option for one question creation, and each option contains different variants. For example, lecturer or administrator could create Boolean question with options Even and Odd, for each option, has its own test cases and expected answers. Then assign options to different students to make sure every student is offered different assignments. In this case, the probability of assignment cheating will be reduced, and that means the barrier against cheating is pushed up. The requirements of the implementation are shown below.

Requirements:

- Even difficulty for assignment variants.
- Still easy to use.
- Maintain the functionality that Code Runner provides.
- Possible to integrate Moodle/CodeRunner.

Chapter 5

Idea Implementation

The most promising idea to implement for Code Runner is shown in Chapter 4 Section 'Idea 1' to make sure every participant gets different questions when they are doing assignments. In order to realize this idea, more functionality should be added to question creation page in Code Runner.

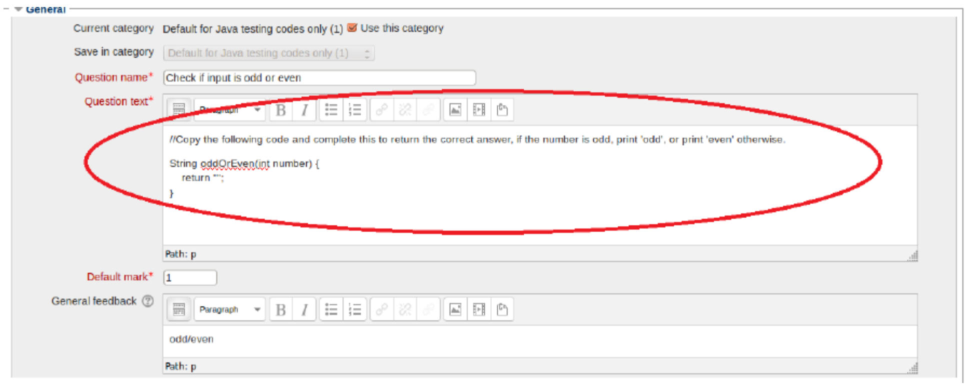
5.1 Idea Description

5.2 Proof of Concept

The idea above relates to question creation page but other functionalities are possible to remain what they used to be in Code Runner. Since Code Runner is a large question type in Moodle, if we try to modify the elements in question creation page, other related functions and arrays type should be modified as well, lots of works need to be done in order to build up the barrier. We do not know yet if the idea will work in Code Runner, thus a test should be made before we move to Code Runner, creating one prototype. Here we will use Java to build the prototype.
For the prototype, the GUI mock-up should be same as Code Runner but additional 'Option' input area for question creators. Also, it might be good to keep question options in the teachers interface to show all question variants. While student interface should be equivalent as Code Runner presents currently.

Because Java is chosen as the language to build up the prototype, as an open source library in Java, JComponent is used to build the interface. The GUI can be presented properly using two components of JComponent, JFrame and JPanel. Java connection library allows Java programming to easily and fast connect to MySql database that Code Runner uses to maintain all question variants, to connect to MySql database is the most significant step for prototype build-up.

The following images show the interface of question creation that presents in Code Runner, which is what the prototype supposes to have as the outcome.


General

Current category: Default for Java testing codes only (1) ✓ Use this category

Save in category: Default for Java testing codes only (1)

Question name*: Check if input is odd or even

Question text*:

```
//Copy the following code and complete this to return the correct answer, if the number is odd, print 'odd', or print 'even' otherwise.

String oddOrEven(int number) {
    return "";
}
```

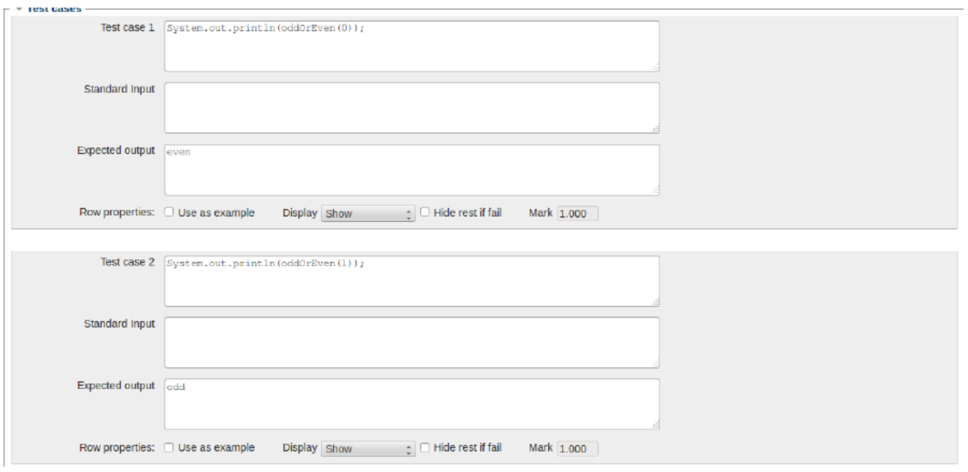
Path: p

Default mark*: 1

General feedback: odd/even

Path: p

Figure 5.1: Question Creation page in Code Runner


Test cases

Test case 1: `System.out.println(oddOrEven(0));`

Standard Input:

Expected output: even

Row properties: ☐ Use as example Display: Show ☐ Hide rest if fail Mark: 1.000

Test case 2: `System.out.println(oddOrEven(1));`

Standard Input:

Expected output: odd

Row properties: ☐ Use as example Display: Show ☐ Hide rest if fail Mark: 1.000

Figure 5.2: Question Creation page in Code Runner

As what have presented above, question name, question description and test cases are all necessary parts that will be inserted into MySql database for students answer comparison. Therefore, in the prototype, we not only have to present proper GUI mock-up, but also have to make it functional, which indicates that all inputs from question creation text area in GUI mock-up should be also inserted into corresponding tables as what Code Runner does, and should be able to run students input as Java programming.

5.3 GUI Mock-up Implementation

JFrame and JPanel in Java library are mainly used to present interface of question creation, 'Option' functionality is added into interface, and for each option created, the variants records for the option will be inserted into local table (JTable, not MySql database at the moment) which allows teachers to view all variants of corresponding options created. Furthermore, 'Preview' button will bring teacher view to student view (student Code Runner assignment interface) for further testing. The prototype interface shows below.

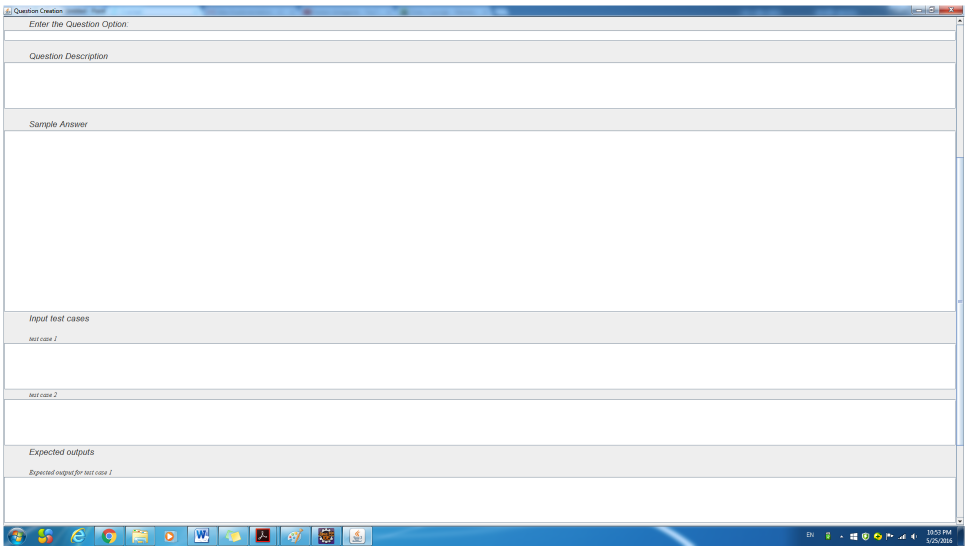
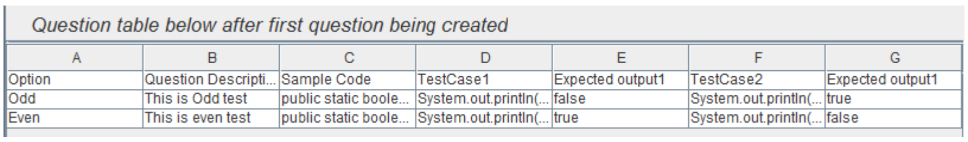


Figure 5.3: Question Creation page at GUI mock-up

The variants in teachers view (local database) will be stored once option created and displayed at question creation page, shows below.(Odd and Even options as the examples)



Question table below after first question being created

A	B	C	D	E	F	G
Option	Question Descripti...	Sample Code	TestCase1	Expected output1	TestCase2	Expected output1
Odd	This is Odd test	public static boole...	System.out.println(...	false	System.out.println(...	true
Even	This is even test	public static boole...	System.out.println(...	true	System.out.println(...	false

Figure 5.4: Local variants database(JTable)

The student view can be shown up by 'preview' function which is also available in Code Runner. The emulated student assignment view shows in Figure 5.5.

After GUI mock-up creation, the basic framework of Code Runner prototype is successfully implemented, and ready to build functionality into GUI mock-up so that

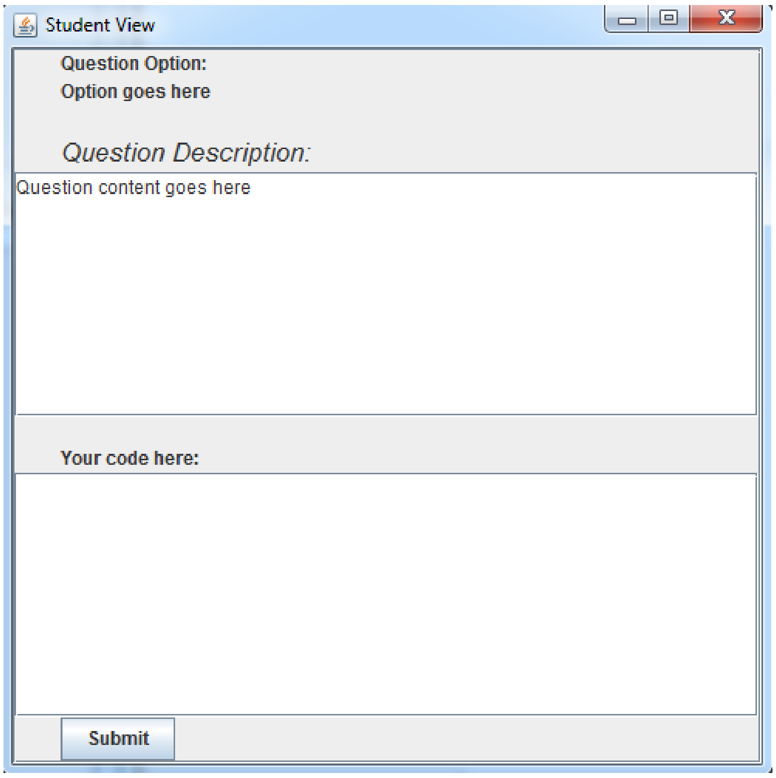


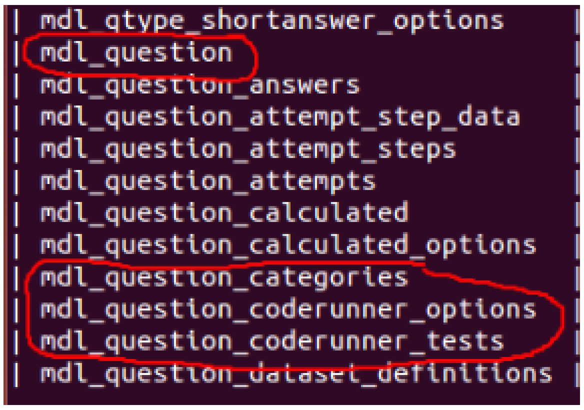
Figure 5.5: Student question view

it is able to deal with the input data and output comparison as what Code Runner does.

5.4 Functionality Build-in

5.4.1 Preparation

Before implementing the functionality, MySql database structure needs to be reviewed, as all data display in Code Runner are retrieved from the database, because what we need to focus on is question creation part. By adding questions to a specific course, 4 tables are modified. This indicates that those 4 tables are related to question creation and answers comparison, because the expected outputs are also defined in question creation part. Thus, when implementing functionality build-up, question variants inputs should be treated as records and inserted into corresponding tables as Code Runner did for further discussion. The following 4 circled tables in MySql database will be used.



```
| mdl_qtype_shortanswer_options          |
| mdl_question                           |
| mdl_question_answers                   |
| mdl_question_attempt_step_data         |
| mdl_question_attempt_steps             |
| mdl_question_attempts                  |
| mdl_question_calculated                |
| mdl_question_calculated_options        |
| mdl_question_categories                |
| mdl_question_coderunner_options        |
| mdl_question_coderunner_tests          |
| mdl_question_dataset_definitions       |
```

Figure 5.6: MySql database tables

5.4.2 Functionality of Prototype

DataBase Schema

The database schema of 4 tables above are needed, to insert records into the corresponding tables. The following 4 tables(Table 4.1-Table 4.4) present the schema for each useful header that probably be used in further.

Table 5.1: mdl_question schema

Field	Type	Null	Key
id	bigint(10)	No	PRI
category	bigint(10)	No	MUL
name	varchar(255)	No	MUL
questiontext	longtext	No	
qtype	varchar(20)	No	

Mdl_question table above gives several headers that used in question creation part. This is the question table which maintains all questions id including Code Runner questions etc. Category header indicates the course number in Code Runner such as Compsci101. Name and questiontext headers gives the question name and question description. The qtype in this project is defined as CodeRunner.

Mdl_question_categories gives the course description with id as primary key. And maintain the course name.

Mdl_question_coderunner_tests links the question test cases to corresponding questions by building up the relationship chain. The testcode stores the test cases. The

Table 5.2: mdl_question_categories schema

Field	Type	Null	Key
id	bigint(10)	No	PRI
name	varchar(255)	No	
contextid	bigint(10)	No	MUL

Table 5.3: mdl_question_coderunner_tests schema

Field	Type	Null	Key
id	bigint(10)	No	PRI
questionid	bigint(10)	No	MUL
testcode	longtext	YES	
stdin	longtext	YES	
expected	longtext	YES	
mark	decimal(8,3)	No	

stdin and expected headers indicate the sample answer and expected output for each test case respectively. The mark header in Code Runner question type is using 'allornothing' model which means students are able to gain full mark only if their submission passes all test cases, otherwise they got zero.

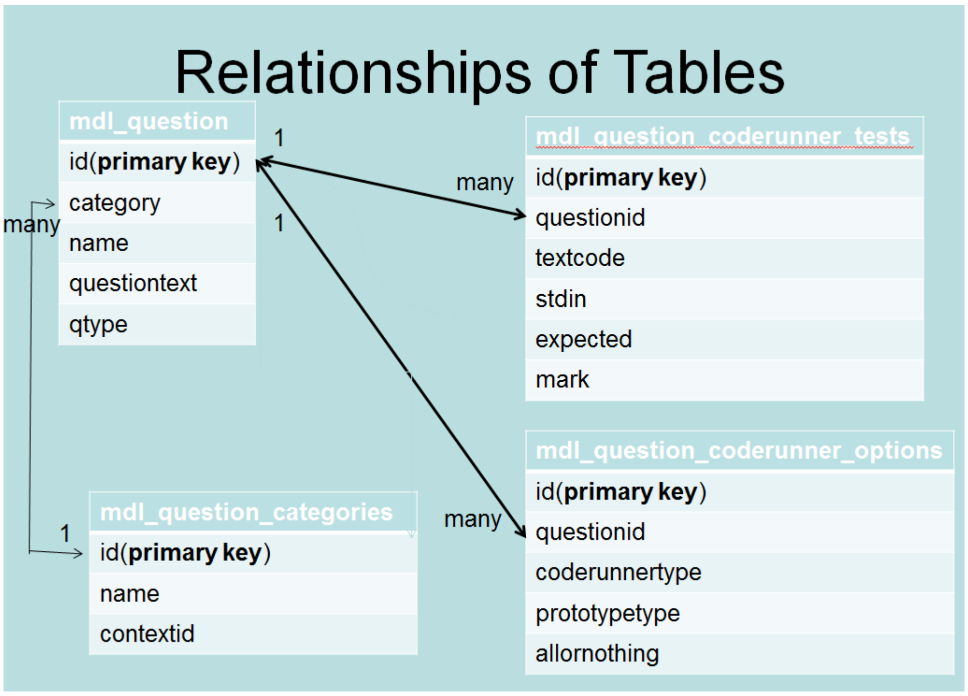
Table 5.4: mdl_question_coderunner_options schema

Field	Type	Null	Key
id	bigint(10)	No	PRI
questionid	bigint(10)	No	MUL
coderunnertype	varchar(255)	NO	MUL
prototypetype	tinyint(1)	No	MUL
allornothing	tinyint(1)	NO	

Mdl_question_coderunner_options table gives the options for question creation in Code Runner. The coderunnertype provides the options of programming languages such as Java, Python. The prototypetype normally is defined as 1 if the question is created and allornothing is the one that has been declared above.

The entity relationship diagram(ERD) for above four tables are presented in Figure 5.7.

After related schema has been checked, the functionality is now ready to implement.



mdl_question
id(**primary key**)
category
name
questiontext
qtype

mdl_question_coderunner_tests
id(**primary key**)
questionid
textcode
stdin
expected
mark

mdl_question_coderunner_options
id(**primary key**)
questionid
coderunnertype
prototypetype
allornothing

mdl_question_categories
id(**primary key**)
name
contextid

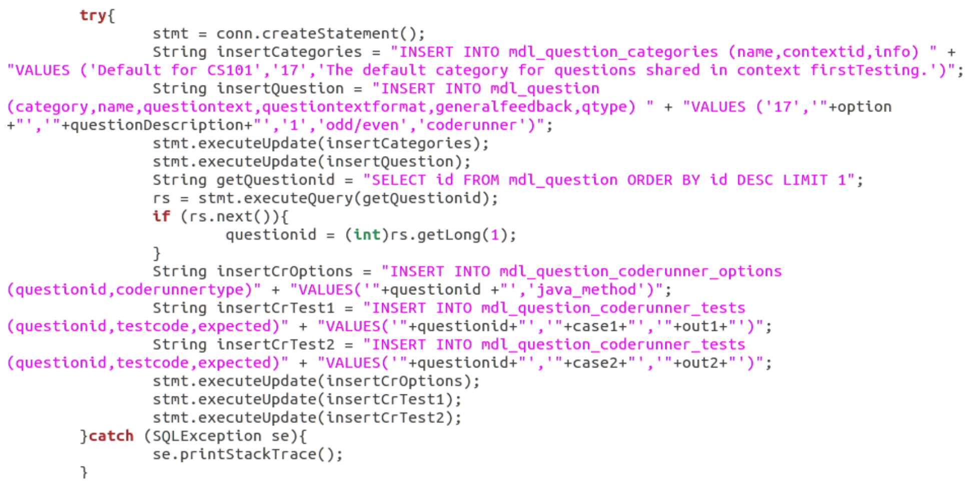
Figure 5.7: Relationship of tables

Functionality Implementation

Based on the database schema above, in the prototype, necessary SQL should be coded to make sure that each button click will lead to records insert or delete from MySql Database. For example, for each time the 'Save' button is clicked, the SQL presented in Figure 5.8 will be executed by MySql Database connection to insert input variants. Where 'Default for CS101' is the already created course in Code Runner.

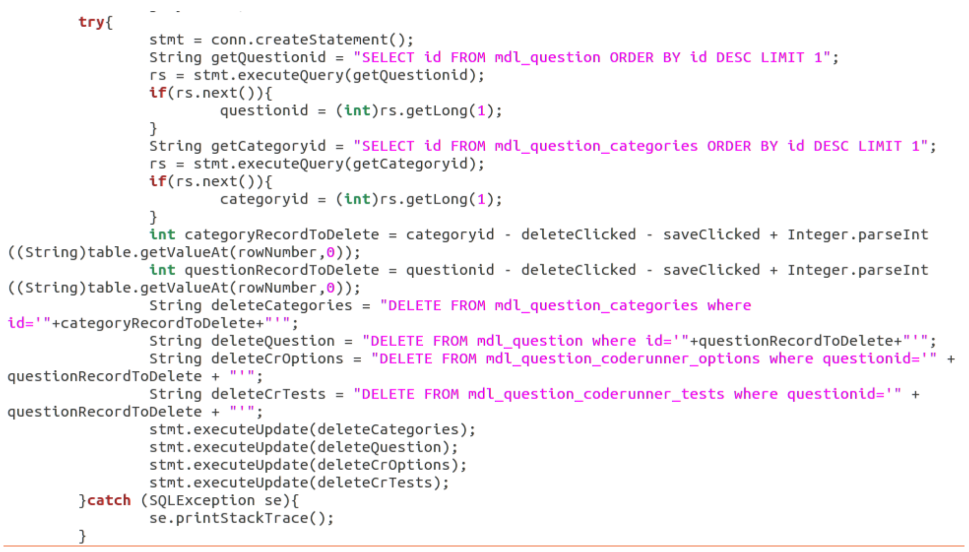
For 'Deletion' button clicked, corresponding records in the local database and MySql Database server should both be removed. The SQL presented in Figure 5.9 will be executed by MySql Database server to delete the selected record. Furthermore, refer to Figure 4.5, the student input should be added into a runnable Java program once 'submit' button is clicked. The prototype should be able to run students input at the background and bring the feedback immediately. By checking the way that Code Runner did is to create a new Java file contains student input and run the file at the Sandbox, which looks as presented in Figure 5.10.

In order to follow the way that Code Runner did, the codes presented in Figure 5.11 will be creating a java file which contains student input and executed it at the Java Running Environment(JRE). Finally, the program will retrieve the expected answer



```
        try{
                stmt = conn.createStatement();
                String insertCategories = "INSERT INTO mdl_question_categories (name,contextid,info) " +
"VALUES ('Default for CS101','17','The default category for questions shared in context firstTesting.')";
                String insertQuestion = "INSERT INTO mdl_question
(category,name,questiontext,questiontextformat,generalfeedback,qtype) " + "VALUES ('17','"+option
+"','"+questionDescription+"','1','odd/even','coderunner')";
                stmt.executeUpdate(insertCategories);
                stmt.executeUpdate(insertQuestion);
                String getQuestionid = "SELECT id FROM mdl_question ORDER BY id DESC LIMIT 1";
                rs = stmt.executeQuery(getQuestionid);
                if (rs.next()){
                        questionid = (int)rs.getLong(1);
                }
                String insertCrOptions = "INSERT INTO mdl_question_coderunner_options
(questionid,coderunnertype)" + "VALUES('"+questionid +"','java_method')";
                String insertCrTest1 = "INSERT INTO mdl_question_coderunner_tests
(questionid,testcode,expected)" + "VALUES('"+questionid+"','"+case1+"','"+out1+"')";
                String insertCrTest2 = "INSERT INTO mdl_question_coderunner_tests
(questionid,testcode,expected)" + "VALUES('"+questionid+"','"+case2+"','"+out2+"')";
                stmt.executeUpdate(insertCrOptions);
                stmt.executeUpdate(insertCrTest1);
                stmt.executeUpdate(insertCrTest2);
        }catch (SQLException se){
                se.printStackTrace();
        }
```

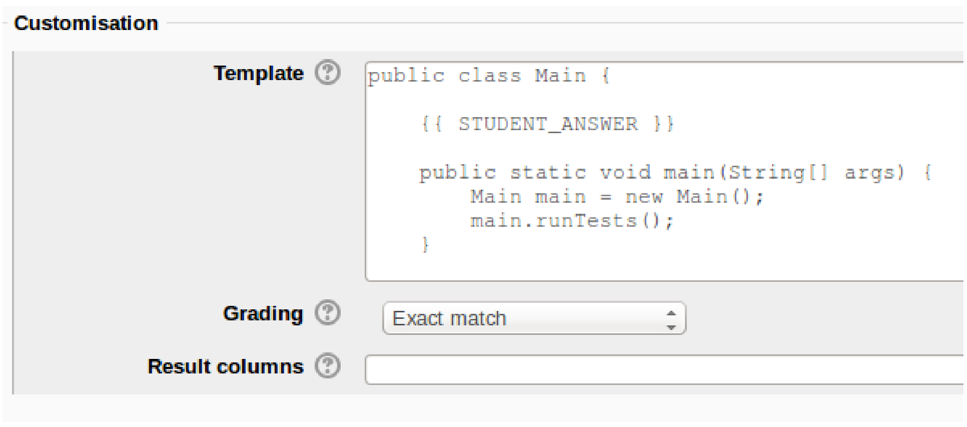
Figure 5.8: insertion SQL in java program



```
        try{
                stmt = conn.createStatement();
                String getQuestionid = "SELECT id FROM mdl_question ORDER BY id DESC LIMIT 1";
                rs = stmt.executeQuery(getQuestionid);
                if(rs.next()){
                        questionid = (int)rs.getLong(1);
                }
                String getCategoryid = "SELECT id FROM mdl_question_categories ORDER BY id DESC LIMIT 1";
                rs = stmt.executeQuery(getCategoryid);
                if(rs.next()){
                        categoryid = (int)rs.getLong(1);
                }
                int categoryRecordToDelete = categoryid - deleteClicked - saveClicked + Integer.parseInt
((String)table.getValueAt(rowNumber,0));
                int questionRecordToDelete = questionid - deleteClicked - saveClicked + Integer.parseInt
((String)table.getValueAt(rowNumber,0));
                String deleteCategories = "DELETE FROM mdl_question_categories where
id='"+categoryRecordToDelete+"'";
                String deleteQuestion = "DELETE FROM mdl_question where id='"+questionRecordToDelete+"'";
                String deleteCrOptions = "DELETE FROM mdl_question_coderunner_options where questionid='" +
questionRecordToDelete + "'";
                String deleteCrTests = "DELETE FROM mdl_question_coderunner_tests where questionid='" +
questionRecordToDelete + "'";
                stmt.executeUpdate(deleteCategories);
                stmt.executeUpdate(deleteQuestion);
                stmt.executeUpdate(deleteCrOptions);
                stmt.executeUpdate(deleteCrTests);
        }catch (SQLException se){
                se.printStackTrace();
        }
```

Figure 5.9: deletion SQL in java program

by unique questionid from MySql Database server and compare answers in order to bring the feedback to the terminal.
By now, it seems proof-of-concept process has been done, all required user interfaces and functionality have been implemented in Java. The next step should bring the proof-of-concept program to testing part in terms of checking MySql Database records and answers comparison results.



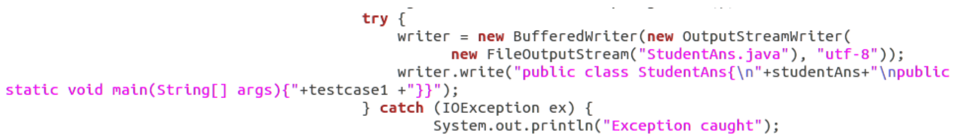
Customisation

Template
```
public class Main {

    {{ STUDENT_ANSWER }}

    public static void main(String[] args) {
        Main main = new Main();
        main.runTests();
    }
```

Grading: Exact match

Result columns:

Figure 5.10: deletion SQL in java program



```
                                          try {
                                              writer = new BufferedWriter(new OutputStreamWriter(
                                                  new FileOutputStream("StudentAns.java"), "utf-8"));
                                              writer.write("public class StudentAns{\n"+studentAns+"\npublic
static void main(String[] args){"+testcase1 +"}}");
                                          } catch (IOException ex) {
                                                System.out.println("Exception caught");
```

Figure 5.11: Java file making code in java program

Chapter 6

Prototype Testing

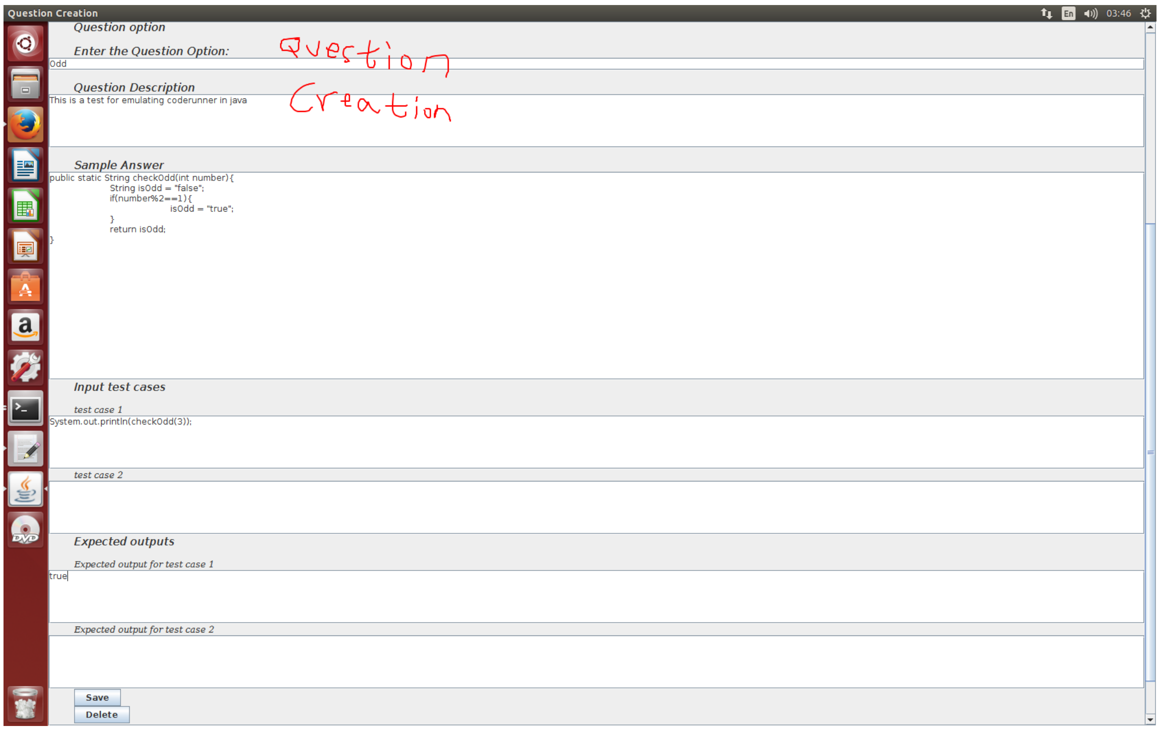
In order to test if question variants from text area of GUI mock-up are successfully inserted into MySql Database server and the deletion works, we have to go back to corresponding tables for records checking.

6.1 Insertion Testing

Based on GUI mock-up, Figure 6.1 shows the pen-testing input for question creation. After all variants from text areas are saved, MySql Database shows that records from corresponding tables that declared above have been modified. Inserted records have been circled in Figure 6.2. Refer to input variants, the inserted records are the data from the question that we just created. Therefore, the insertion has been successfully completed.

6.2 Student Answer Testing

After the question has been created and inserted into the server, we go to student question view to check if the program is able to create Java file contains student answer and run the new student Java program at the background. Figure 6.3 presents the testing code which refers to the question that just created. And a new Java file contains the testing code has been created and presented in Figure 6.4.
After the 'Submit' button that displays is clicked in Figure 5.3, the feedbacks bring back to the terminal for both correct and incorrect answers, which is presented in Figure 6.5 and Figure 6.6 respectively.
By now, variants creation and partial sandbox functionality(read and execute students input) are both working well and able to connect MySql Database server. Because the created variants are able to present to students and insert to the database tables. And the prototype is able to execute answer from students and gives feedback to



Question option

Enter the Question Option:

Odd

Question Description

This is a test for emulating coderunner in java

Sample Answer

```
public static String checkOdd(int number){
          String isOdd = "false";
          if(number%2==1){
                    isOdd = "true";
          }
          return isOdd;
}
```

Input test cases

test case 1

```
System.out.println(checkOdd(3));
```

test case 2

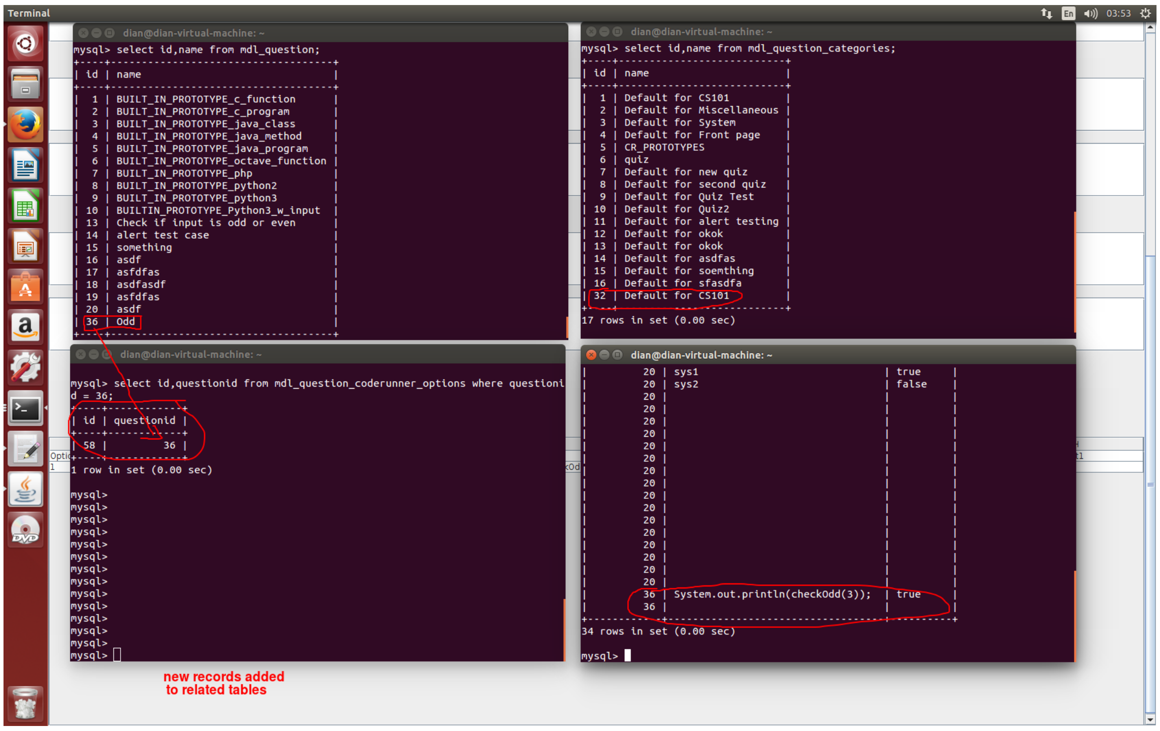
Expected outputs

Expected output for test case 1

true

Expected output for test case 2

Figure 6.1: Testing variants input


Terminal
03:53
dian@dian-virtual-machine: ~
dian@dian-virtual-machine: ~
dian@dian-virtual-machine: ~
dian@dian-virtual-machine: ~

```
mysql> select id,name from mdl_question;
+----+--------------------------------------+
| id | name                                 |
+----+--------------------------------------+
|  1 | BUILT_IN_PROTOTYPE_c_function        |
|  2 | BUILT_IN_PROTOTYPE_c_program         |
|  3 | BUILT_IN_PROTOTYPE_java_class        |
|  4 | BUILT_IN_PROTOTYPE_java_method       |
|  5 | BUILT_IN_PROTOTYPE_java_program      |
|  6 | BUILT_IN_PROTOTYPE_octave_function   |
|  7 | BUILT_IN_PROTOTYPE_php               |
|  8 | BUILT_IN_PROTOTYPE_python2           |
|  9 | BUILT_IN_PROTOTYPE_python3           |
| 10 | BUILTIN_PROTOTYPE_Python3_w_input    |
| 13 | Check if input is odd or even        |
| 14 | alert test case                      |
| 15 | something                            |
| 16 | asdf                                 |
| 17 | asfdfas                              |
| 18 | asdfasdf                             |
| 19 | asfdfas                              |
| 20 | asdf                                 |
| 36 | Odd                                  |
+----+--------------------------------------+
```

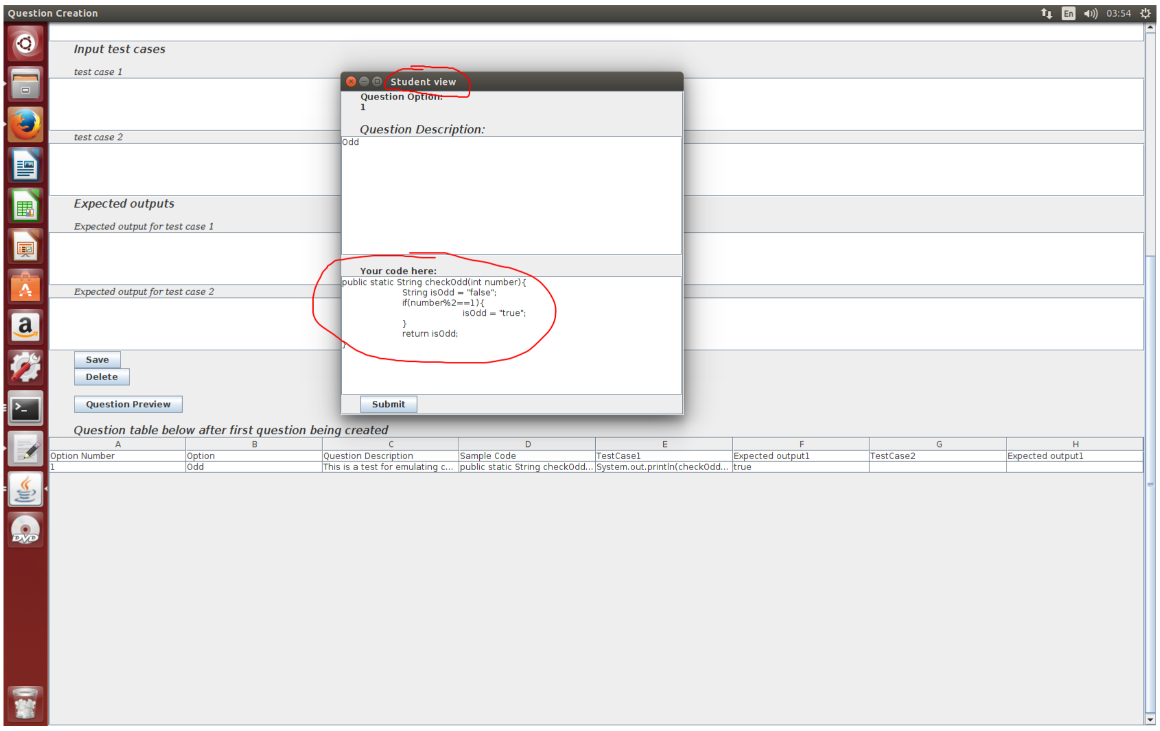
```
mysql> select id,name from mdl_question_categories;
+----+---------------------------+
| id | name                      |
+----+---------------------------+
|  1 | Default for CS101         |
|  2 | Default for Miscellaneous |
|  3 | Default for System        |
|  4 | Default for Front page    |
|  5 | CR_PROTOTYPES             |
|  6 | quiz                      |
|  7 | Default for new quiz      |
|  8 | Default for second quiz   |
|  9 | Default for Quiz Test     |
| 10 | Default for Quiz2         |
| 11 | Default for alert testing |
| 12 | Default for okok          |
| 13 | Default for okok          |
| 14 | Default for asdfas        |
| 15 | Default for soemthing     |
| 16 | Default for sfasdfa       |
| 32 | Default for CS101         |
+----+---------------------------+
17 rows in set (0.00 sec)
```

```
mysql> select id,questionid from mdl_question_coderunner_options where questioni
d = 36;
+----+------------+
| id | questionid |
+----+------------+
| 58 |         36 |
+----+------------+
1 row in set (0.00 sec)

mysql>
mysql>
mysql>
mysql>
mysql>
mysql>
mysql>
mysql>
mysql>
mysql>
mysql>
mysql>
mysql>
mysql>
```

```
|          20 | sys1                              | true      |
|          20 | sys2                              | false     |
|          20 |                                   |           |
|          20 |                                   |           |
|          20 |                                   |           |
|          20 |                                   |           |
|          20 |                                   |           |
|          20 |                                   |           |
|          20 |                                   |           |
|          20 |                                   |           |
|          20 |                                   |           |
|          20 |                                   |           |
|          20 |                                   |           |
|          20 |                                   |           |
|          20 |                                   |           |
|          20 |                                   |           |
|          20 |                                   |           |
|          20 |                                   |           |
|          36 | System.out.println(checkOdd(3));  | true      |
|          36 |                                   |           |
+-------------+-----------------------------------+-----------+
34 rows in set (0.00 sec)

mysql>
```

new records added
to related tables

Figure 6.2: Insertion check



Question Option:
1

Question Description:
Odd

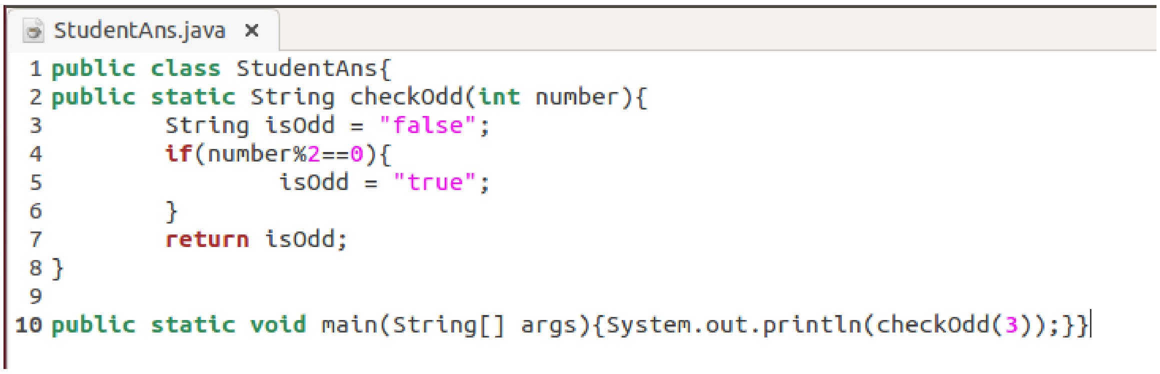
Your code here:

```
public static String checkOdd(int number){
                String isOdd = "false";
                if(number%2==1){
                                isOdd = "true";
                }
                return isOdd;
}
```

Question table below after first question being created

A	B	C	D	E	F	G	H
Option Number	Option	Question Description	Sample Code	TestCase1	Expected output1	TestCase2	Expected output1
1	Odd	This is a test for emulating c...	public static String checkOdd...	System.out.println(checkOdd...	true		

Figure 6.3: Testing code input



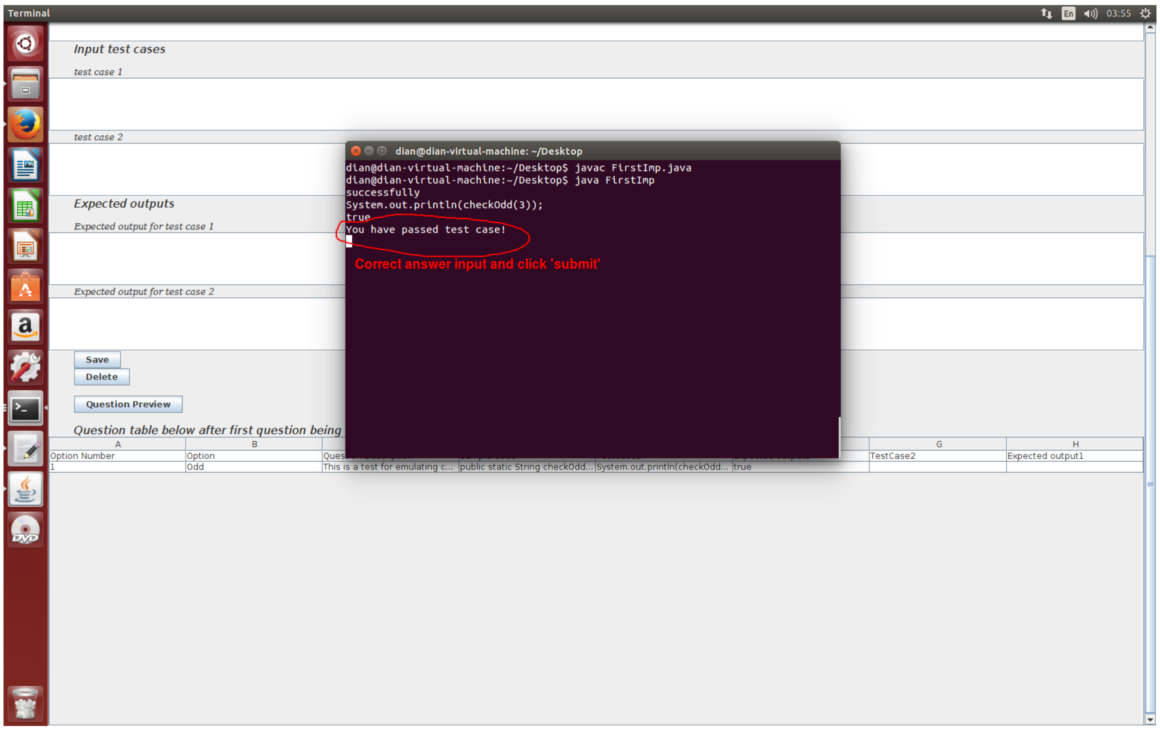
```
public class StudentAns{
public static String checkOdd(int number){
        String isOdd = "false";
        if(number%2==0){
                isOdd = "true";
        }
        return isOdd;
}

public static void main(String[] args){System.out.println(checkOdd(3));}}
```

Figure 6.4: Created Java program

the terminal. As Code Runner is also able to remove unnecessary variants from the server, which takes us to the last testing step.

6.3 Deletion Testing

Refer to GUI mock-up, the local database table (JTable) gives all the variants that created, if the deletion has been successfully done, record in JTable and MySql Database will be both removed after 'delete' button being clicked.



Input test cases

test case 1

test case 2

Expected outputs

Expected output for test case 1

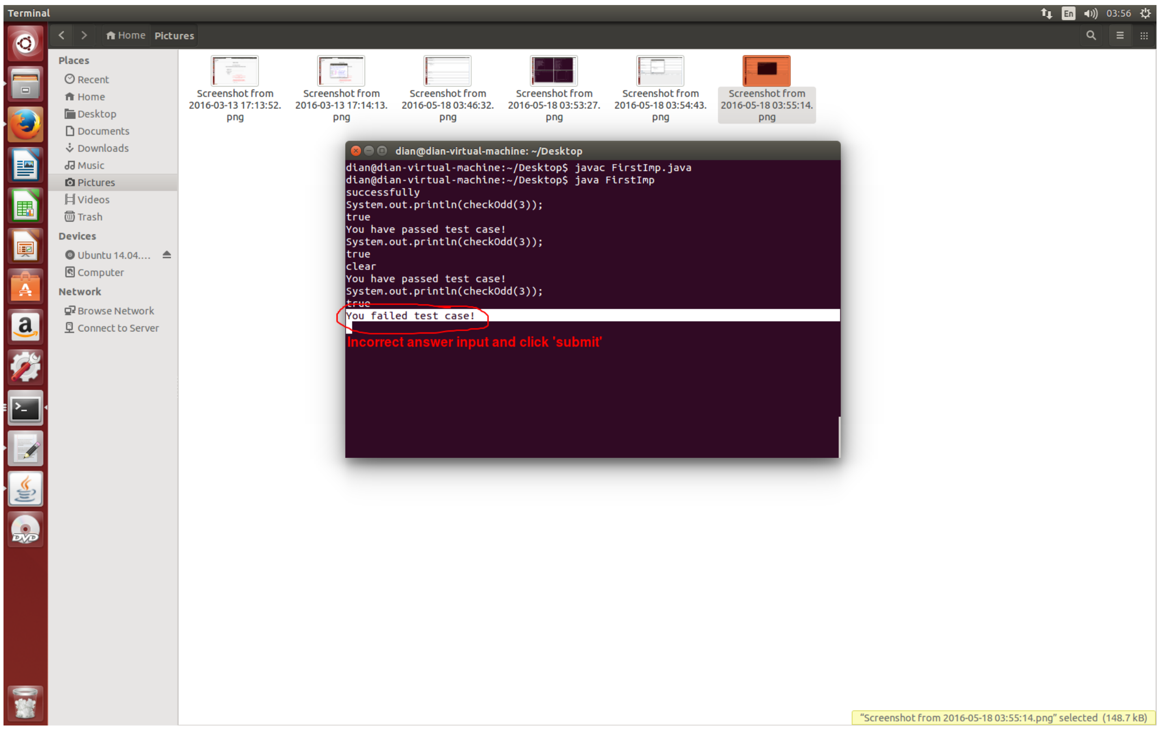
Expected output for test case 2

Question table below after first question being

```
dian@dian-virtual-machine:~/Desktop$ javac FirstImp.java
dian@dian-virtual-machine:~/Desktop$ java FirstImp
successfully
System.out.println(checkOdd(3));
true
You have passed test case!
```

A	B	C	D	E	F	G	H
Option Number	Option	Ques				TestCase2	Expected output1
1	Odd	This is a test for emulating c...	public static String checkOdd...	System.out.println(checkOdd...	true		

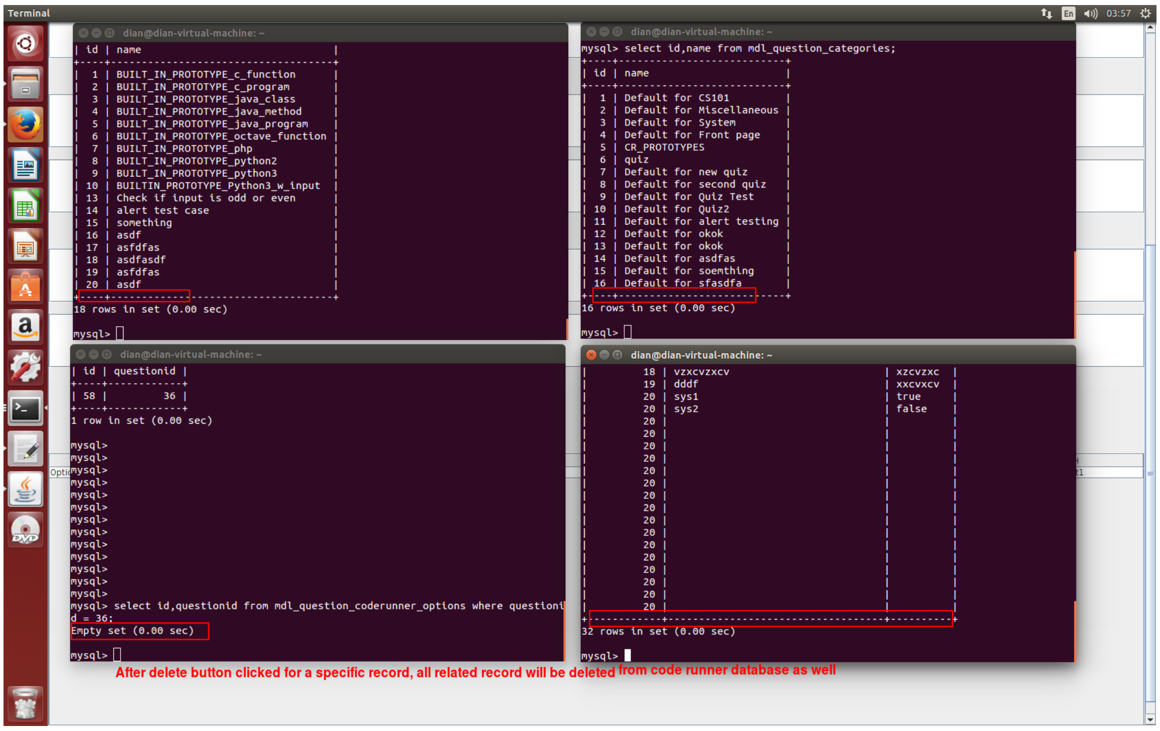
Figure 6.5: Feedback for correct answer


Terminal
03:56
Home
Pictures
Places
Recent
Home
Desktop
Documents
Downloads
Music
Pictures
Videos
Trash
Devices
Ubuntu 14.04....
Computer
Network
Browse Network
Connect to Server
Screenshot from 2016-03-13 17:13:52. png
Screenshot from 2016-03-13 17:14:13. png
Screenshot from 2016-05-18 03:46:32. png
Screenshot from 2016-05-18 03:53:27. png
Screenshot from 2016-05-18 03:54:43. png
Screenshot from 2016-05-18 03:55:14. png
"Screenshot from 2016-05-18 03:55:14.png" selected (148.7 kB)
Incorrect answer input and click 'submit'

```
dian@dian-virtual-machine:~/Desktop$ javac FirstImp.java
dian@dian-virtual-machine:~/Desktop$ java FirstImp
successfully
System.out.println(checkOdd(3));
true
You have passed test case!
System.out.println(checkOdd(3));
true
clear
You have passed test case!
System.out.println(checkOdd(3));
true
You failed test case!
```

Figure 6.6: Feedback for incorrect answer

From Insertion Testing part, it shows that 'Odd' option has been inserted, thus in this part, 'Odd' option becomes the deleting variant. After, 'delete' button has been clicked, 4 tables that show in Figure 5.2 is going to be checked again, the inserted variants disappeared which implies that deletion command in the prototype works in MySql Database server. Figure 6.7 presents the testing result.



```
| id | name                                 |
+----+--------------------------------------+
|  1 | BUILT_IN_PROTOTYPE_c_function        |
|  2 | BUILT_IN_PROTOTYPE_c_program         |
|  3 | BUILT_IN_PROTOTYPE_java_class        |
|  4 | BUILT_IN_PROTOTYPE_java_method       |
|  5 | BUILT_IN_PROTOTYPE_java_program      |
|  6 | BUILT_IN_PROTOTYPE_octave_function   |
|  7 | BUILT_IN_PROTOTYPE_php               |
|  8 | BUILT_IN_PROTOTYPE_python2           |
|  9 | BUILT_IN_PROTOTYPE_python3           |
| 10 | BUILTIN_PROTOTYPE_Python3_w_input    |
| 13 | Check if input is odd or even        |
| 14 | alert test case                      |
| 15 | something                            |
| 16 | asdf                                 |
| 17 | asfdfas                              |
| 18 | asdfasdf                             |
| 19 | asfdfas                              |
| 20 | asdf                                 |
+----+--------------------------------------+
18 rows in set (0.00 sec)

mysql>
```

```
mysql> select id,name from mdl_question_categories;
+----+---------------------------+
| id | name                      |
+----+---------------------------+
|  1 | Default for CS101         |
|  2 | Default for Miscellaneous |
|  3 | Default for System        |
|  4 | Default for Front page    |
|  5 | CR_PROTOTYPES             |
|  6 | quiz                      |
|  7 | Default for new quiz      |
|  8 | Default for second quiz   |
|  9 | Default for Quiz Test     |
| 10 | Default for Quiz2         |
| 11 | Default for alert testing |
| 12 | Default for okok          |
| 13 | Default for okok          |
| 14 | Default for asdfas        |
| 15 | Default for soemthing     |
| 16 | Default for sfasdfa       |
+----+---------------------------+
16 rows in set (0.00 sec)

mysql>
```

```
| id | questionid |
+----+------------+
| 58 |         36 |
+----+------------+
1 row in set (0.00 sec)

mysql>
mysql>
mysql>
mysql>
mysql>
mysql>
mysql>
mysql>
mysql>
mysql>
mysql>
mysql>
mysql>
mysql> select id,questionid from mdl_question_coderunner_options where questioni
d = 36;
Empty set (0.00 sec)

mysql>
```

```
|         18 | vzxcvzxcv                            | xzcvzxc  |
|         19 | dddf                                 | xxcvxcv  |
|         20 | sys1                                 | true     |
|         20 | sys2                                 | false    |
|         20 |                                      |          |
|         20 |                                      |          |
|         20 |                                      |          |
|         20 |                                      |          |
|         20 |                                      |          |
|         20 |                                      |          |
|         20 |                                      |          |
|         20 |                                      |          |
|         20 |                                      |          |
|         20 |                                      |          |
|         20 |                                      |          |
|         20 |                                      |          |
|         20 |                                      |          |
|         20 |                                      |          |
|         20 |                                      |          |
|         20 |                                      |          |
+------------+--------------------------------------+----------+
32 rows in set (0.00 sec)

mysql>
```

After delete button clicked for a specific record, all related record will be deleted from code runner database as well

Figure 6.7: Deletion testing

Combining all testing results above, the prototype is able to provide proper interfaces and deal with question creation, deletion and answers comparison tasks as exactly what Code Runner does. Proof-of-concept has been successfully completed in Java. This process proves that the idea from Brain Storm part is possible to implement in Code Runner site as Moodle is an open-source platform and MySql Database schema matches and allows income data from outside.

Chapter 7

Discussion

It believes that this anti-cheat system is more advanced as it maintain most of functionality of Code Runner system, except adding more than one option in question creation. As we known, there is also a similar system called 'Question Bank' exists in Code Runner, the main different between my anti-cheat system and Question Bank is that Question Bank asks question creator to create different questions. Hence, it cannot guarantee that the difficulty of each created question are similar. However, as each option bases on each question description, anti-cheat system is able to give similar assignments to student which avoids unfair assignments.

However, any anti-cheat system may not be able to guarantee that there is no one is possible to make cheat in Code Runner. So in this place, students could still cheat by asking someone else to do the assignment for them. Also, for Stage I courses, there are usually more than 500 students for each Computer Science course, it is highly unlikely that lecturer needs to create more than 500 options for each question, which implies that cheating is still possible by finding same option offered and copy-paste solution.

Overall, with anti-cheat system, the probability of cheating will be effectively reduced, which is what we supposed to have at the end of this Project.

Chapter 8

Further Work

As the proof-of-concept has been done, based on the ideas from Brain Storm chapter, the next step will be back to Code Runner platform. To implement the idea that used in proof-of-concept into Code Runner by using PHP programming skills, in order to make the 'Option' functionality realized for question creation in Code Runner. Apart from that, being able to assign each variant to an unique student in Code Runner. This is going to be the first step to do in the future.

Furthermore, more ideas against cheating probably need to be generated to prevent different cheating ways happening in Code Runner. Based on generated ideas, some more research has to be done to determine if ideas are feasible or not. Code implementations should keep going as more and more feasible ideas come up.

Chapter 9

Conclusion

Code Runner is a plugin of Moodle and widely used at the University of Auckland for programming assignments because of its convenient and safety. However, it is not able to detect cheating becomes the most significant weakness of Code Runner, and might lead to negative effects among students.

This study bases on the vulnerability of Code Runner to discuss some ideas that could be possible to prevent cheating in Code Runner. Research about similarity checking and programming variation have both been done. The research results show that due to short programming length, a high probability of similar submissions occurs frequently in Code Runner. It indicates that the idea about checking similarity is not an effective way to prevent cheating. Also, from second research result, students can possibly search solutions online or do programming languages translations to achieve higher marks. So it is unfair to hard working students and might cause faith lost in Code Runner.

More variants can be defined in one question became one feasible idea to be implemented. In order to prove that new functionality is possible to be added in Code Runner and matches the corresponding database schema, proof-of-concept process has been done by building up a prototype in Java program. And it has been successfully completed.

More feasible ideas need to be thinking of and research has to be done. Also, more implementations based on generated ideas and research results need to be done in Code Runner in the future.

Bibliography

[1] J.Cole, H.Foster, Using Moodle:Teaching with the popular open source course management system, 0'Reilly Media, Inc, 2007.

[2] M.Org, Pedagogy(2013). Retrieved from
URL http://docs.moodle.org/23/en/Pedagogy.

[3] R.Lobb,CODE RUNNER(2013). Retrieved From
URL https://github.com/trampgeek/CodeRunner.

[4] CopyScape. Online diff checker.
URL http://www.copyscape.com/compare.php.

[5] SortSearch. Retrieved from
URL http://interactivepython.org/runestone/static/pythonds/

[6] Wikipedia. Retrieved from
URL https://en.wikipedia.org/wiki/MySQL.