# Feature Enhancement of an E-learning Platform for Automatic Evaluation of Programming Assignments

**Master Thesis**

Submitted in partial fulfillment of the requirements of the degree of
Master of Technology

by

**Swaresh Uttam Sankpal**

**(153050085)**

under the guidance of

**Prof. Varsha Apte**


Master of Technology Programme in

Computer Science and Engineering

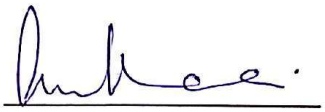
Indian Institute of Technology, Bombay

July, 2017

Dissertation Approval

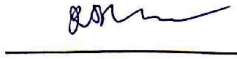
This dissertation entitled **Feature Enhancement of an E-learning Platform for Automatic Evaluation of Programming Assignments** by **Swaresh Uttam Sankpal** is approved for the degree of **Master of Technology** in **Computer Science and Engineering** from **IIT Bombay**.

Prof. Varsha Apte
CSE Dept, IIT Bombay
Project Guide



Prof. Kameswari Chebrolu
CSE Dept, IIT Bombay
Examiner

Prof. Bhaskaran Raman
CSE Dept, IIT Bombay
Examiner



Chairperson

## Declaration

I declare that this written submission represents my ideas in my own words and where others' ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

S. U. Sankpal

(Signature)

Swaresh Sankpal

(Name of the student)

153050085

(Roll No.)

Date: 29/6/2017

# Acknowledgment

I have this opportunity to express my deep sense of gratitude and special thanks to my thesis guide Prof. Varsha Apte for her constant guidance and motivation throughout the course of this project.

I would also like to thank my colleagues, for the enthusiasm they have shown in sharing their knowledge with me. I also thank my family members for their constant support and faith in my work.

**Swaresh Uttam Sankpal**
M.Tech.-2, CSE
IIT Bombay

# Abstract

The importance of programming has been noticed around the globe and hence all universities have included programming in their curriculum. Introductory programming course is made compulsory for all fields of engineering colleges nowadays. These programming courses generally also have a lab based programming course associated with it. In these labs, students have to solve programming questions related to topics that are covered in lectures. These student submissions have to be evaluated by TAs(Teaching assistants) and feedback has to be given as a part of evaluation process. This evaluation work was initially carried out manually by TAs i.e by downloading the student submissions and running test cases provided by instructor against the program to check the correctness. This evaluation technique had limitations like unfair grading, human errors, high amount of human efforts with limited number of TAs. Offline scripts were introduced later to automate this evaluation procedure, which decreased the human efforts at some extent but had few limitations like the time taken completing the whole evaluation procedure was too high. To tackle all the problems listed above, EvalPro was developed by IIT Bombay in 2014. EvalPro has been used since 7 times in a span of two years. It has highly reduced the human effort required for evaluation of programming assignments of such a big class and hence has been preferred by many professors. EvalPro has also successfully conducted RA exam twice (2016 & 2017). EvalPro has not only successfully conducted introductory programming course labs but has also conducted the advance programming courses labs like Data structures where the input files would be of size around 10-50 MB and also there would be hundreds of input files. This was the rigorous test for EvalPro and it performed pretty well. A lot of issues were reported as well as observed during this period of deployment and it has been developer's constant efforts to fix these issues and add new features to make EvalPro a better tool.

# Contents

# List of Figures

# Chapter 1

# Introduction

Programming has become very important part of higher education nowadays and hence all universities have made it compulsory to all the students irrespective of their majors. Currently, our capabilities can massively augmented by machines, which can be only communicated by means of programs. Hence learning programming is important. Also, learning to program is important because it develops student's analytical and problem solving abilities. By designing programs, we learn many skills like Critical reading, Analytical thinking and Creative synthesis which are important for all professions . Teaching introductory programming courses is very difficult since all students have different skill set. Also, some students have no exposure to programming as well as computers. Hence, its very necessary to take all the above mentioned factors into consideration while designing problems so that both naive and good programmers learns something new every lab.

CS101 is an introductory programming course at IIT Bombay. This course has a lab every week and each lab would have around 4 problems on average. The course has around 500 students registered on average, which means around 2000 submissions are made per week. Assessment of such large number of programs is very tedious and time consuming job. Initially, this evaluation work was carried out by Teaching Assistants(TAs) by downloading student's submissions and running instructor provided test cases on it. For small sized classes, this evaluation procedure works fine but for courses like CS101, it becomes very difficult to assess and give timely feedback with limited number of TAs. Students often think their program is correct if they don't get any errors and runs correctly on some specific input. This is not the case very often, since it may happen that students program may fails on some other corner test cases that they haven't thought of. In this scenario auto assessment plays a vital role in evaluating and thus helping the learn-

ing process. Timely feedback of an program is also very important factor, since it helps students to debug their programs before deadline is over and re-submit the program for evaluation. To solve above mentioned issues, EvalPro was developed by IIT Bombay in 2014. Many other similar Automated Assessment tools exist like Webcat, BOSS, Autograder, EvalPro, Prutor etc.
EvalPro is a web-based automated programming assignment evaluator that gives immediate feedback based on results obtained by executing the student's code against instructor provided test cases. EvalPro is developed using Django MVC framework. It is integrated into e-learning platform named Bodhitree. EvalPro provides a standalone programming platform i.e student has to do programming offline and upload the code on bodhitree web application to evaluate the it. Students learn a lot about systems along with programming when coding is done offline. A student's submission is ran against instructor provided test cases to check the correctness of the program in EvalPro. From instructor's perspective there are lot of features like modular evaluation, manual feedback provisioning, plagiarism detection, crib management system, multiple language support, exam feature, obtain statistics of class etc which makes a job of instructor as well as TAs a lot simple. From student's perspective also, there are a lot of features like advance correctness checker, test case level partial marking, immediate feedback etc which makes evaluation scheme a little flexible and responsive. EvalPro also has crib management system which allows to raise a crib if there is any. Beside these advantages of EvalPro, there were many limitation to its functionality which were needed to be addressed to make it a better tool. Such limitations were observed in live deployment of EvalPro in courses like CS101, data structures and RA exam 2015  2016. We have addressed such limitations and implemented solution for each one of them to make EvalPro a much better tool.

In motivation chapter, we will discuss about the shortcomings leading to development of features that we have implemented. Details of similar auto evaluation tools and comparison of such tools with Evalpro will be discussed in related system chapter. Next chapter is about EvalPro, its 3-level assignment structure, its main features which makes it an advance auto evaluator and at last the desired features that lag in EvalPro's functionality. Design  Implementation chapter include complete details i.e current approach, its limitations, our solution and its implementation of all the feature that we added to EvalPro's functionality. Final chapter of this report discuss about the future work that can be done in EvalPro to make it the best tool for automating the evaluation process of programming assignments

in academic institutions like IIT Bombay.

## 1.1 Contribution to this thesis

EvalPro has a lot of features that makes it a very good tool for automating evaluation of programming assignment in both introductory programming courses like CS101 and also advances courses like Data structures. We have added many features to EvalPro's functionality to make it a better tool. These features include

- Execution and display of graphics programs through web application
- Feature to copy assignments from one course to another
- Backup assignments in trash before deleting it completely
- Execution time calculation of programs
- Feature to enable feedback based manual grading of an assignment
- Email support in Crib management system
- Feature that maps students and TA's together
- Feature to save intermediate student submissions
- Added In-Browser Code Editor to EvalPro
- Feature to compare output files visually
- Feature to calculate indentation percentage of a program
- Download all files related to an assignment/ course in a single attempt
- Feature to publish/ unpublish solution code on demand
- Feature to extend EvalPro's functionality for Non-Programming assignments

# Chapter 2

# Motivation

Lot of university courses include programming nowadays. Evaluation of programming assignments in such courses with large class size with limited number of TAs is really a challenging task. Since these lab courses have credits assigned, proper evaluation procedure is must. Introduction to programming course in IIT Bombay had a class size of around 500 students. This course had a lab every week, which included on average 4 programming questions. So, on average there would be 2000 submission on average to check every week by around 50 TAs. EvalPro was developed to automate this task and it performed well when it was first used in Autumn 2015. We will discuss further the various techniques used before to achieve the goal of programming assignment evaluation and how EvalPro overcame the limitations of those techniques.

## 2.1 Manual Evaluation

Manual Evaluation Procedure -

1. Download the submission
2. Untar the submission
3. Download the test cases provided by instructor
4. Compile student's submission
5. Run the executable against input of test case and store the output into a temporary output file
6. Compare the actual output against expected output using diff checker

7. Repeat step 5 and step 6 for each test case

8. Store the results into a spreadsheet for each submission

This evaluation procedure is followed by TAs to evaluate submission of one student. It is very time consuming job and also there is high possibility of human errors. In step 6, TA's usually use diff checker provided by linux. There is high possibility that students may print some extra statements with desired output. In such case, diff checker will fail, even when the output is correct. Such errors can be taken care for small test cases by visual comparison, but for larger test cases it is almost impossible to check if expected and actual output is correct. Also, there is no provision for students to check if their program runs on test cases that will be considered for evaluation. Students might think if their program doesn't have any compilation errors and runs on simpler test cases, his/her program is correct which might not be always true. Since, this procedure is time consuming, error prone and there is no immediate feedback, offline script based evaluation was introduced and implemented in many courses.

## 2.2 Script based offline evaluation

This is the evaluation method followed by EvalPro. EvalPro overcome all the limitation of above mentioned evaluation methods. A lot of time time and resourses are saved by the automation that EvalPro does.

Procedure for script based offline evaluation -

1. Download all submissions

2. Run script (only once)

    - Untar submission
    - Compile submission
    - Run executable against input of each test case and compare the actual output with expected output
    - Store the marks into csv file

There are many limitations to this type of evaluation also. For example, there is a tight constraint on name of the submission. There is no immediate feedback is this type of script based offline evaluation. Once script is run, then only feedback

can be given to student that if their program is completely correct or not. After first evaluation, students have to make changes in their programs as per feedback and resubmit the assignment. Lot of time is wasted in such type of evaluation, both of students as well as TAs. This can be overcome by building a system that gives immediate feedback to students about their program, so that they can change it immediately and resubmit. Also if student prints extra print statements, he/she gets zero marks according the script based evaluation which uses traditional diff checker internally. Also if some student submits malicious code, it will affect the machine on which evaluation is carried on. Hence, it is very insecure and time consuming evaluation method.

## 2.3 Immediate feedback based online evaluation

This is the evaluation method followed by EvalPro. EvalPro overcomes all the limitations listed in above evaluation methods. The student's submission in run in a sandbox, so that if some student submits a malicious programs it does not affect the host system. This is called sandboxing. EvalPro has an advance correctness checker which is advanced version of normal diff checker. It is very helpful feature specially for courses like CS101 where students normally put print statements to debug their program and end up getting zero marks because of normal diff checker. EvalPro's evaluation method is not completely binary. It has a feature that gives partial marks for partial correctness, which is very helpful. Also, EvalPro provides immediate feedback about the correctness of the program which enable students to debug their program and resubmit. This scheme of evaluation saves a lot of time compared to script based offline evaluation where submission has to be done in two phases, followed by two times evaluation.

## 2.4 Shortcomings leading into features I have implemented

Recently many universities have started giving graphics assignments in introductory programming courses. Graphics program help build the logical thinking needed for programming and also the interactiveness helps the learning process. CS101 in IIT Bombay also includes such graphics assignments in initial weeks of the lab course. Evaluation of such programs is a tedious work. TA's have to down-

load the student's submission, run it on a local machine and check the output. This procedure has to be repeated for each student assigned to TA. This evaluation procedure consumes a lot of time and there is high probability of human errors. When a graphics program is evaluated in EvalPro, the graphical output is available on server's display where EvalPro resides and not on client's display. We have implemented a feature that executes such program on server side and displays the graphical output on client side. We will discuss about this feature in chapter [5] in detail.

Execution time of a program is also very important parameter for evaluation in advance programming courses such as data structures. It also reflects the complexity of a program at some extent. Recently, Data structures lab course at IIT Bombay had an assignment where instructor wanted to calculate the execution time of a student's submission while assigning grades. Execution time calculation of this lab's submissions was done manually by downloading student's submission and checking the run time of program by using time function provided by linux. This is a also a time taking evaluation method. We have implemented a feature in EvalPro that does this calculation within the system

EvalPro has an assignment delete option, which deletes all the data and submissions related to an assignment. If someone by mistake clicks it, complete assignment is vanished. One such case happened in CS101 where an assignment was deleted by mistake, and complete data was lost. Trash is the best solution to avoid such scenarios.

An instructor might want to reuse the assignment created earlier in previous course by some other instructor in the same course or one of the other courses he owns . To allow reuse of older assignments, we have added a repository feature in EvalPro that allows instructor to save an assignment into repository and copy from repository for future purpose.

Manual feedback on lab assignments is a vital part of evaluation process since it helps students understand their mistakes and can help them avoid them in future. In CS101 taken at IIT Bombay in Autumn 2015, TAs use to evaluate the submissions and write the feedback in a google spreadsheet along with final marks i.e test case level marks given by EvalPro plus additional marks based on indentation, logic etc . This process requires a lot of management and there is high probability of human errors. Similarly, in CS101 offered at IIT Dharwad in Autumn 2016 this manual feedback was given to individual students by the means of e-mail which is again a time taking procedure.

There were many issues that students face were observed during the lab sessions like to check the correctness of program and to see the output format students use to download the test case one by one through bodhi tree. This process would use to take a long time if the number of test cases were more. This led to a requirement of a bulk download of test cases option which would download the test cases in a single click. Also there were other issues like student use to make last minute changes into their program and would add a small bug that would get them zero marks despite of their partially correct code that was submitted before. It was also observed that student even saved the different versions of their code in their local machine, so that at least they get partial marks if they add any last minute bugs into the code. So, it was very necessary that EvalPro saves all the submissions that students make till the deadline and let them choose the one which is to be considered for evaluation. In current version of EvalPro, it is very hard to compare expected output and actual output of the program especially when there are spacing issues or if the there are very small errors in fairly large output files. Students use to waste a lot of time in finding out the problem in their code since it was very difficult to see the minor differences in two outputs. To tackle this issue, visual aids showing the difference between expected output and actual output of the program would really help students to debug their program quickly.

There were issues reported by instructor as well as observed by TAs from instructor's point of view, which were needed to be fixed. Crib management system was used well enough in all the deployment of EvalPro in last two years. The TAs reported that they were unable to know about the cribs that students post on EvalPro unless they login and check the dashboard. It is very unlikely for TAs to constantly check the dashboard for cribs. To solve this problem email based notifications were sent to TAs for every crib that student makes on EvalPro and also a notification mail was sent to students when TAs respond to the cribs submitted. Another issues that was observed was that there is no mapping of student to TAs in EvalPro which makes it very difficult for instructor as well as TAs to distribute the evaluation work and cribs within themselves. To resolve this issue, a feature was added to EvalPro's functionality that would handle the mapping between students and TAs and make the evaluation and cribs process work efficiently.

Many of the auto evaluation tools have an in-browser editor embedded into it. These editors have a lot of advantages like it reduces the time overhead that student makes in case of upload based submissions, also it will help instructors change the student's code buggy code inside editor and check if it works and evaluate based

on that. There are many more such advantages of in-browser editor that make it a vital part of auto evaluator and it was the time that such editor should be added to EvalPro.

# Chapter 3

# Related Systems

Many tools have been developed in different universities for automating the evaluation procedure of programming assignments. We will discuss about them in this chapter.

## 3.1 Boss

Boss [1] is an online submission and assessment system developed at University of Warwick.

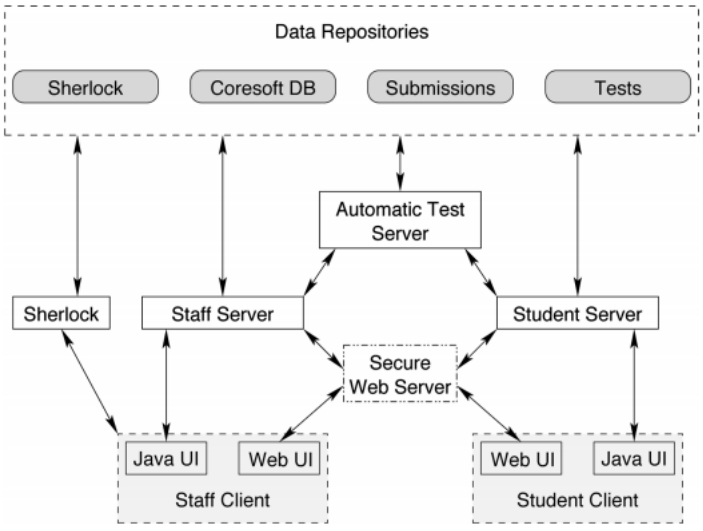



Figure 3.1: Architecture of BOSS, source:[1]

Main feature of BOSS-

- Plagiarism Checker :It has plagiarism checker called Sherlock which checks plagiarism before evaluating the program.
- Evaluation :BOSS validated output using scripts for C,C++ programs and Junit for java programs.

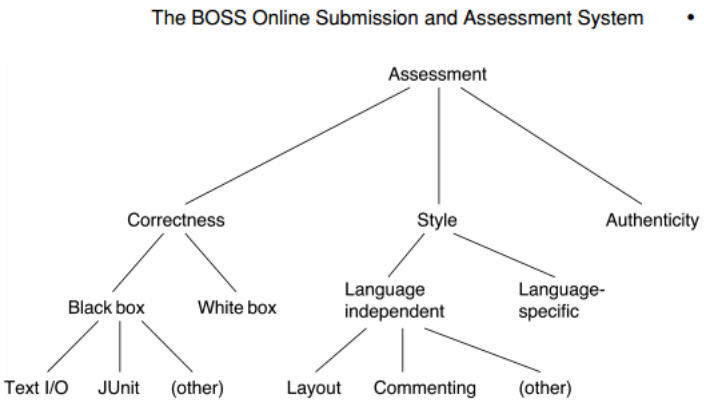



Figure 3.2: Online submission and assessment system of BOSS, source:[1]

- Dividing functionality between separate servers has provided technical and organizational benefits.

Student server and staff server has limited access to reduce security risks. Feedback is provided manually. It was observed that it took 1 to 2 hours to make test cases, output files and few seconds to evaluate one student's submission.This solution is very efficient that the original manual assessment.

## 3.2 Prutor

Prutor is a Web-based IDE. It has a plugin based architecture. This type of architecture is very helpful as any external tool can be easily integrated with Prutor. Prutor can work with any compiler/interpreter which can be invoked on command line. Tested for gcc, clang, Python, Prolog etc. It uses external tools for all functionalities except feedback generation. It uses docker for scalability purpose.

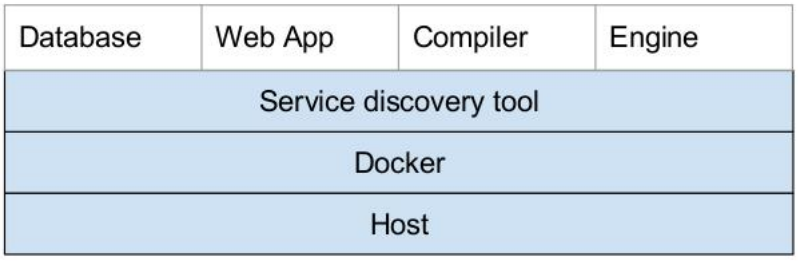



Figure 3.3: Architecture of Prutor

Main features of Prutor :

- Plug-in based architecture : This type of architecture helps a lot to expand the tool, since any feature/tool implemented already can be integrated with prutor very easily.
- Better support for TurtleSim programs : Grading a turtlesim program in evalpro is tedious and time consuming. But, in Prutor it is very easy. Evaluation takes very less time for turtlesim programs in Prutor than evalpro.

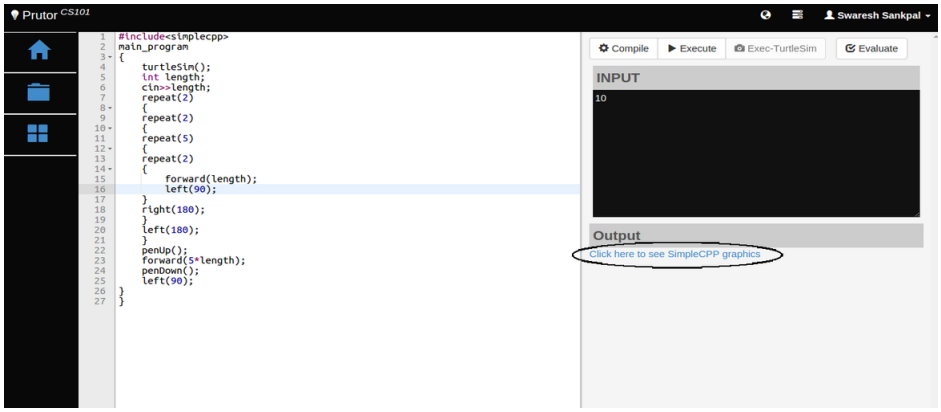



Figure 3.4: Executing a turtlesim program

- Upgrade/Downgrade Problems : Student can change the difficulty of problem by using upgrade or downgrade option.
- Automated Test Case Generation : Designing test cases manually consumes the most time in problem generation. To automate test case generation Prutor uses KLEE software. KLEE generates a set of test cases depending on the model solution provided.

- Saving history of a program. Prutor saves the versions of students code after every save that student does. This helps students to track back to some previous working version and also helps faculty to observe the approach that students follow to solve a particular assignment.

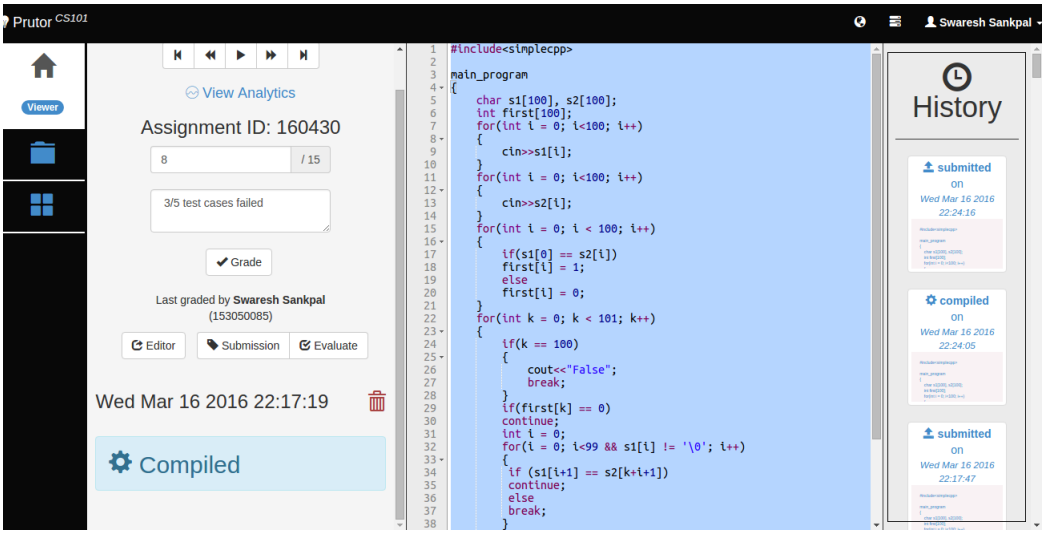



Figure 3.5: History of a program.

- Syntactic Feedback : An inline error message in the code with much more simpler terminology is generated so that the student is able to understand. Future work is to generate errors messages in native language

- Semantic Feedback : This requires large number of model solutions. Difference is calculated between every model solution and student's code. Feedback is given according to model solution with miniumum distance with student's submission.

- Ace Editor : Prutor used Ace as its editor. Ace is an embeddable code editor written in JavaScript. It matches the features and performance of native editors such as Sublime, Vim and TextMate. It can be easily embedded in any web page and JavaScript application.

## 3.3 Codeboard

Codeboard [4] is a Web-based IDE developed to teach programming. It is an open source and available under the MIT license. Integration of Codeboard with

MOOCs is very easy, hence is used very often in various MOOC courses.

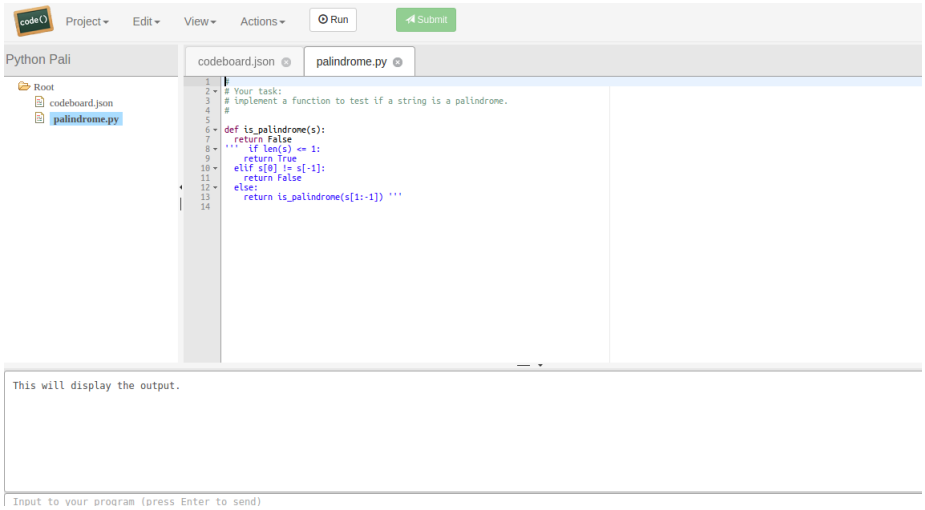



Figure 3.6: Web-Interface of Codeboard

A project in Codeboard is made up of components Project name, Project description, Programming language, Accessibility (public or private project), List of project owners, List of project users (only used by private projects), Allowing submissions, LTI settings.
Every project has a pecial configuration file named "codeboard.json" which defines for each programming language how the project gets compiled and/or executed.

- Security : Codeboard sandboxes all compilations and executions of projects.Programs terminates if following limits are exceeded: total CPU time used by the project: $>$ 12 seconds
  total session time for executing a projet: $>$ 15 minutes
  number of threads created by a project: $>$ 70 (value is approximative and may slightly vary)
  number of output characters created by a project: $>$ 15,000

- Evaluation : Two types of grading is supported currently-

  – Automatic grading using a result string. Every project in Codeboard can implement automatic grading by simply printing a special string as the last output when the project is executed. This string must be of the form:

<!–@test=the_grade_value;num_test_passed;num_test_failed;–>

On submission, Codeboard compiles and executes the submitted program and checks if the program output ends with a string satisfying the result string format.

- Automatic grading using unit tests.Codeboard supports unit tests for following prog. languages-
  - Java-JUnit
  - Haskell-HSpec
  - Python-UnitTest

  On submission, Codeboard will run all unit tests available in test folder as specified in config file and grade based on the number of passing and failing test cases in this folder.

- LMS integration : Codeboard can be integrated into any platform that supports LTI. Codeboard supports LTI(Learning Tools Interoperability) protocol which allows educational platforms to securely integrate with externally hosted tools(codeboard). An educational platform can send student information to Codeboard and, in return, Codeboard reports the grades of students back. Integration is very easy and hence it is used by various courses in educational platforms to teach programming like edX,Coursera etc.

## 3.4 INGInious

INGInious [2] was developed for auto-assessment at Universite catholique de Louvain, Belgium. It has a plug-in based architecture which also supports assessment of MOOCs with the help of plugins.

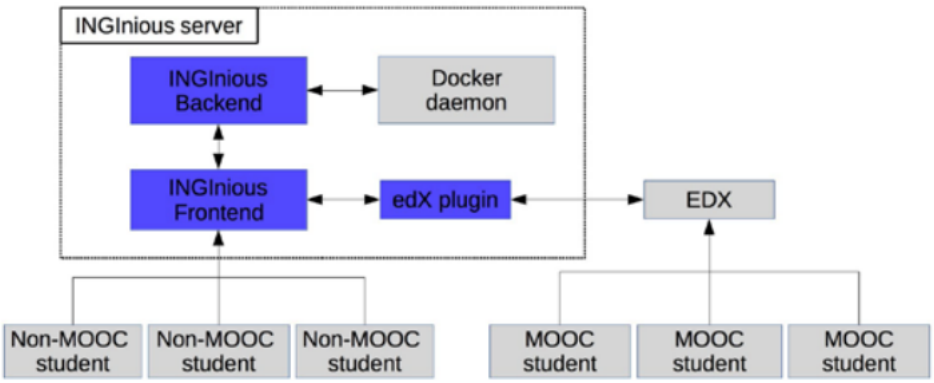



Figure 3.7: Architecture of INGInious, source:[2]

Architecture :

- Web-based interface(frontend) for non-MOOC students.
- Frontend also manages the database using MongoDB which store submissions and student statistics.
- Frontend also consists of plugin system which enables communication between edX and INGInious platform.
- The backend creates, manages and deletes containers, running the code made by the students.

Security and Scalability :

- Use of dockers instead of virtual machines since creating new environments(vm) is hard and there is a huge overhead.
- Docker provide safe and closed environments called containers.
- Containers are very light weight.
- Very less space is used to store the images.
- INGInious launches a container each time a submission is made, from the appropriate container image.

- INGInious adds some constraints, such as timeouts to the container to make sure that they end correctly.

Validation : The Autograder server used a virtual machine with 6 (virtual) CPUs and 12 GB of memory. MOOC exam was held in the month of Jan'15.During this period, around 600 students made 5976 requests to INGInious, with an average running time of 5.17 seconds. Memory usage never crossed 30 %.

## 3.5 ProgTest

ProgTest [3] is an environment developed for submission and evaluation of programming assignments based on testing activities at Sao Paulo University,Brazil.Testing plays a vital role in learning programming and hence is given importance in evaluating assignments in ProgTest.It uses Junit for unit testing and Jabuti for structural/style checking. Testing gives programmers a better understanding of his/her own code and helps learn programming skills better.ProgTest evaluates a student's submission on basis of:

- Instructor's code.
- Instructor's test cases.
- Student's code.
- Student's test cases.

A student has to submit program as well as test cases used by him to test his own program.ProgTest test grades a program on basis of 4 tests

- Instructor's test cases on Instructor's program.
- Student's test cases on Instructor's program.
- Instructor's test cases on Student's program.
- Student's test cases on Student's program.

Let's consider following example of factorial to understand the evaluation procedure of ProgTest:

```
public class Factorial {

    public static int factorial(int x) {
        if(x == 0 || x == 1)
            return 1;
        else
            return x*factorial(x-1);
    }
}
```

(a)

```
public class Factorial {

    public static int factorial(int x) {
        if(x == 1)
            return 1;
        else
            return x*factorial(x-1);
    }
}
```

(b)

```
public class Factorial {

    public static int factorial(int x) {
        if(x == 0 || x == 1)
            return 1;
        else if(x == 60)
            return 2;
        else
            return x*factorial(x-1);
    }

}
```

(c)

```
public class FactorialTest {

    @Test
    public void test1() {
        assertEquals(1, Factorial.factorial(0));
    }

    @Test
    public void test2() {
        assertEquals(1, Factorial.factorial(1));
    }

    @Test
    public void test3() {
        assertEquals(2, Factorial.factorial(2));
    }

    @Test
    public void test4() {
        assertEquals(120, Factorial.factorial(5));
    }
```

(a)

```
public class FactorialTest {

    @Test
    public void test1() {
        assertEquals(1, Factorial.factorial(0));
    }

    @Test
    public void test2() {
        assertEquals(1, Factorial.factorial(1));
    }

    @Test
    public void test3() {
        assertEquals(2, Factorial.factorial(60));
    }

    @Test
    public void test4() {
        assertEquals(120, Factorial.factorial(5));
    }
}
```

(b)

```
public class FactorialTest {

    @Test
    public void test1() {
        assertEquals(1, Factorial.factorial(0));
    }

    @Test
    public void test2() {
        assertEquals(1, Factorial.factorial(1));
    }

    @Test
    public void test3() {
        assertEquals(2, Factorial.factorial(60));
    }

    @Test
    public void test4() {
        assertEquals(120, Factorial.factorial(5));
    }
}
```

(c)

Figure 3.8: Example of program and test cases submitted by student, source:[3]

In above example:

- If student submits program (b) and test set (b), problem is detected when instructor's test cases are run on student's program since he has not handled input 0.
- It student submits program (c) and test set (c), there will not be any failure of instructor's test cases if all inputs are below 60.But, the problem will be detected when student's test cases is run on instructor's program.

| Assignment | Algorithm | Error | Test Set | $P_{Inst}$-$T_{Inst}$ | $P_{St\_i}$-$T_{St\_i}$ | $P_{Inst}$-$T_{St\_i}$ | $P_{St\_i}$-$T_{Inst}$ | Suggested Grade |
|---|---|---|---|---|---|---|---|---|
| 1 | Bubblesort | No | Strong | 100 | 100 | 100 | 100 | 10 |
| 2 | Quicksort | Small | Strong | 100 | 96.33 | 100 | 96.33 | 9.76 |
| 3 | Insertion Sort | No | Medium | 100 | 52.33 | 50.33 | 100 | 6.76 |
| 4 | Mergesort | Small | Medium | 100 | 46.67 | 50.33 | 92.67 | 6.32 |
| 5 | Heapsort | Big | Strong | 100 | 74.33 | 100 | 74.33 | 8.29 |
| 6 | Selection Sort | No | Weak | 100 | 39 | 39 | 100 | 5.93 |
| 7 | Shellsort | Big | Weak | 100 | 38.67 | 39 | 56.33 | 4.47 |
| 8 | Bubblesort | Small | With a Bug | 100 | 75.67 | 100 | 61 | 7.89 |

Figure 3.9: ProgTest Results, source:[3]

## 3.6 Comparison Table

| | EvalPro | Prutor | Codeboard | INGInious | BOSS |
|---|---|---|---|---|---|
| Web-based IDE | No | Yes | Yes | Yes | Yes |
| Standalone Programming | Yes | No | No | No | Yes |
| Output Comparison | Advanced | Normal | Normal | Normal | Normal |
| Simplecpp support | low | high | | | |
| Scalability | low | high | high | high | low |
| Integration with MOOCs | hard | hard | easy | easy | hard |
| Multiple courses support | Yes | No | Yes | Yes | No |
| TA workload | high | moderate | | | |
| Plug-in based architecture | No | Yes | | Yes | |
| Debugging | Hard | Easy | Moderate | Moderate | Moderate |

# Chapter 4

# EvalPro

**EVAL**uation of **PRO**gramming Assignment is a part of Bodhitree web application which aims to provide instant and automatic evaluation of programming assignment. It was developed by IIT Bomabay in 2014. It was developed in vision to automate the evaluation procedure of a data structure lab. Many features were added later to make it compatible with other programming course and was also tested in CS101, an introductory programming course at IIT Bombay. It is also being used in Data structures lab course at IIT Bombay and Introductory programming course at IIT Dharwad. In this chapter, first we will discuss about the 3-tier structure of lab problem/question in EvalPro followed by main features of EvalPro which makes it an advance tool for automation of programming assignments followed by the short description of desired features that EvalPro should posses. At the end, we will discuss the problems that were noticed in current deployment of EvalPro.

## 4.1 Assignment Structure

A typical programming question in EvalPro has a 3-tier hierarchical structure that aims to support modular evaluation which is one of the key features of EvalPro. It also support the test case level marking scheme of EvalPro. An assignment is divided into multiple section and each section is further divided into multiple test cases.

### 4.1.1 Assignment

Assignment part defines the Problem statement and other properties related to an assignment such as -

- Name of the assignment
- Type of lab : This field reflects whether the lab is a normal (ungraded) or an exam (graded).
- Deadlines : A typical assignment in EvalPro has two deadlines i.e soft deadline and hard deadline. The two deadlines were designed to support grading scheme where instructor wants to give extra time to students but also wants to reduce the credits given if they submit between soft and hard deadline.
- Publish time : The assignment get published at this time i.e students can view the assignment only after this publish time. This field was designed to allow instructor schedule the lab at any time he wants.
- Programming language : As of now C, C++, Java, Python and Simplecpp are supported.
- Assignment Document : This field is where instructor can upload the lab statement file, all test case files, images, pdf's related to an assignment. This part is useful to provide students all the test case files, because it takes a lot of time to click on individual test case and download input and output files associated with it. Any kind of resources required like slides, presentations can also be uploaded here.
- Helper code : Instructor can upload header files required for an assignment in this field
- Advance correctness checker : This is an advance feature of Evalpro that was designed to make the evaluation scheme a little flexible so that student gets non-zero marks even when he/she prints some extra statements other than desired output. Two other fields in assignment which comes along with this are importance of order and the error rate. These values should be set according to use case. This feature has not been implemented in any other similar auto evaluation tools studied.

- Bulk upload : This feature wad added recently to EvalPro since it was observed that creating assignment takes a lot of time when there are too many test cases. Test cases can be put in a hierarchical structure and upload here in compressed format. This feature automatically adds all sections and test cases present in uploaded tar to the assignment created. It reduces reasonable amount of manual work needed to create a complete assignment.

- Description : This is where instructor can define a problem statement of an assignment.

### 4.1.2 Section

This is the second level of a 3-tier structure of a lab question in EvalPro. This level of assignment was designed basically to group test cases based on use case. Typically for normal assignments i.e non-OOP assignments there should be 2 sections one is Practice and other is Evaluate. For an OOP assignment number of sections can be more than one depending upon the functionalities instructor wants to check. The properties of a section are as follow -

- Section Name

- Type of Section : A section can be either be of type practice or evaluate. The are two differences between practice and evaluate section, one is that the students will be able to see the the input and output files of only practice sections but not of evaluate section and second is that the student's submission is only graded on basis of test cases under evaluate sections.

- Compiler command : There are two empty fields in this section, one is for setting compiler flags and other is for describing the file name that are supposed to be executed. In basic programs file name will be the file that student's will submit. If additional file names are provided here, they must be uploaded in additional source files part of section.

- Additional source files : Additional source file if any, should be uploaded here.

- Makefile

- Bulk upload : Similar to bulk upload option in assignment

- Description

- Test case level partial marking : This feature is a part of advance correctness checker mentioned above in assignment part. This feature provides student's submission partial marks when he/she gets a partially correct answer.

### 4.1.3 Test case

This is the third level of the 3-tier hierarchical structure of a lab question. Each section mentioned is group of the test cases. Each Test case is associated with following fields -

- Test case name
- Marks
- Input File : Input file corresponding to test case should be uploaded here.
- Output File : Output file corresponding to test case should be uploaded here.
- Description

## 4.2 Main Features of EvalPro

### 4.2.1 Modular Evaluation

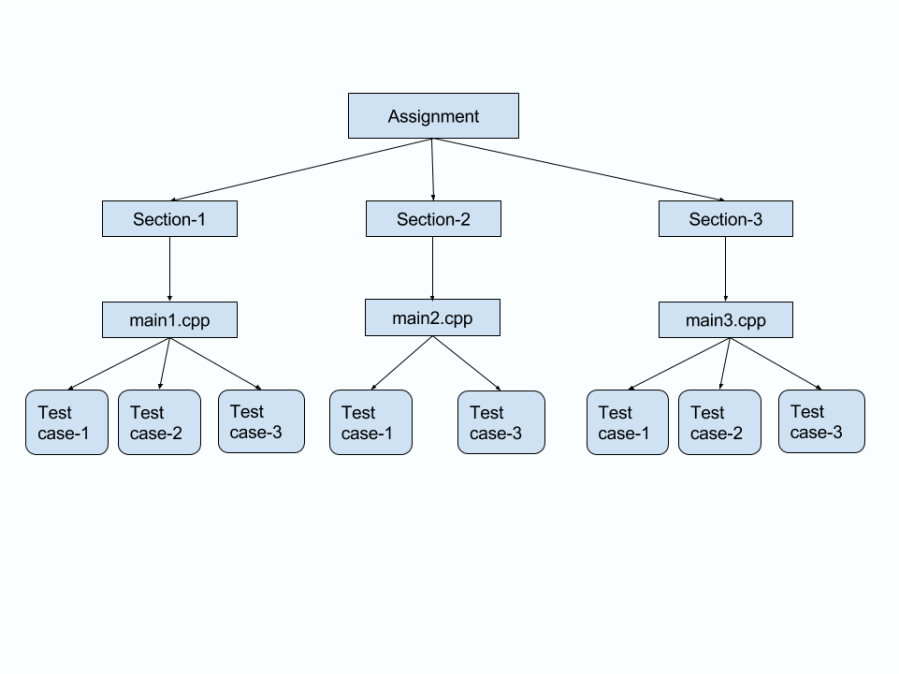



Figure 4.1: Modular evaluation in Evalpro

It is one of the main features of Evalpro. It is useful for programs with concepts like structs, classes etc. Basically, a program is divided into sections(functions). For every section we have separate main files.For each executables of these main files,there are many test cases associated. It is very useful in evaluation when some functions are working while some are not. For example say some function goes into infinite loop or segmentation fault occurs, this will affect working of succeeding function called which are even correctly implemented. This way, we can determine how many functions are correctly implemented independently. This unique feature is not found in any other auto assessment tools.

### 4.2.2 Advance correctness checker

Even after clearly specifying the output format, many students make formatting error. For example, student is asked to find prime factors..
They format output to :

Prime factor is : x

Answer is: y

This make auto-evaluation a little difficult despite of correct answer provided by student. Output comparison is made intelligent in Evalpro to handle such formatting problems. Following figure shows the approach followed :

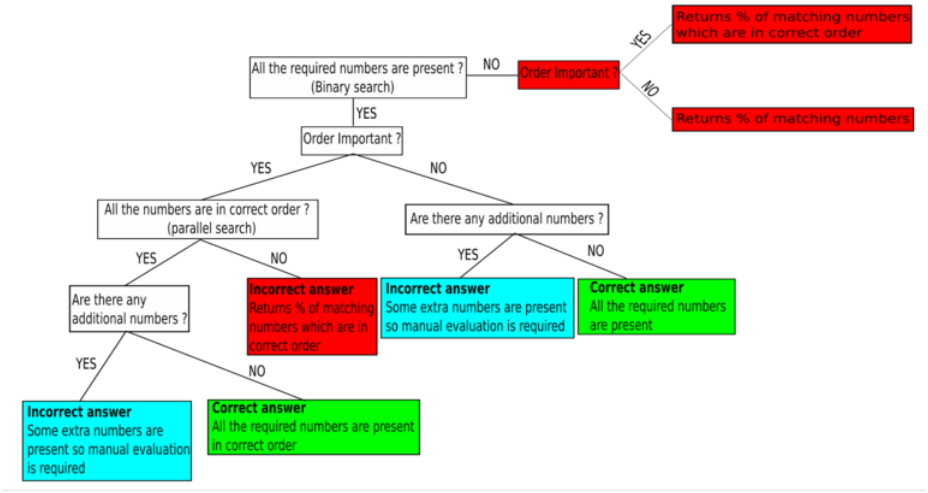



Figure 4.2: Output comparison and partial marking in EvalPro, source: nirav's MTP report

### 4.2.3 Test case level partial marking

Grading of an assignment in many auto-assessment tools is binary i.e test case is given either 1 grade or 0 based on the diff checker that compares expected output and student's output.This type of evaluation is too strict. It student misses 1 or 2 numbers out of 10, he gets 0 grade for the particular test case. In Evalpro, partial marking can be given on test case level as shown in above above figure.

### 4.2.4 Automated test case generator

In EvalPro , student's code is tested on different inputs. It is difficult for instructor to generate all the variations of inputs for the given assignment. To tackle this problem automated test case generator was added to EvalPro. With help of this feature, instructor create array of numbers, array of array, array matrix etc. This is helps a lot when instructor wants to to create many test cases.

### 4.2.5 Crib management system

Cribs are also a important part of good evaluation scheme. EvalPro has a crib management system that allows students to make a crib for any particular assignment. Instructor can view the cribs made by students in dashboard section. Instructor has privilege to assign the cribs among the TA's. Any of the TA's can add reply back to cribs by adding comments to it. Every crib is tagged with either Resolved or Unresolved tag. Once the crib of student is resolved completely by TA's, he/she can change the status of crib from unresolved to resolved.

### 4.2.6 Sand boxing

EvalPro runs a student's submission inside a sandbox to protect the host machine from any kind of malicious code. This sandbox has some configuration parameter which are set to defaults if not configured. These parameters can be configured from the link present on assignment page. The configuration parameters of this sandbox are -

- CPU Time :
  It indicates the total amount of time for which program can use CPU. It does not include the time which is spent in waiting for input and output
- Clock Time :
  It indicates the total amount of time for which program can run on the server. It consist of CPU execution time and waiting time for input and output.
- Memory Limit :
  It indicates the maximum amount of memory that program can use.
- Stack Size Limit :
  It indicates the maximum size of stack that program can use
- .Maximum Child process :
  It indicates the maximum number of process that program can fork.
- Maximum number of files open :
  It indicates the maximum number of files that program can open at a time
- Maximum size of file :
  It indicates the maximum size of file that program can write.

### 4.2.7 Exam Feature

Examination is a vital part of any course and needs to be more secure than the normal labs. In EvalPro, an assignment can be of two types either lab or exam. EvalPro puts some tight constraints on assignments tagged as exam, since they have to handled very carefully. Instructor can allow only particular machines to have access to exam by providing IP addressess of those machines. Duration per student is maintained. Extra time can be allocated if someone comes late for exam. If some student changes the machine within the duration of exam, instructor has to approve this action.

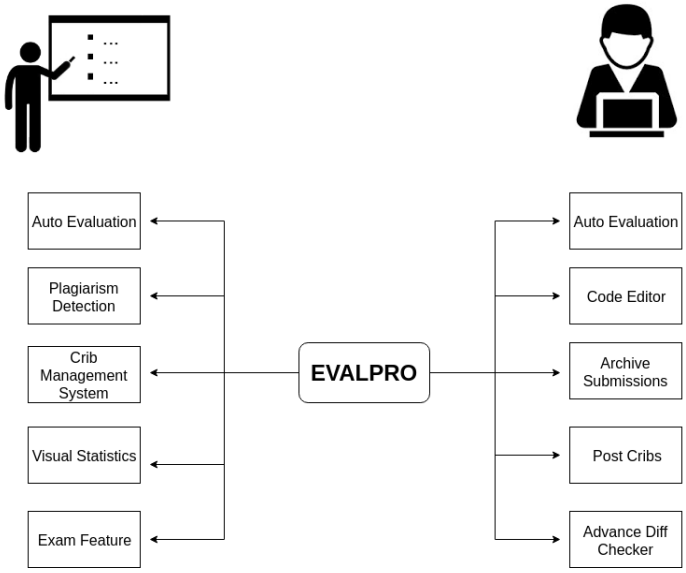



Figure 4.3: Main Features of EvalPro

## 4.3 Desirable Features in EvalPro

- Time complexity based evaluation
  Time complexity is a very big concern in advance computer science courses like Data Structures. For example, instructors asks students to implement merge sort in a lab, but some students since didn't knew or didn't want to implement merge sort and implemented bubble sorted instead. Execution time feature can be extended or some new logic has to be implemented in EvalPro to make it do evaluation based on time complexity of a program similar to what hacker rank has implemented.

- Multiple language support in single assignment
  As per current architecture of Evalpro, one assignment supports a single programming language and if we want to give students multiple programming language option we have to create an assignment for each programming language. A single assignment should support multiple programming languages.

## 4.4 Problems occurred in recent deployments of EvalPro

In 2015-2016, EvalPro was deployed in Data Structures lab course at IIT Bombay and CS101 at IIT Dharwad. There were some issues in EvalPro that came up during this period. Some of the issues were fixed by me. All issues are listed below -

- Multiple evaluate requests of same submission
  In data structures lab course, it was observed that students use to evaluate the same submission multiple times before the results of previous request of evaluate would arrive. This created a large number of redundant request that would affect the server's performance. We wrote a javascript that disables the evaluate button once first request is made. To evaluate again, students need to refresh the page or delete and re-upload the submission. This decreased the server load at some extent.

- Problem loading large input/output files in results page
  There was an assignment in data structures which had very large input/output files. When a submission is evaluated, a html page is shown which consists of input,expected output and actual output for every test case. Because of large input/output files which were of size around 20 MB to 30 MB, the results page was not able to load and would give nginx bad gateway error eventually. We solved the problem by hiding the input/output files for that particular lab and only showing the obtained marks.

- Plagiarism checker not working properly
  Plagiarism checker needs a perl file named moss, which does the authentication, sends the data to moss server for plagiarism detection and shows the results received. This feature of plagiarism was added in Autumn 2015, and the moss file exists only in the bodhitree version that was running CS101 in

that semester. This file has to be integrated with bodhitree through phabricator, so that the plagiarism feature is added to bodhitree permanently. We have done the setup required for plagiarism checking on both servers running data structures and CS101 of IIT Dharwad. We are also working on pushing this feature to phabricator.

- A student can see all the cribs
  It was observed in data structures lab that when students started putting cribs for assignments, they were able to see all the cribs associated with that assignments made by other students as well. This was fixed by only showing those cribs that are made by user who is logged in.

- Downgraded version of GCC compiler and Python
  The version of GCC compiler on bodhitree machine which hosted data structures courses was lower than the machines used in labs. There was a crib in one lab of data structure, where student complained that her code was working on lab mahcine but not on bodhitree. It was found out later that she has used one inbuilt function which is implemented in newer version of gcc but not in the older version which is on bodhitree. Also, the python version on bodhitree was very old, which raised error while using the plagiarism feature of EvalPro and suggested python version upgrade as a fix to this. Hence python and g++ versions of bodhitree machines were upgraded as a fix to above two problems.

- Need of execution time feature
  In data structures lab course at IIT Bombay, there were many assignments that had to be evaluated on the basis of execution time of student's code i.e efficiency of their code. There was no such feature in EvalPro at that time which would calculate the execution time of student's code and report it to instructor. The TAs use to download the student's program on their local computers and check the running time of the code and then award the marks. This evaluation process was time taking and had to be solved to make EvalPro compatible with advance courses like data structures.

In 2016-2017, EvalPro was deployed in Data Structures lab course at IIT Dharwad, IIT Goa, Project staff exam and RA selection test 2017. There were some issues in EvalPro that came up during this period. All issues are listed below -

- Need of Scoreboard
  RA selection test that was conducted in May 2017 used EvalPro for the programming test. There were total of 6 programming question and there were two programming language options provided - C and C++. So, 12 assignments were created in total for the selection exam. To shortlist the candidates on the basis of their scores in programming, it took around 60-90 min for a group of 15 people which is a lot of human efforts and there might be a high chance of human errors. To solve such problems, there should be a Scoreboard feature in EvalPro that keeps all assignment marks of all students at a single place.

- Need of hierarchy over assignments
  In RA selections test 2017, there were 12 assignments created for programming test. The assignment deadlines were set before the exam started, but exam started late than the expected time and the deadlines were required to be adjusted for all assignments. As per current structure of assignment, instructor had to change the deadlines of all 12 assignments one by one which is a tedious work. There should a hierarchy over assignments which would club them together and have common fields like deadline, publish time etc.

- Need of student to TA mapping
  As per current structure of assignment, all cribs related to an assignments gets assigned to a single TA. This process increases load on a single TA and needs to be distributed among all TAs. In such case, instructor had to email constantly TAs that who should solve which crib. This problem becomes critical in case where there are 500 students in a course. To tackle such problem, there should be student to TA mapping defined inside EvalPro. This mapping would also help in distributing the student evaluations among TAs.

- Need of email support in Cribs
  There is no notification provided to TA when a crib is raised by a student. TAs have to constantly check dashboard to see if any new cribs are made. Email support in cribs would really help the crib management system to improve.

- Total marks of an assignment
  In RA selection test it was observed that total marks for an assignment cannot be viewed anywhere. The marks distribution was announced separately by instructor for the exam. Displaying total marks would really help students to prioritize the assignments and solve them.

# Chapter 5

# Design & Implementation

EvalPro was developed to automate the process of programming assignment evaluation and has improved a lot in other aspects as well. It has been developer's constant efforts to find out the fields in which EvalPro lack and overcome those limitations. Following figure shows the features we have added to EvalPro and the areas in which EvalPro has improved.

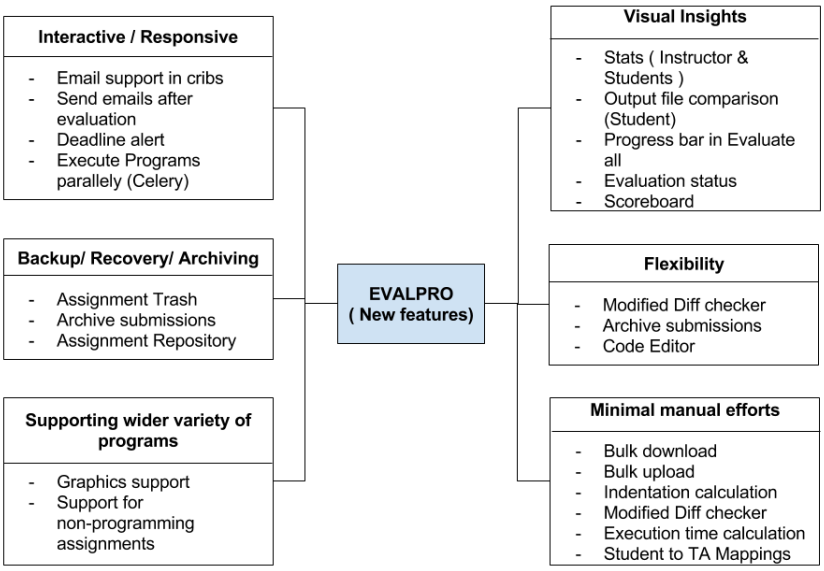



Figure 5.1: Feature Categorization

Above figure clearly shows that EvalPro is no longer only a auto evaluator, but a complete programming lab management tool since it offers much more functional-

ities than only an auto evaluator and can be easily used for programming lab course of 500-1000 students. Still there are few areas in which EvalPro can be improved which are listed in section Future work.

## 5.1 Execution and display of graphics programs through web application

Recently instructors in various universities teaching introductory programming course have started use of graphics programs in labs to teach programming. According to research, graphics programs helps the learning process required in programming. Graphics program helps build the logical thinking of students. CS101 offered at IIT Bombay in Autumn 2015 and Spring 2016, CS101 offered in IIT Dharwad in Autumn 2016, all these courses had assignments on simplecpp graphics for initial 3-4 weeks in lab. Evaluation of these assignments requires a lot of human efforts. We have added a feature to EvalPro which executes the graphics program on server machine and shows the graphical output on client machine. We will discuss about this feature in subsequent subsections in detail.

### 5.1.1 Current approach

The evaluation procedure of simplecpp graphics programs are currently done in following way -

- TAs download the student's graphics submission to their local machine
- TAs compile the student's program
- TAs run the executable on test cases given by instructor or their own custom input
- TAs compare the graphical output against the expected output given by instructor

### 5.1.2 Issues in current approach

- Time consuming : The current procedure mentioned above is a time consuming process. A TA in CS101 is responsible for grading on average 12 student's assignments. Each lab has on average 4 such programs and 3-4 different test

cases to be tested on each program. A TA has to evaluate around 50 such programs on 3-4 different test cases. So, totally a single TA has to execute graphical program around 150 times and check output visually on local machine for one week's lab.

- Manual errors : EvalPro puts a tight constraint on assignment file name for the purpose of its automatic evaluation procedure. When TA's download the student's submission, linux automatically renames the submission since the file name is unique for an assignment. TA's can either delete student's submission from current directory in linux machine after the evaluation of particular student or rename every submission corresponding to each student. There is high probability that errors will occur while grading the students.

### 5.1.3 Solution

Simplecpp graphics package developed by Prof. Abhiram Ranade is used for graphics based assignments in CS101 offered at IIT Bombay and IIT Dharwad. Simplecpp package uses x window programming internally to show graphical output on linux desktop display. The X Window System [5] (X11, or shortened to simply X, and sometimes informally X-Windows) is a windowing system for bitmap displays, common on UNIX-like computer operating systems. X provides the basic framework for a GUI environment: drawing and moving windows on the display device and interacting with a mouse and keyboard. X uses a client–server model in which X server communicates with various client programs. The server accepts requests for graphical output (windows) and sends back user input from keyboard and mouse. Usually the X-server and X-client application are on the same machine, but it also works when server and client are on different machines. This functionality of X window system can be used for the purpose of automatic execution of simplecpp graphics programs on EvalPro server and show the output on client's desktop i.e student's local machine.

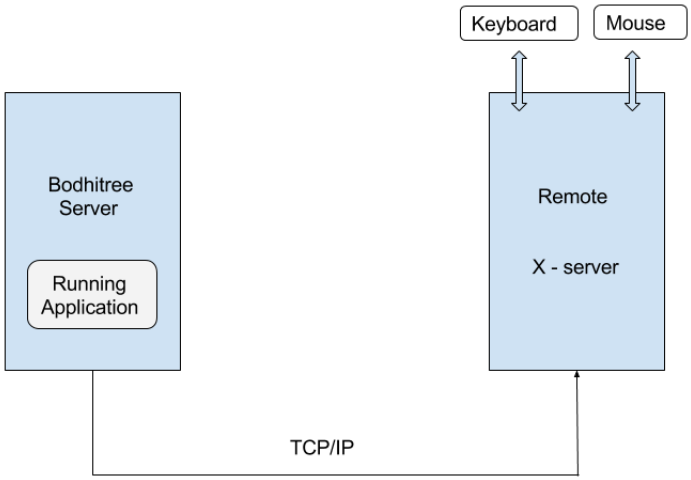



Figure 5.2: Remote display of graphical program running on bodhitree server

### 5.1.4 Implementation

#### Client side configuration

Usually the X-server and X-client application reside on the same machine. X-applications send data to X-server to display the graphical content. The display environment variable of host machine decides where the the output of the application will be rendered. This is set to default i.e 0.0, which is default display variable of host machine. If the output has to be displayed on remote machine, the display environment variable should be set as the IP:0.0 where IP is the IP address of the remote machine. Once application specifies the X-serves it is trying to connect with the help of this display environment variable, the request is authenticated at the remote machine whether the machine from which the data is coming is present in the access control list of its own system. Each system maintains its own access control list that specifies the list of computers that can access the X-server. We need to configure student's local machine, to allow server machine i.e bodhitree server to access its display. The configuration needed is as follow -

- X11 forwarding should be enabled on student's machine, which is by default enabled
- Setup X-server to listen tcp connections : X11 networking is disabled by default in many distributions. To allow tcp connections, add a line specifying "xserver-allow-tcp=true" to configuration file of display manager. Default display manager of the student's machine can be found by the output of "cat /etc/X11/default-display-manager". In latest machines, lightdm is the default display managers.
- To check whether the configuration is correct or not, enter "nmap localhost" and check if a display port is running on tcp port 6000
- To add bodhitree server machine in access control list of student's machine, type "xhost +IP" in terminal. IP mentioned here is the IP address of the bodhitree machine.

We have written a script to automate this configuration process which takes 3 arguments - on, off and connect. This script will be downloaded from bodhitree website. Steps that student's should follow to configure their own machine -

1. ./script on
   Script will enable X11 networking and restart the display manager
2. ./script connect
   Script will add bodhitree server to access control list of student's machine
3. ./script off
   Script will disable the X11 networking. This has to done once student/TA is done with the evaluation

Once X11 networking is enabled and bodhitree server is added to access control list, student's machine is ready to run graphical programs on EvalPro and watch the output on local machine.

**Server side implementation**

Every time EvalPro runs a graphical program, it has to describe internally the address of machine it wants the send the output. The execution of commands in EvalPro is done with the help of python subprocess. We have to run the executable of graphical program in subprocess environment where display variable is set to

the IP address and display number of the destination computer. For example, if graphical output has to be sent to IP address 10.129.1.1, the display variable should be set to 10.129.1.1:0.0 , where 0.0 is the deafault display number of the remote machine.

Also the evaluation process of normal programs in EvalPro doesn't fit into context of graphical programs. Normal programs here are the programs with non-graphical output. When a normal program is evaluated in evalpro, the testcase are run on the executable one by one and the results are stored in database and displayed on a new html page. In graphics based programs if same procedure is used, student/TA cannot know which output corresponds to which input. So we have changed the evaluation procedure of a graphical program, so that student/TA knows which output corresponds to which input. Once student/TA clicks on Practice/Evaluate button, he/she is redirected to html page that shows all test cases for that assignment. He/she can run his/her program on a particular test case and check the graphical output. Hence, when a student/TA evaluates his assignment, his/her program is run on all testcases one by one rather than section wise execution. Student/TA can also download the test case from this page to get to know about input. We have added a new checkbox in create assignment form, this checkbox has to be checked to mark assignment as an assignment with graphical output. Following snapshot shows the new page that enables student/TA to do testcase wise evaluation -

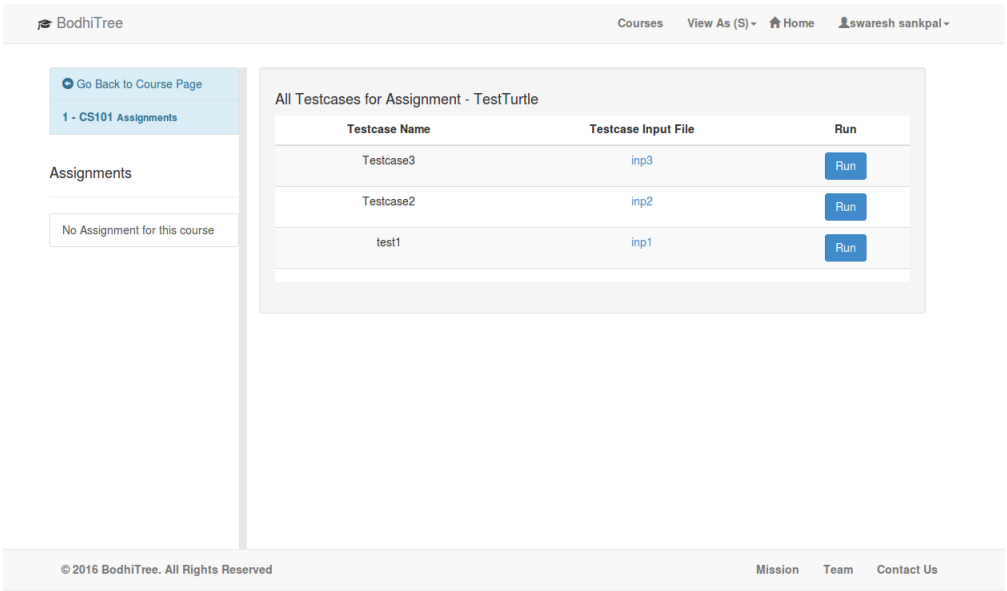



Figure 5.3: Testcase level execution of graphical program

The feature was tested on various graphical programs that were given as assignments in previous year CS101 course at IIT Bombay as well as current CS101 course at IIT Dharwad. The feature handled low end graphical programs very well, but some high end programs that were given in CS101 course at IIT Bombay in Autumn 2015 like ball game and photo electric programs ran comparatively slower. Some results are shown below in the form of screen shots -

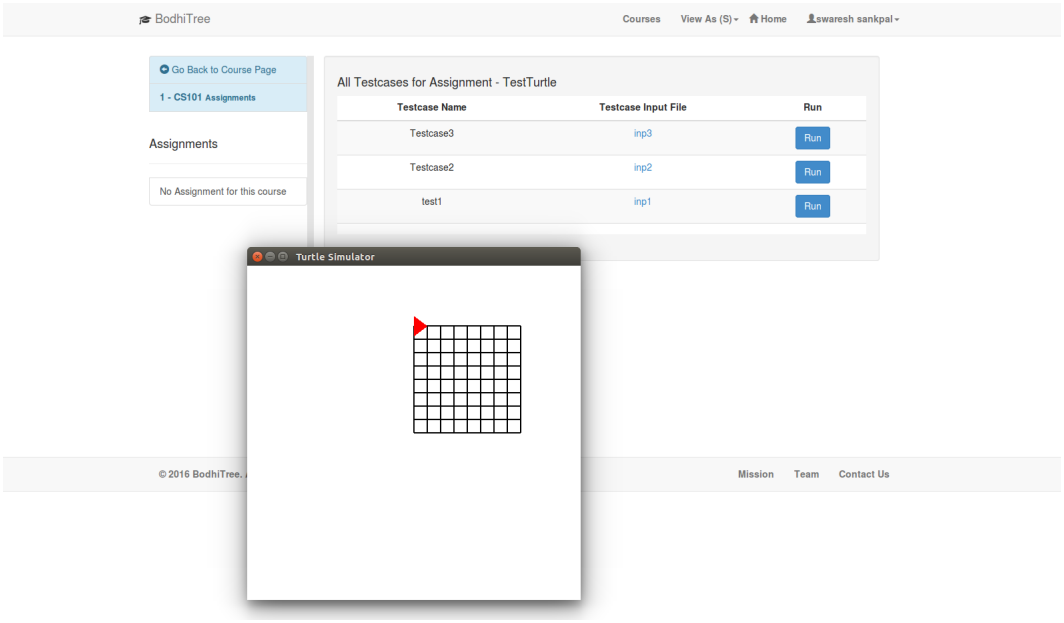



Figure 5.4: Output of a graphical assignment aiming at drawing grid

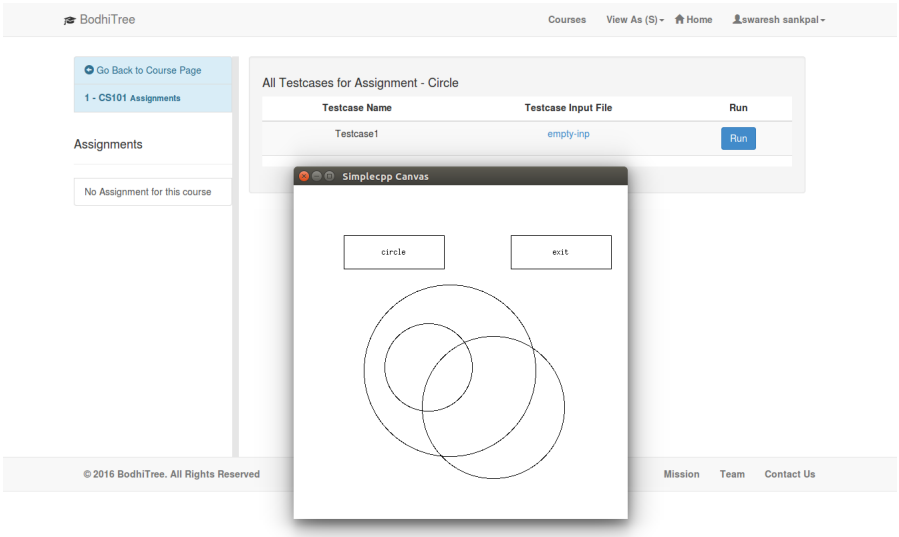



Figure 5.5: Graphics assignment with aim of drawing circles on user clicks

## 5.2 Execution time calculation of programs

Execution time of a program reflects how efficient is the code at some level. Execution time is as important as the output of the program. It also reflects the complexity of a program at very large inputs. Due to the high speed processors available nowadays, it takes nanoseconds to milliseconds of time to run a moderate sized program. Hence, we can't determine the complexity of a program by executing it against smaller sized inputs. Recently, a assignment in Data Structures lab course at IIT Bombay had this need to calculate running time of student's submission. This functionality is not offered by EvalPro currently. We have implemented a feature that gives execution time of a program for each test case and also the total time taken by student's submission for complete assignment. Instructor can also sort student's submission on basis of execution time.

### 5.2.1 Limitation

There is no feature implemented in EvalPro currently that determines the execution time of a program.

### 5.2.2 Solution

EvalPro uses safeexec (safe execution environment) [6] as a sandbox to prevent the host system from any kind of malicious code that might crash the system. Safeexec is an opensource available under the MIT License. Safeexec provides a sandbox to run programs inside a safe environments. Safeexec allows instructor to set limits on system resources like cpu time, clock time, stack size, number of open files, memory size etc. We can extract the execution time from safeexec code. The challenge here is to access that execution time from safeexec code into Django code and store it in a database. To calculate the execution time in microseconds, following formula is used -

execution time = ru_stime.tv_sec + ru_utime.tv_sec (in seconds)
ru_stime.tv_usec + ru_utime.tv_usec (in micro seconds)

where ru_utime = amount of user time process has used where user time is the CPU time spent executing the user program rather than in kernel system calls

ru_stime = amount of system time process has used, where system time is CPU time spent on system call execution

### 5.2.3 Implementation

We print the above calculated time in safeexec code using printstats function. Safeexec is executed once for every execution of a program against all test cases available in assignment. Hence, we can store the execution time of a program against each test case. Printstats prints the output in the stderr. We have written a regular expression matcher which runs against the stderr of the safexec and stores the extracted execution time in database. EvalPro shows the results table in a new html page, after execution of all test cases is done. We are displaying the execution that program took for each test case along with marks obtained. Also, the total execution time of assignment is calculated by adding the execution time of all test cases and displayed on AllSubmission page.

### 5.2.4 Results

Data structures lab at IIT Bombay had an assignment related to linked list data structure. Instructor wanted students to print the execution of program against the two large test cases that they have provided. There was no option for TA's other than downloading the submissions and manually running it on local machine and validate the execution time. I created the same assignment on development server in my local machine. I downloaded 8 submission of that week's lab and tested the implemented execution time feature on that submissions. Execution was done only for two large test cases provided in lab which were of 20 MB and 4 MB respectively. Following are the results obtained -

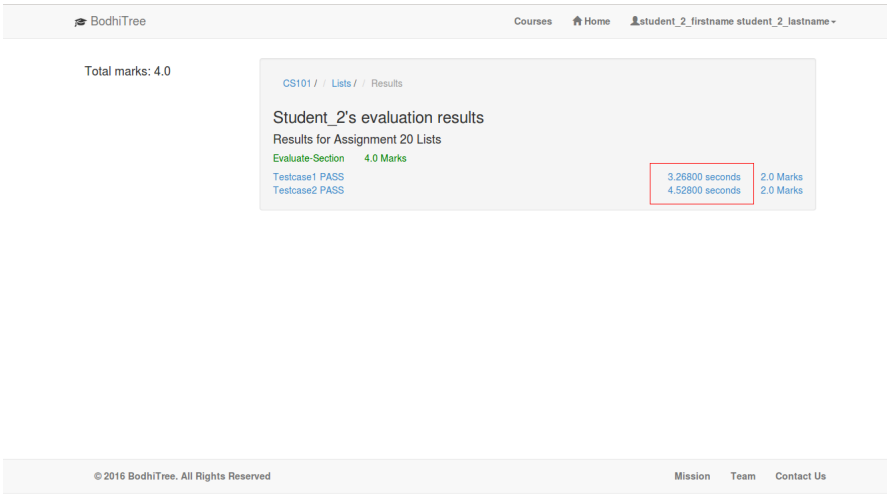



Figure 5.6: Test case wise execution time of a program

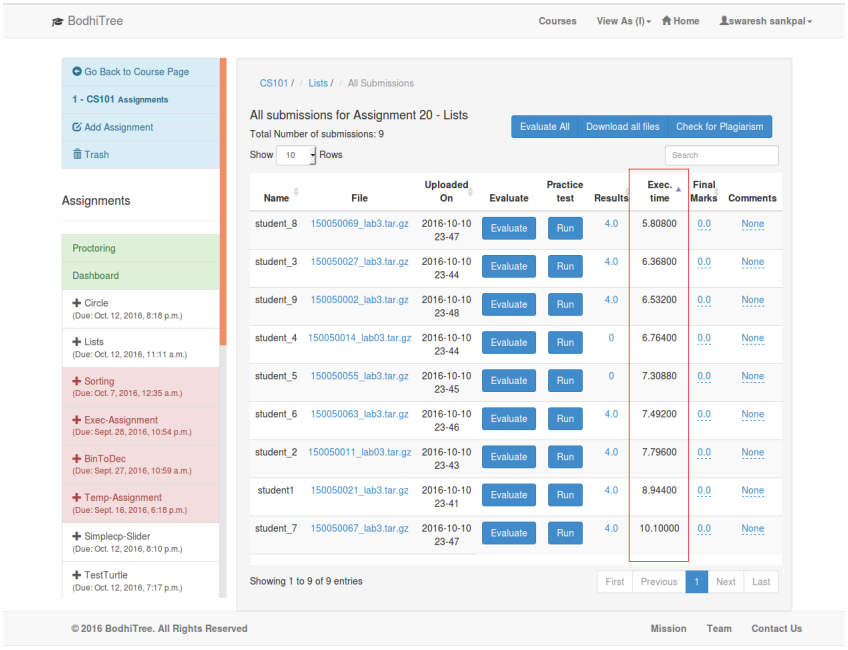



Figure 5.7: Total execution time of a program (Sorted in ascending order of execution time)

To validate the data, I ran the same set of programs against same set of test cases on system and calculated execution time using time function in linux. The difference between system time and EvalPro time was plotted using gnuplot. Following are the obtained graphs -

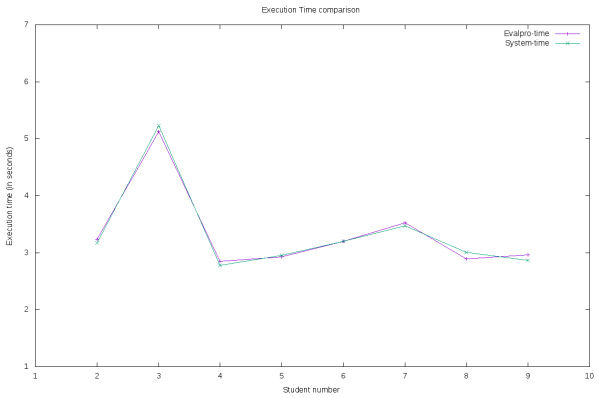



Figure 5.8: Comparison between EvalPro time and System time for test case 1

Average error difference between EvalPro time and system time for test case 1 among 8 students was found to be 0.062 seconds.

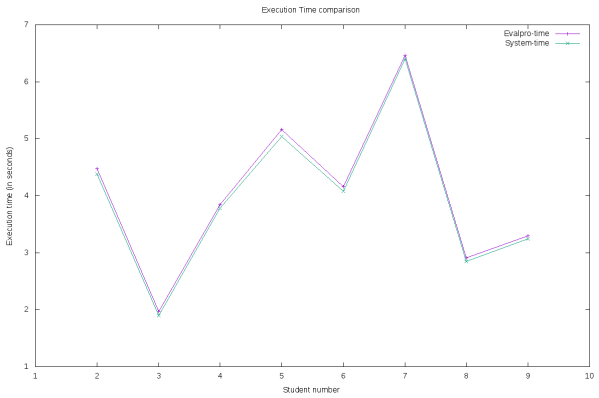



Figure 5.9: Comparison between EvalPro time and System time for test case 2

Average error difference between EvalPro time and system time for test case 2 among 8 students was found to be 0.0775 seconds.

## 5.3 Assignment Repository

A lot of assignments are created in a span of a typical semester i.e 3-4 months. For example, CS101 lab course at IIT Bombay has on average 3-4 assignments a weeks, and there are around 12-15 labs conducted in a complete semester and there are 3 batches running parallely throughout the semester. All these assignments sum up to 4*12*3 = 150 (approx). There are around 150 unique assignments created each semester for a particular course which is a huge number. Also, CS101 course runs every semester in IIT Bombay with same number of students registered. Saving these assignments in a central repository will help instructors taking the same course in future and also other course instructor who wants to include same assignment in to his/her course. Assignment repository serves two purpose, one is reusing the assignment that was created in same course in previous years and other is using other course assignment in instructor's own course.

### 5.3.1 Implementation

We have created a dummy course named "Repository" which serves the purpose of saving the assignments. All actions related to repository can only be performed by instructor. Instructor can perform three actions as mentioned below -

1. Save an assignment to repository
   There is an option on assignment page to save an assignment to repository.

When instructor clicks on this save button, a copy of current assignment is created in repository course. An assignment is associated with multiple sections and each section has multiple test cases. Along with assignment, sections and test cases needs to be copied to repository course. When instructor saves an assignment, EvalPro copies an assignment along with sections and test cases associated with it.

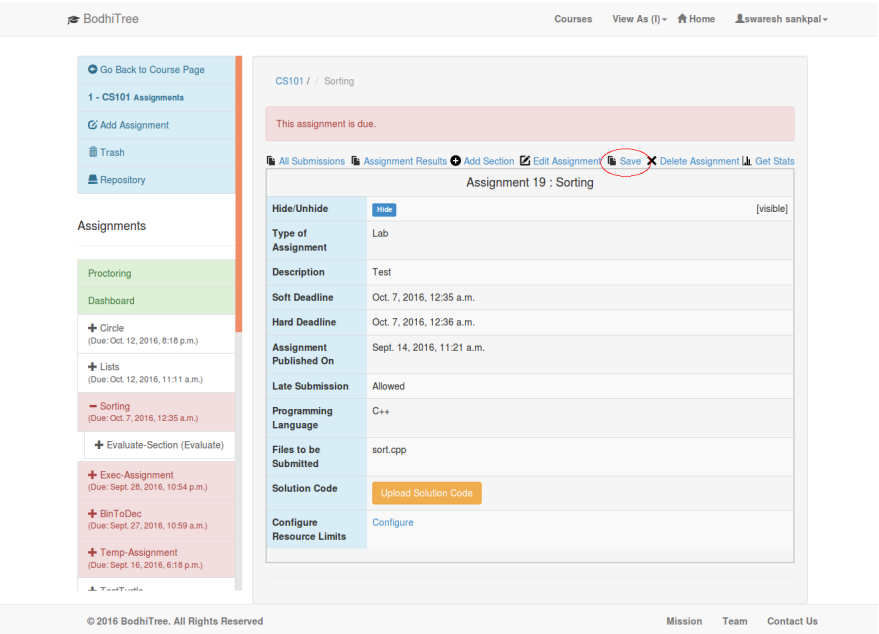



Figure 5.10: Save option that copies assignment from one course to another

2. Copy assignment from repository to current course

   We have given an option to instructor to choose to copy an assignment from repository into current course. This action can be performed from create assignment page by clicking on button titled with "choose from repository".

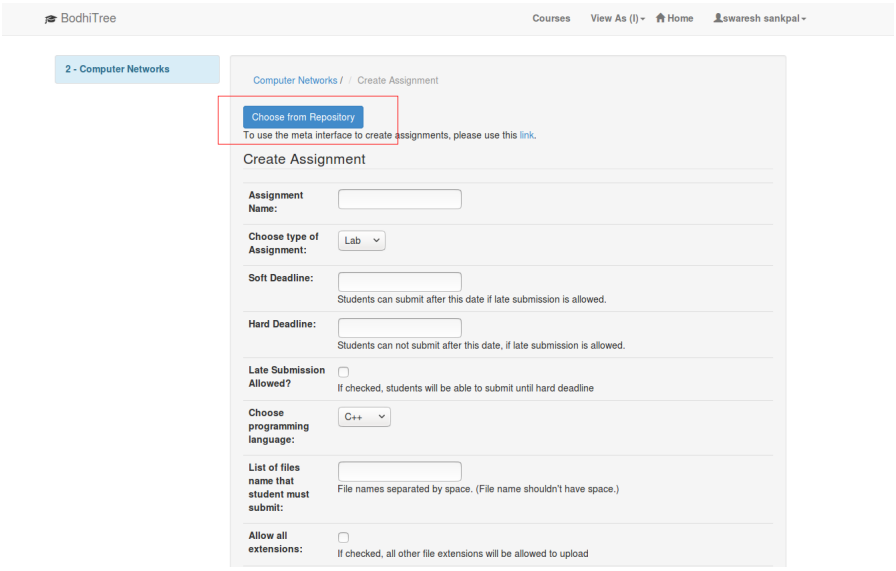



Figure 5.11: Choose from repository option in Create Assignment page

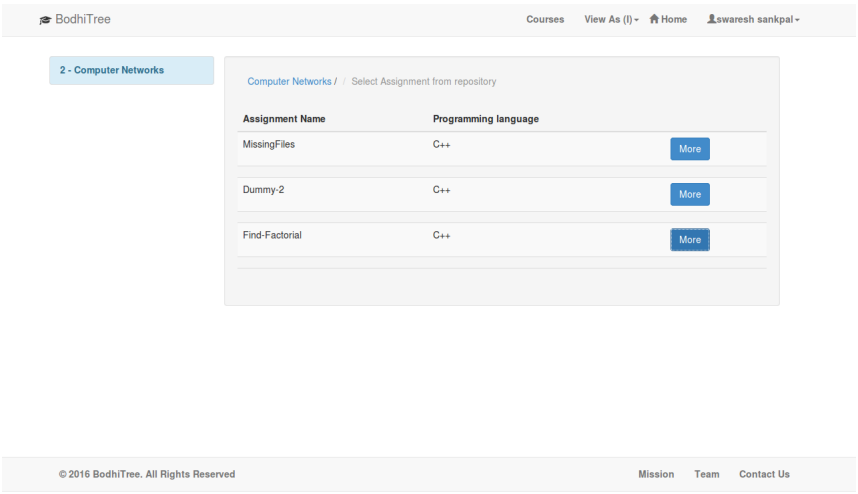



Figure 5.12: List of assignments in repository

The instructor can now choose an assignment from the list of assignments displayed in repository. Instructor now gets to see a list of assignments present in repository along with the corresponding programming language and a "more" option which internally has -

- Assignment description

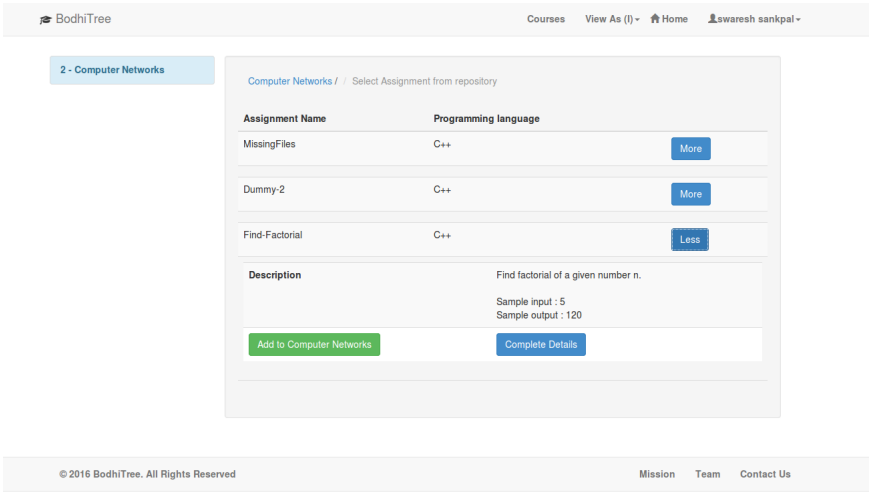



Figure 5.13: Description of an assignment in repository

- Complete details
  Instructor can view the complete details of an assignment he/she is interested to copy. This page show complete details of that assignment along with number of test cases present in assignment in a minimal design in a new tab.

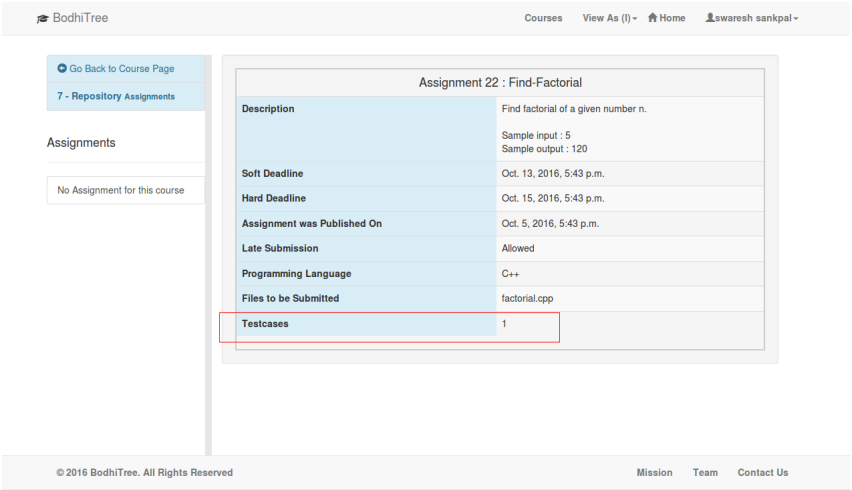



Figure 5.14: Complete details of an assignment in repository

- Add to current course
  This option adds the assignment selected to current course of instructor. It also shows a pop-up to confirm the copy action. If assignment with

same name selected already exists in current course, EvalPro throws an exception saying assignment with same name already exists in current course. If instructor selects yes option the paticular assignment is copied from repository course to current course.

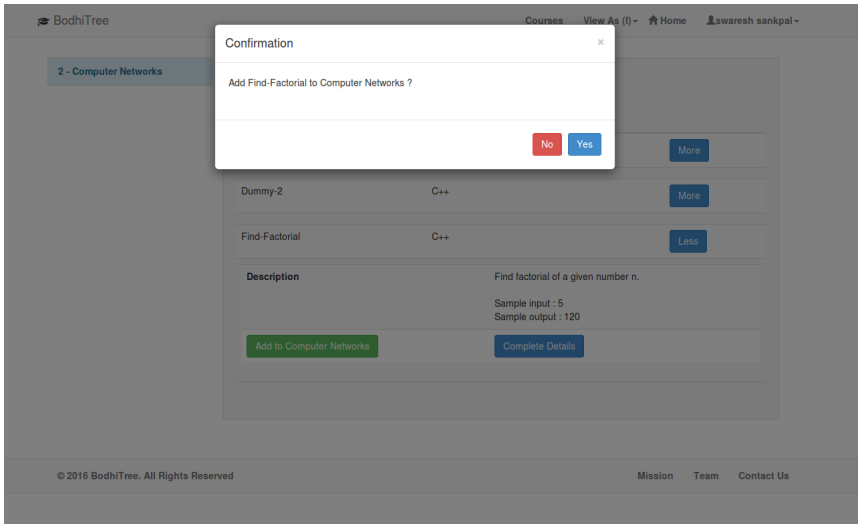



Figure 5.15: Confirmation related to copy request made by an instructor

3. Copy assignment from repository option on assignment page

This has similar internal functionality of copying assignment as shown above. In above method the instructor can only copy an assignment from repository to his current course, but from this repository option an instructor can copy an assignment from repository to any course he/she owns.

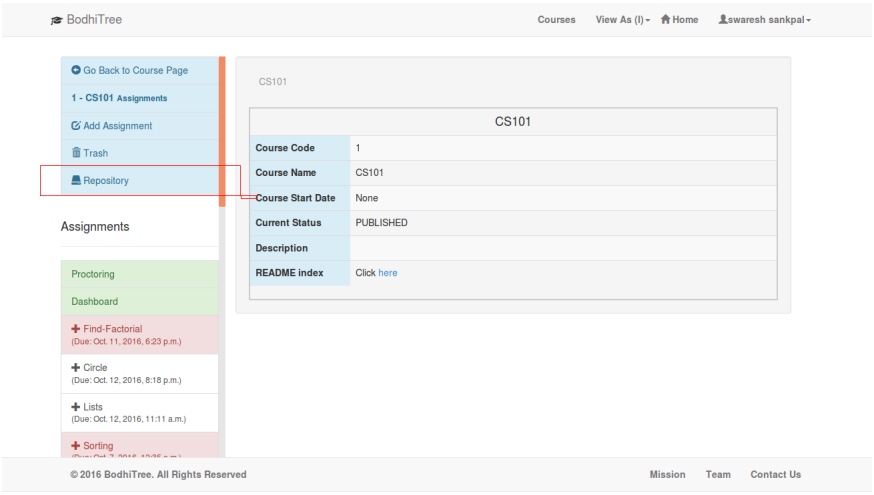



Figure 5.16: Repository

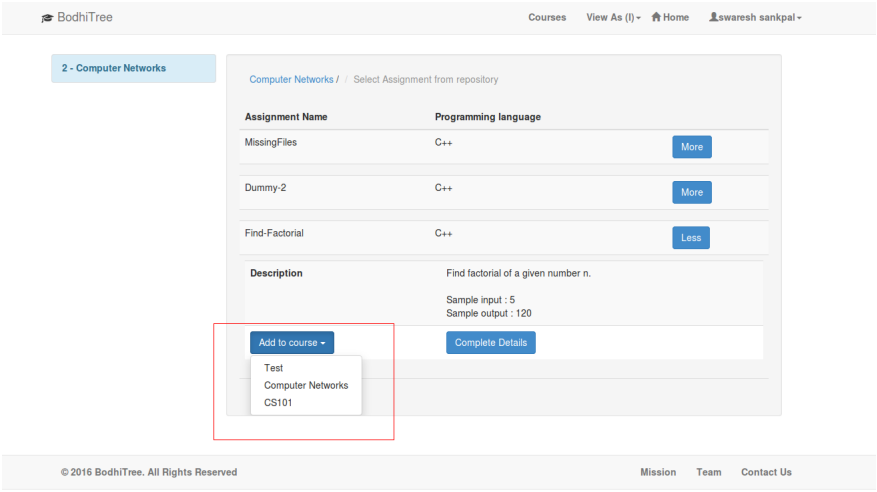



Figure 5.17: Copy assignment from repository to any course that instructor owns

## 5.4 Assignment Trash

Currently in EvalPro, there is an option to delete an assignment which deletes a complete assignment along with its sections, test cases, submissions etc. This action is sensitive and should be reconfirmed as many times as possible. Once an assignment is deleted, there is no option to recover it. There were incidents in development server as well as production server where this option was mistakenly

used and all data was lost unintentionally. There would be no other option than re-creating as assignment to cover up the delete action. The concept of trash would solve this problem in EvalPro and hence we developed such feature.

### 5.4.1 Implementation

We added a boolean field "trash" to assignments model which is set to "false" as default. When an assignment is deleted, this trash field of that assignment is set to "true". The deleted assignment is no more seen in the list of assignments of that course. The assignment is moved to trash of that particular course.

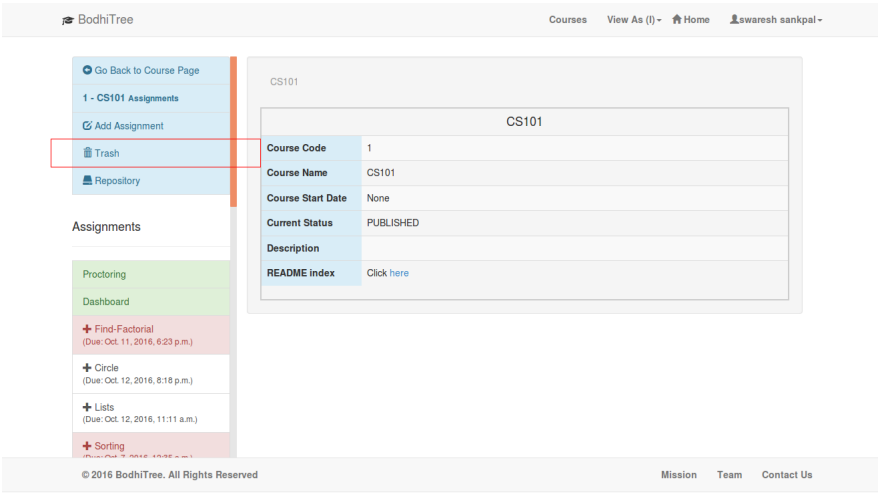



Figure 5.18: Trash of an course

For every assignment in trash there are two actions that can be performed -

1. Restore
   This moves the assignment from trash to the current course.

2. Permanently delete
   This option removes the assignment completely from database.

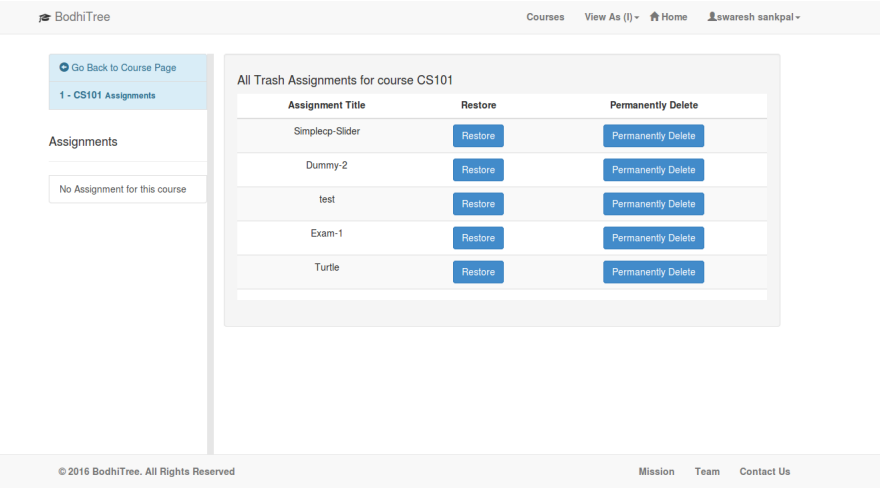



Figure 5.19: Assignments in Trash

## 5.5 Manual grading of a submission

Grading scheme adopted during CS101 offered in Autumn 2015 -

1. Evaluate a submission of a student using evaluate button
2. Grade a submission based on marks given by EvalPro plus the additional marks according to the grading scheme decided by instructor
3. Enter these marks into a google sheet along with feedback about the submission
4. Share this sheet with students so that they can see their marks along with TA's feedback

Issues with above approach -

- Management of this google sheet is a time taking and tedious work
- High possibility of human errors

### 5.5.1 Solution

1. Give an option along with each submission of student to give total marks i.e EvalPro marks plus extra marks as per grading scheme made by instructor.

2. Along with total marks there should be an option to give feedback for every submission

3. From student's view, there should a page where they can view all the submission made by them along with marks and comments given by TA

### 5.5.2 Implementation

- To enable instructor/TA give total marks and comments on student's submission, we have added two columns i.e final marks and comments to existing all submissions page. TA's can evaluate the submission using evaluate button, based on that and marking scheme of instructor, they can enter the total marks in final marks column added and give feedback through comments field provided. Hence, a submission is completely evaluated from a single page in bodhitree. Time and human resources are saved compared to initial approach of grading in EvalPro.

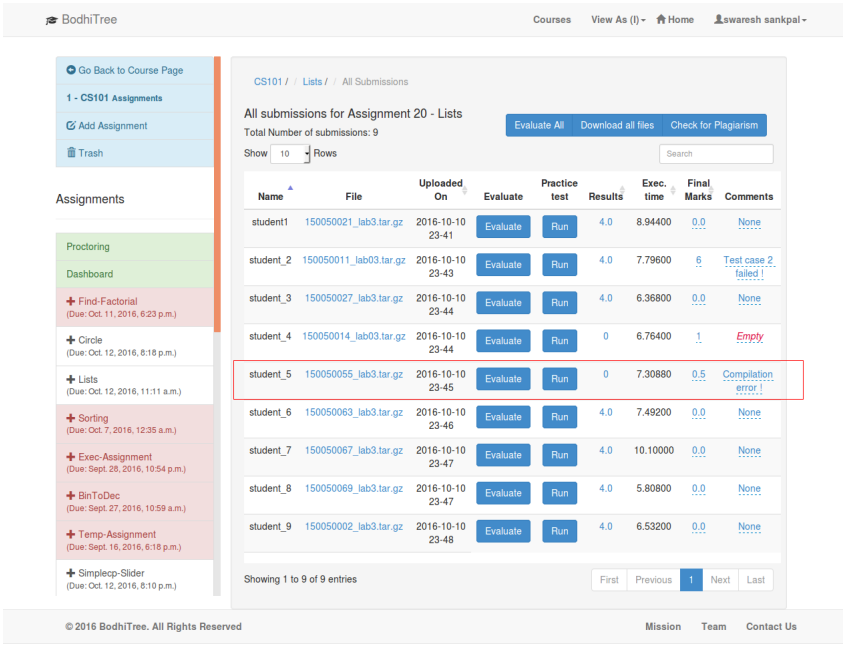



Figure 5.20: Manual grading of an assignment

- To enable students to see their marks and comments, we have added a new html page that shows all submissions made by students along with the marks obtained and the feedback given by TA. Students can view this page from "Past submissions" option in menu where we find logout option.

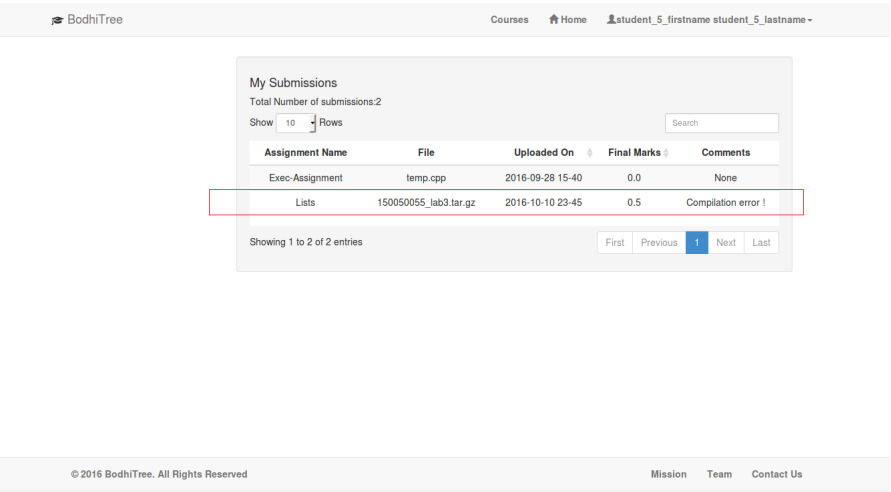



Figure 5.21: Past submission page for students

## 5.6 Email support in Crib management system

### 5.6.1 Problems in current implementation

Crib management system is one of the main features of EvalPro and is used very frequently by students to post cribs and instructors to resolve them. It was observed that when students use to post a crib or instructor use to post a comment on crib, there was lack of notification system which would result in longer time in resolving crib. TAs had to constantly check the dashboard to see if there any new cribs made by students.

### 5.6.2 Solution and Implementation

To solve the problem explained above, we have implemented an email feature that notifies TA, when student posts a crib and notifies student when TA posts a comment on student's crib through email. This feature does two tasks :

- Send email to TA when a new crib is posted by student
  This email includes the student username who has posted a crib, assignment name with which crib is assiciated and the crib description.

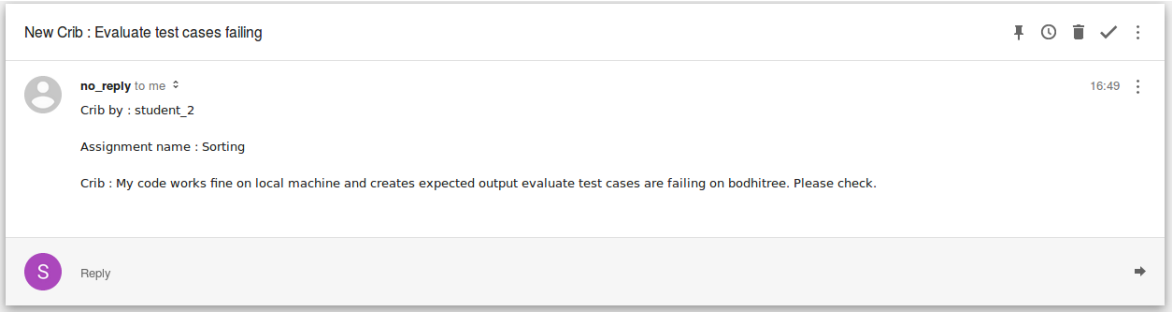



Figure 5.22: Example of an email sent to TA when student posts a new crib

- Send email to student when TA comments on his/her crib
  This email includes the TA's username who has commented on his/her crib, assignment name with which crib is related and the comment made by TA.

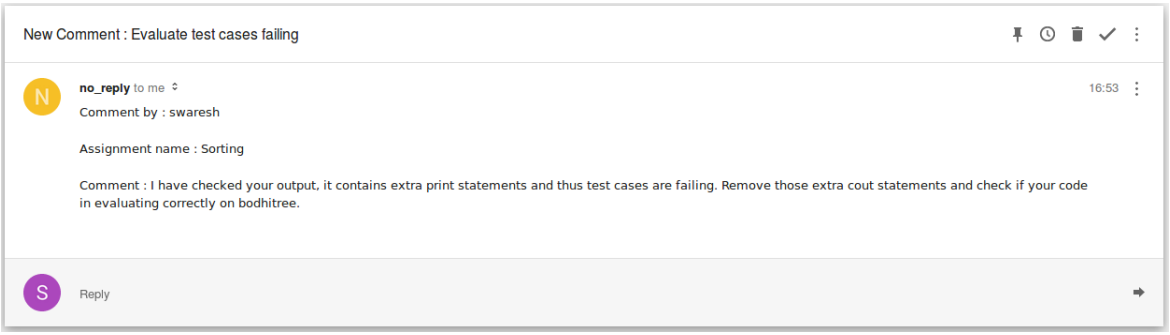



Figure 5.23: Example of an email sent to student when TA posts a new comment on his/her crib

## 5.7 Feature to map students and TAs together

### 5.7.1 Problems in current Implementation

- In recent deployments of EvalPro, it was observed that instructor use to divide the evaluation work equally within TAs and the evaluation task of each TA was either communicated through personal emails or through a excel sheet containing student to TA mappings. This technique works fine though it has some limitations like TAs have to search the submissions that has to be evaluated by them from all the submission that have been submitted for that assignment and there were many cases where TAs use to forget to evaluate some of the submissions. But, this techniques performs badly in courses such

as CS101 where more than 1000 submission are made every week. So, there was a need to create mapping between student and TAs to carry out the evaluation process of an huge class smoothly.

- According to current architecture of EvalPro, a single TA is assigned for a single assignment to resolve all the cribs that are made for that particular assignment. This was a bad design since it increase work load on a single TA and will take more time for a single TA to solve all the cribs for that assignment specially for a class of 500 students. Student to TA mapping for every assignment will distribute the cribs equally among the TAs and help the crib resolving process work faster and efficient.

### 5.7.2 Solution & Implementation

To solve the above mentioned two problems, we have developed a feature that maps students and TA for each assignment. We have provided instructor with three different policies to create these mappings between students and TAs:

1. Automated TA allocation Policy
   This policy creates two lists, one of students registered for current course and other of TAs associated with the course. Then it equally divides the students among the available TAs.

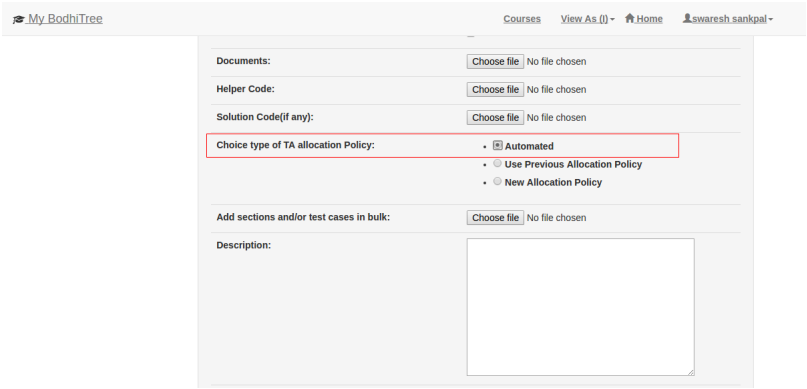



Figure 5.24: Automated policy option of TA allocation

2. New TA Allocation Policy
   According to this policy, instructor can upload a csv file containing the student and TA mappings.

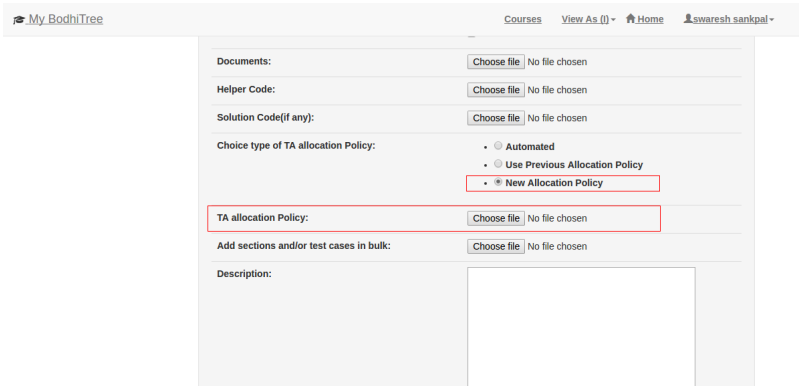



Figure 5.25: Upload a new TA allocation policy

EvalPro performs following validation checks on the file uploaded to create mappings :

- Only comma separated csv file are allowed.
- There should be two columns in csv file with names student_username and ta_username respectively.

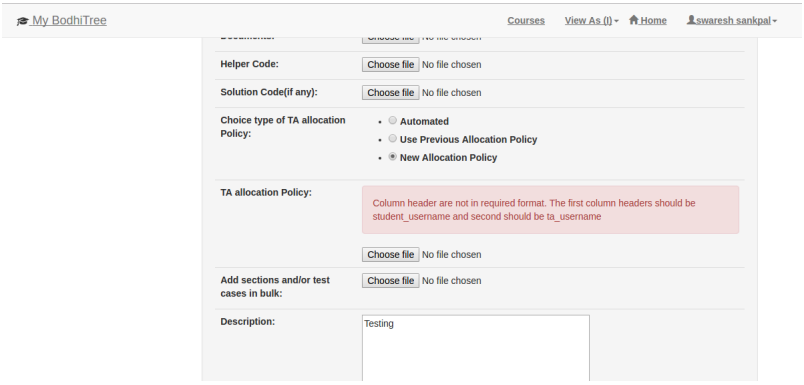



Figure 5.26: Form error with wrong column headers in TA allocation document

- All the column 1 entries should be valid usernames of students and column 2 entries should be valid usernames of TA.

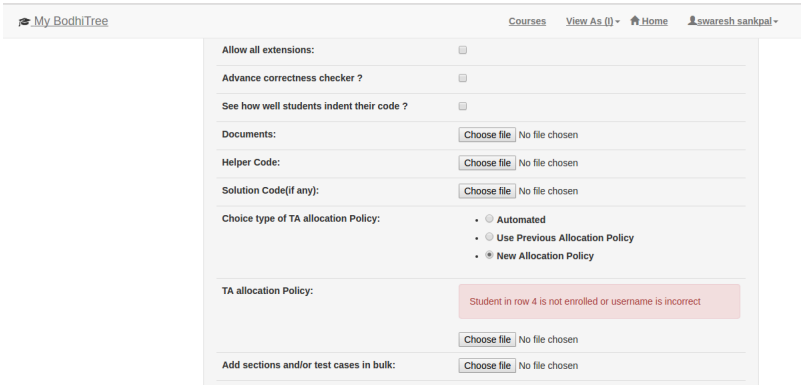



Figure 5.27: Form error with wrong student username in TA allocation document

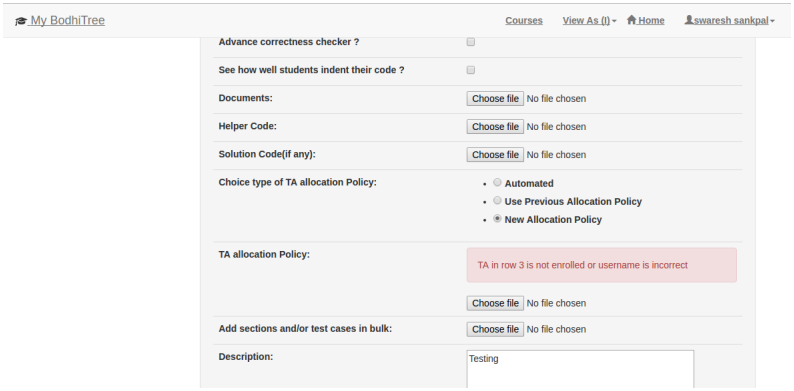



Figure 5.28: Form error with wrong TA username in TA allocation document

If above instructions are not followed, form validation fails and shows respective erros.

3. Use Previous TA allocation Policy
   This policy provides instructor with the option to use the mappings of the previous assignments into the current assignment. This option basically copies all mapping from selected previous assignment into the newly created one.

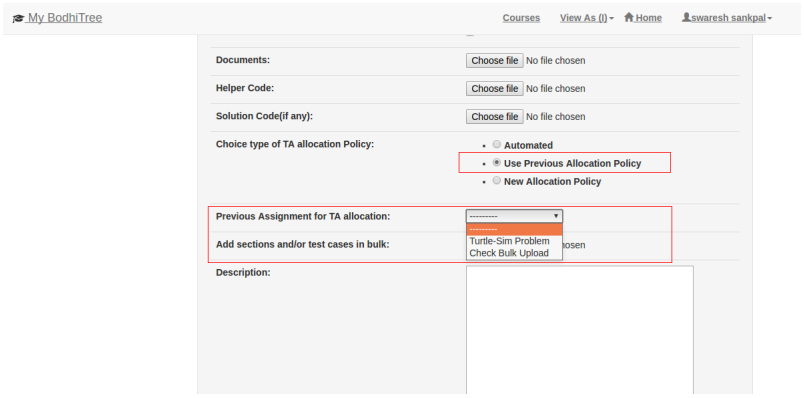



Figure 5.29: Use previous TA allocation policy

We have created a radio select as shown in figure below in all submissions page that gives an option to TA to either view all the submissions of the assignment or only submission that has to be evaluated by him. This reduces the time taken by TA to find out his/her students and evaluate their submissions and also no submission are neglected for evaluation.

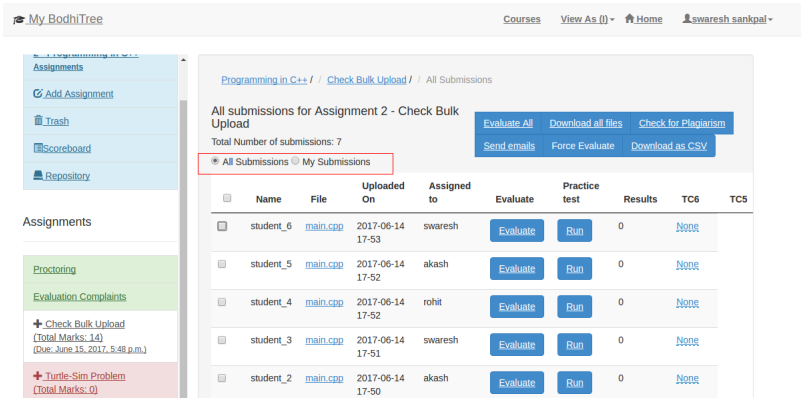



Figure 5.30: My submissions/ All submissions radio select in all submission page

## 5.8 Bulk Download

### 5.8.1 Problems in current Implementation

Current approach to download test case files:

- Click on assignment page
- Click on program page

- Click on test case page
- Click on input file/ output file to download it

For example, if an assignment consists of two sections and 4 test cases in each section, it takes total of around 25 clicks to download all test cases files which is a very bad design for download option in web application. This was a critical bug of EvalPro since students and instructors often download these test case files to check the correctness of program and needed to be fixed soon.

Also it will be very helpful if all the assignment files like assignment document, helper code as well as problem statement is downloaded along with all the test cases of the assignment. This feature can be extended to course level so that an instructor can download all the files related to all assignment present in his/ her course for future use in a single click.

### 5.8.2 Solution & Implementation

- Download all files of an assignment
  We have implemented a feature that download all files related to an assignment in a single click. This download option exists on assignment details page as shown in figure below:

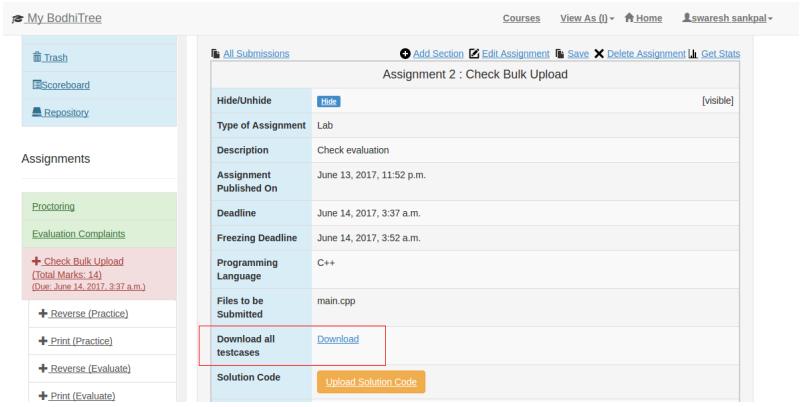



Figure 5.31: Download all files of an assignment

In student's view, only practice test cases will be downloaded before the deadline and all test cases after deadline. While in case of instructor, all test cases will be downloaded at any moment of time. The downloaded tar includes four folders - Problem description, Assignment Document, Helper code and

test cases. Tree representation of a typical download is shown in screen shot below.

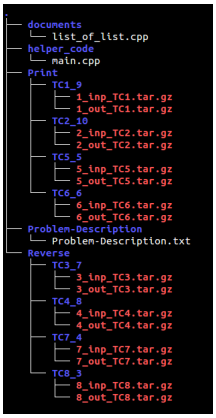



Figure 5.32: Hierarchy of a downloaded folder

- Download all files of a course
  Similar to download option in assignment level, we have implemented a download option at course level that downloads all files related to a course. This option is provided on course index page as shown in following figure:

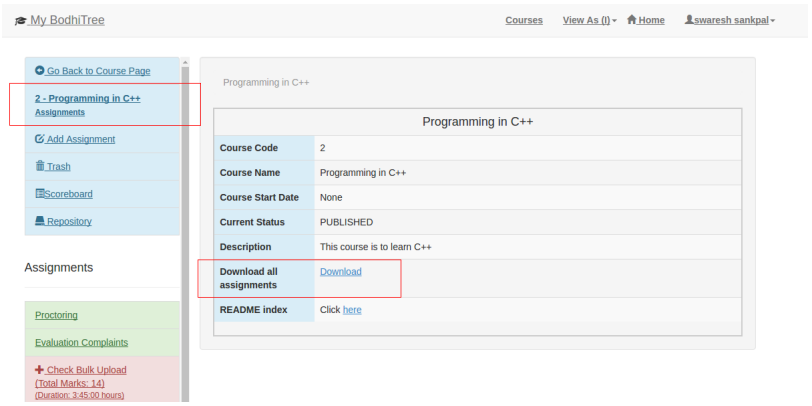



Figure 5.33: Download all assignment files of a course

Tree representation of folder downloaded from above mentioned option is shown in figure below

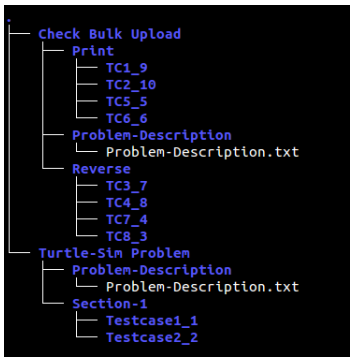



Figure 5.34: Hierarchy of a downloaded folder

## 5.9 Visual aids to compare output files

EvalPro, runs the student code against the test cases provided by instructor and shows the result i.e actual output and expected output for each test case so that the students find out the logical errors in their code if there are any. In current implementation of EvalPro, following issues related to comparing actual output and expected output were found.

### 5.9.1 Problems in current implementation

- Last year in data structures lab at IIT Bombay, it was observed that students had to scroll up/down frequently to compare actual output with expected output since they were displayed one after the another. This was a major usability issue in EvalPro. To tackle this we displayed the actual output and expected output side by side to improve the usability.

- Another issue was regarding white spaces mismatch that would cause the test case to fail. It impossible to find out the space mismatches with the previous implementation of EvalPro, hence a advanced visual comparison tool was required to catch such mismatches.

- According to current implementation of displaying actual and expected output, it is very hard to find minor mismatches in a very long output. Visual differences if provided will help catching such errors very easily.

### 5.9.2 Solution & Implementation

We have used a python module named **Difflib** [8] which provides classes and functions for comparing sequences. It can be used for comparing files, and can produce difference information in various formats, including HTML and context and unified diffs. We have used make_file method of this module to compare two files and generate a HTML string containing a table showing line by line differences with inter-line and intra-line changes highlighted. These highlighted changes are very useful in determining the deltas in actual output and expected output. This HTML string is then parsed using **Beautifulsoup** [9] library of python and the desired HTML format is generated. Following are the cases in which this visual aids feature outperforms the existing feature of actual output and expected output comparison.

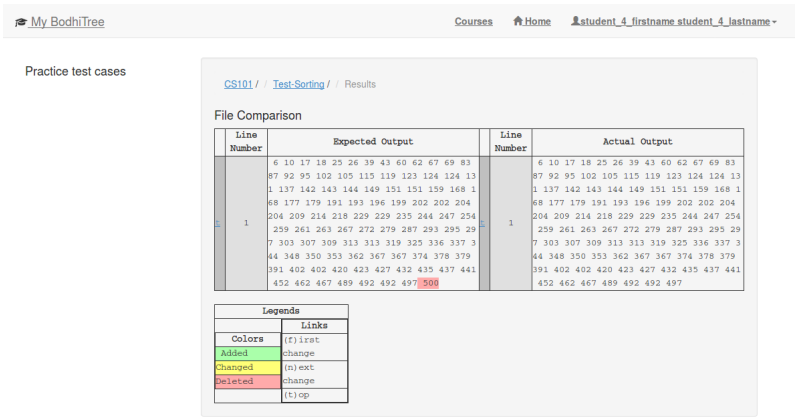



Figure 5.35: Last number missing in actual output

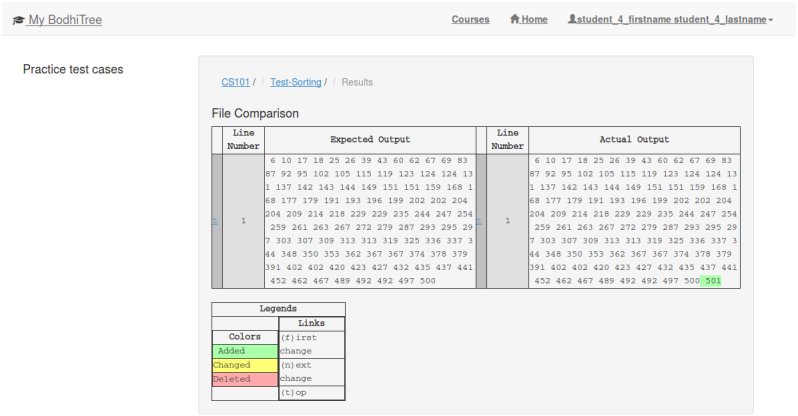



Figure 5.36: An extra number in actual output

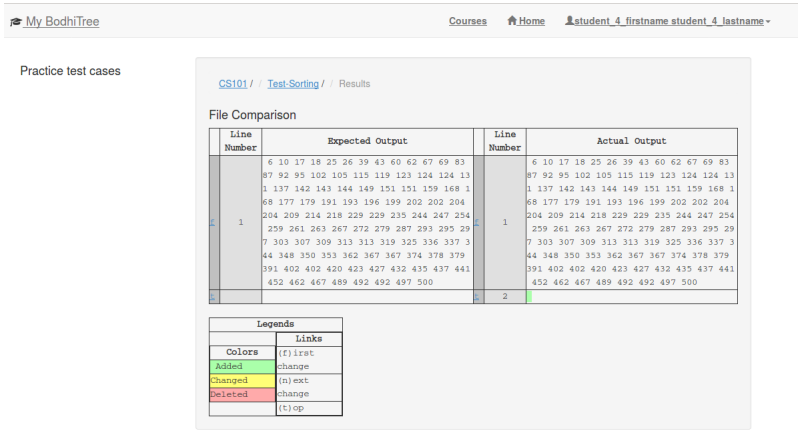



Figure 5.37: Extra newline at the end of actual output

## 5.10 In-Browser Code Editor

It was observed that almost all the auto evaluation tools had In-Browser editor integrated within because it has lot of advantages that students can make use of and focus on programming aspect. It was time to integrate such in-browser editor into EvalPro. We used Codemirror[7] text editor as a in-browser editor in EvalPro. CodeMirror is an open-source project shared under an MIT license. It is the editor used in the dev tools for Firefox, Chrome, and Safari, in Light Table, Adobe Brackets, Bitbucket, and many other projects. CodeMirror is a versatile text editor implemented in JavaScript for the browser. It is specialized for editing code, and comes with a number of language modes and addons that implement more advanced editing functionality. A rich programming API and a CSS theming system are available for customizing CodeMirror to fit our application, and extending it with new functionality.
It has many features like:

- Support for over 100 languages out of the box
- A powerful, composable language mode system
- Autocompletion (XML)
- Code folding
- Configurable keybindings
- Vim, Emacs, and Sublime Text bindings

- Search and replace interface
- Bracket and tag matching
- Support for split views
- Linter integration
- Mixing font sizes and styles
- Various themes
- Able to resize to fit content
- Inline and block widgets
- Programmable gutters
- Making ranges of text styled, read-only, or atomic
- Bi-directional text support

Following are the screen shots of Code mirror editor integrated into EvalPro.

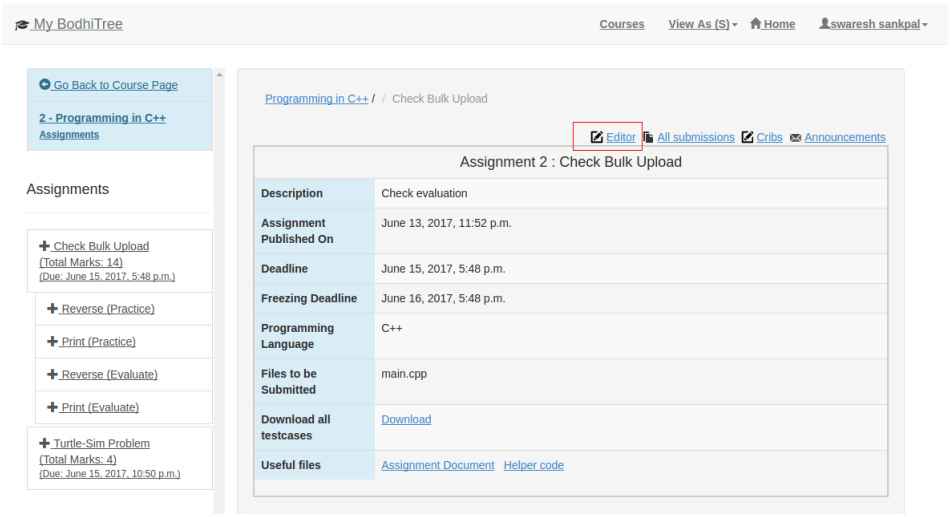



Figure 5.38: Editor option on assignment details page in student's mode

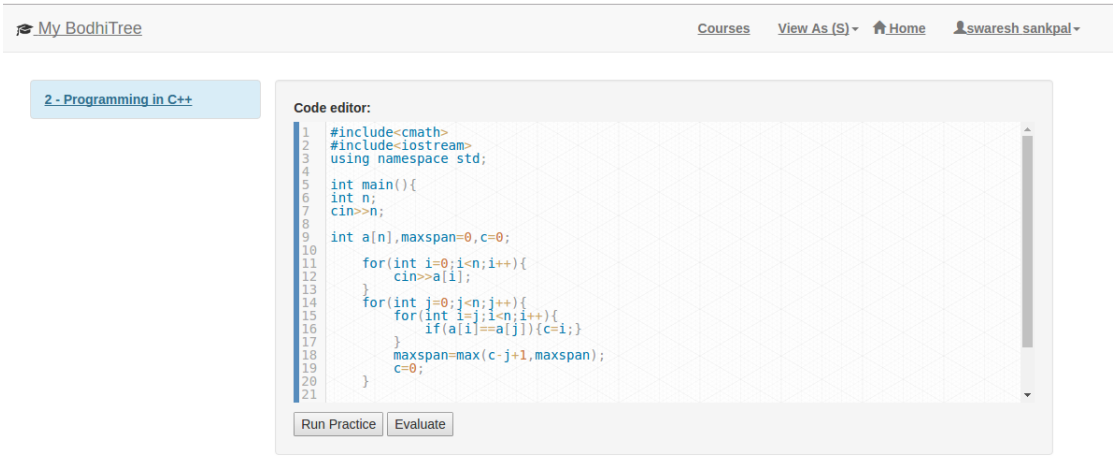



Figure 5.39: Codemirror integrated into EvalPro

When a student writes code through code editor and clicks on Run Practice test/ Evaluate, EvalPro creates a Upload object and the code written in the editor is saved in the database and test cases are run against it to check the correctness of the program. If student make changes into the code and re-submits only then the new Upload object is created, else if there is no change in the code and student click run/ evaluate multiple times, older results which were saved are returned instead of re-evaluating.

## 5.11 Archive student's intermediate submissions

### 5.11.1 Problems with current implementation

Students usually make multiple uploads before the deadline of an assignment if they find submission partially or completely incorrect. Currently in EvalPro, only the latest submission is saved i.e the older submission is overwritten by the newer one. It may happen that student changed his code which was partially working before and uploaded a newer one, and this newer version gets zero or doesn't even get compiled. In this case, the last version which doesn't work is considered for evaluation except the fact that his intermediate submission was partially working. Hence, there must be a feature in EvalPro that saves these intermediate submissions of students along with marks so that if they make some mistake and upload wrong code, they get graded based on the submission with highest grade obtained till end.

### 5.11.2 Solution & Implementation

- Save intermediate submission
  According to current implementation of EvalPro, when student uploads a new code, the older one is replaced by the newer one. We have changed this architecture to archive all the intermediate submission of student. Multiple files with same name cannot exist in the same directory, hence we had one more hierarchy to differentiate between different submissions of same student. Following is the directory structure of a student with 7 submissions for the same assignment :

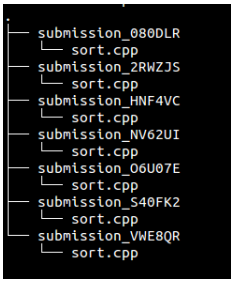



Figure 5.40: Directory structure with multiple submission of the same student

- Freeze submission
  Since all the intermediate submission are saved, it needs to be decided by student that which submission has to be considered for evaluation. To support this decision, a new django model is created in EvalPro that saves the submission ID for every assignment per student that has to considered for evaluation. Also, a new HTML as shown below is created to view all the submission made by student for that assignment.

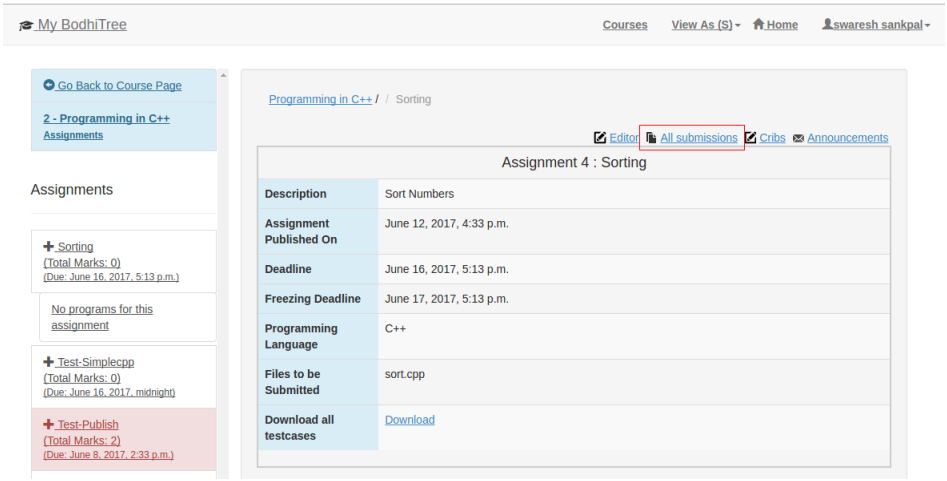



Figure 5.41: All submissions option in student's view of assignment details

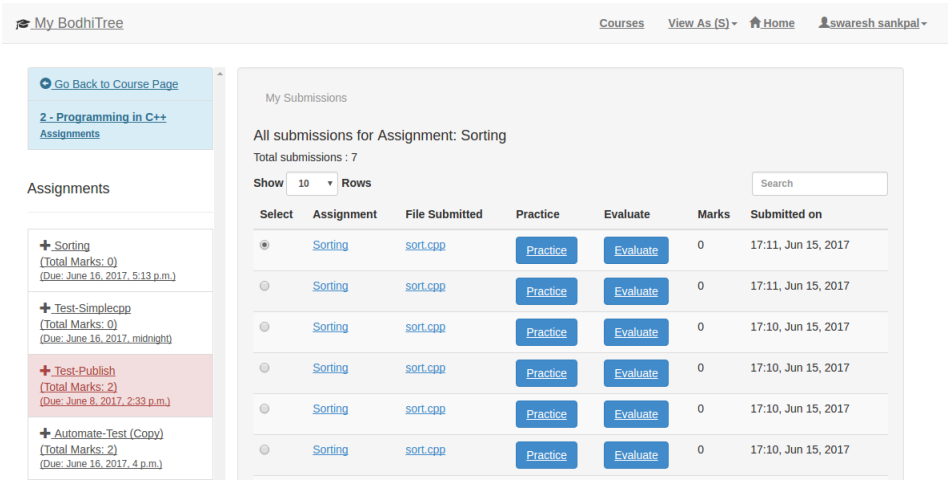



Figure 5.42: All submissions page of a student for particular assignment

We have provided student with a freeze button that helps student to select the submission that has to considered for evaluation. Student can freeze the submission till the freezing deadline, which has been newly added to EvalPro's assignment structure.

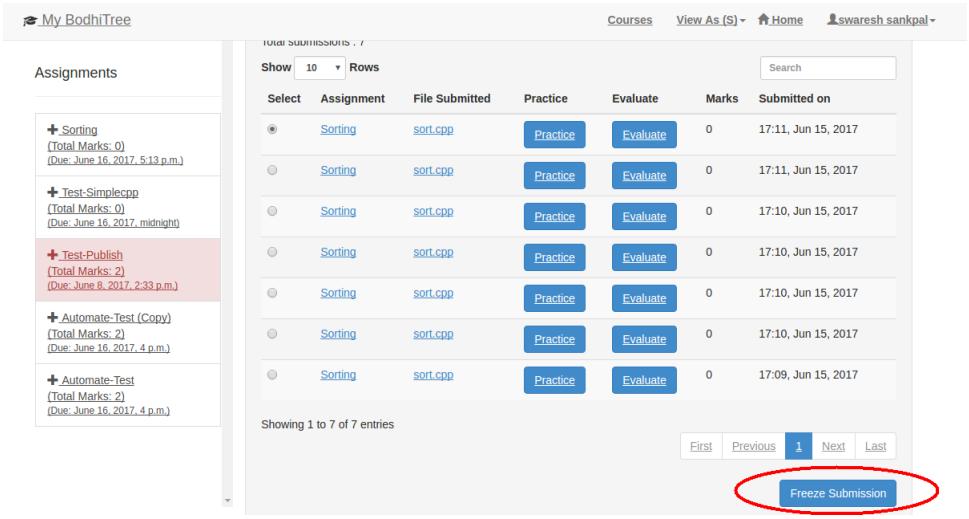



Figure 5.43: Freezing submission option for students

- Freezing dealine
  Concept of hard deadline has been removed from EvalPro completely since many of the instructors using EvalPro found it redundant. Freezing deadline was introduced to support this feature of archiving intermediate submissions and allow student to decide upon the submission to be considered for evaluation even after the deadline. This allows students to worry less about the errors that may cause their submission to fail completely and focus on getting full marks.

- Use of celery to delete redundant submissions
  This feature of archiving intermediate submissions creates a lot of files per student and if the class strength is high, this feature causes high disk usage. If a student makes say 10 different submissions, for a class of strength 100, 1000 files are saved on the disk. This means 900 extra files are saved as compared to previous architecture. To reduce disk usage, we have developed a module using **Celery** [10] that deletes all the files other than the one that has selected by students for evaluation after freezing deadline is over.

  – A task is schedules at freezing deadline when an assignment is created which deletes all the files other than the one that has selected by students for evaluation. This task is scheduled using apply_async function of celery.

  – The task id of the task responsible to delete all files except the ones

chosen for evaluation by students is stored in assignment object in the database.

– Whenever an assignment's freezing deadline is edited, the task that was scheduled before needs to be deleted and a new task has to be created to delete the files. The older task is deleted using revoke function of celery.

## 5.12 Publish/ Unpublish solution code on demand

### 5.12.1 Problems with current Implementation

- Only instructor can download the solution code.
- Solution code cannot be uploaded after the deadline.
- Download option available just after deadline is over. It may happen that instructor wants to extend the deadline after the original deadline is over, but solution code would have already published. So, there is a need of on demand publish option for solution code feature.

### 5.12.2 Solution & Implementation

- We have provided option for students to download the solution code once it is published by instructor as shown in figure below. Download option will only be visible to student when deadline is over and solution code is published by instructor.

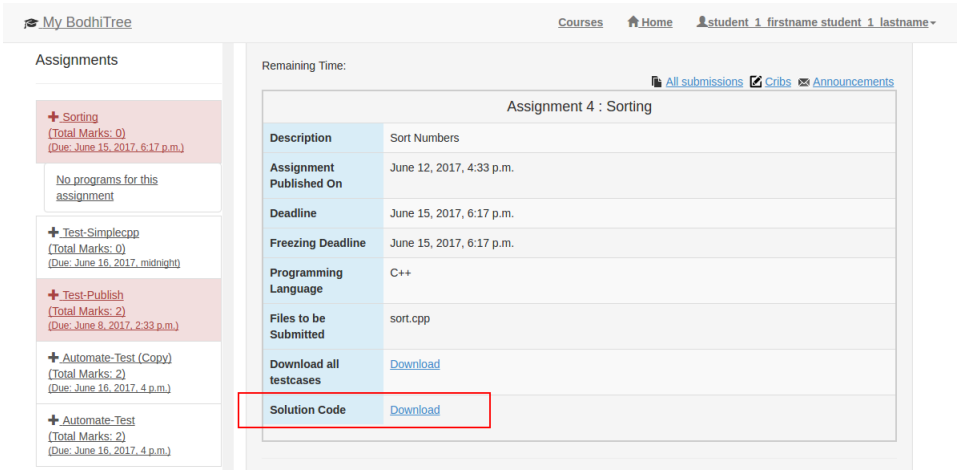



Figure 5.44: Download solution code option for students

- Now instructor can upload a solution code even after the deadline is over.
- We provided Publish/ Unpublish option for instructor to publish/ unpublish the solution code as shown in screen shot below.

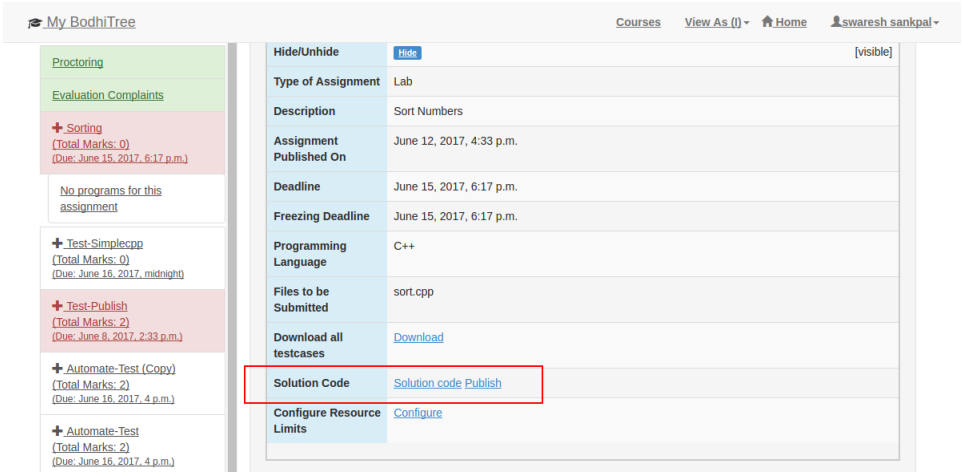



Figure 5.45: Publish/ Unpublish option for instructor

## 5.13 Calculate Indentation percentage of a program

In the basic programming course, It is required for students to learn coding style along with programming language. Here coding style is set of rules that is used while writing a code. It is as important as logic of program because it becomes easy for someone to understand the code if it is written in good coding style. For any program, we can say in YES or NO that it is correctly indentated or not, But we can determine that up to what extent(in percentage) it is correctly indented. For programming language like python we do not need to worry about code indentation because it does not run if it is not correctly indented, but programing language like C,C++, they do not take care of this so we need to use some mechanism to calculate the indentation of the given code.
This feature was implemented by Nirav but was not integrated with EvalPro and also it worked for submission with single file. My task in this feature was to integrate this feature into EvalPro and also extend this feature to submission with multiple files(tar).

Rules used to calculate indentation percentage :

- First remove all the blank lines from the code and convert all the tabs into spaces.

- From this remaining lines, find number of initial spaces for each line and count the number of liens that is correctly indented
- How to decide that any line is correctly indented or not ?
  - if line does not contain "{" or "}" than number of initial spaces should be same as previous line
  - If line contain "{" than number of initial spaces should be same as previous line and next line's initial spaces should be equal to current line's initial spaces+distance(d)
  - if line contain "}" than number of initial spaces should be equal to initial spaces of previous lines-distance(d)
  - if line contain "{" and "}" then number of initial spaces should be same as previous lines
- Calculate the percentage of lines which are correctly indented

### 5.13.1 Solution & Implementation

- We have created a boolean field at assignment level that decides whether to calculate the indentation percentage of student's program or not.

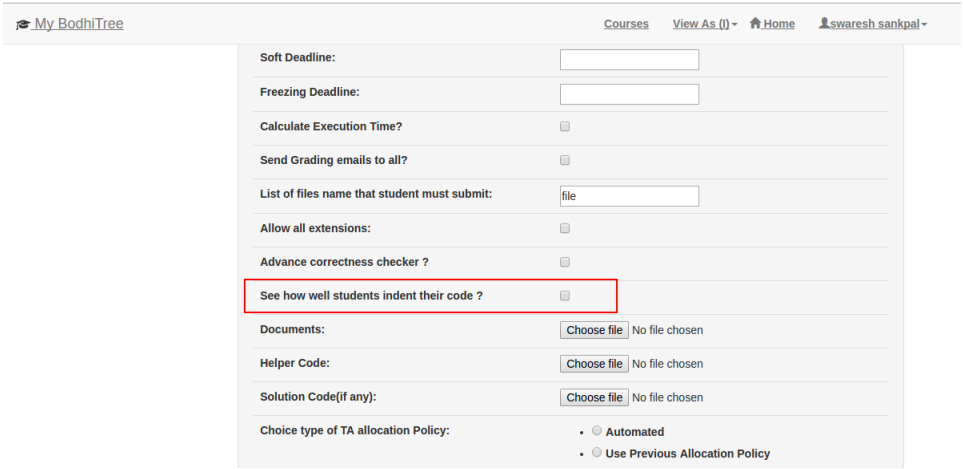



Figure 5.46: Indentation flag in Create Assignment page

- It was observed in Computer Networks course at IIT Bombay that the indentation feature works perfectly fine for single file submissions but irregular results were obtained for submissions with multiple files(tar upload). The

indentation feature was developed only to work with single file uploads. We extended this feature for submission with multiple files. To validate the results of indentation feature, we cross checked the marks given by TAs by manually checking the indentation with indentation percentage given by EvalPro. Most of the submission got 3 marks, there were 3 submissions which were given 2 marks for indentation by TAs. So, we took 3 submission with 3 marks, 3 submissions with 2 marks and uploaded in the new assignment created on server with latest code. Following are the results :

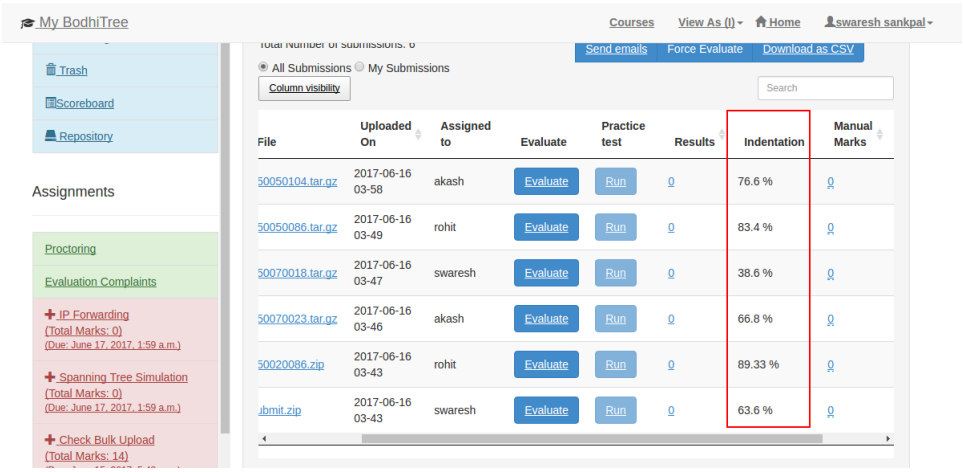



Figure 5.47: Indentation Percentage in All submission page

| Roll no. | Marks | Indentation % |
|---|---|---|
| 150050107 | 2 | 63.6 |
| 150070018 | 2 | 38.6 |
| 150070023 | 2 | 66.8 |
| 150050086 | 3 | 83.4 |
| 150050104 | 3 | 76.6 |
| 150020086 | 3 | 89.33 |

Table 5.1: Indentation validation experiment

## 5.14 Minimal Design for Non-Programming Assignments

EvalPro was developed in vision of automating the evaluation of programming assignments. EvalPro is a very good platform to host non-programming assignments too but some professors found it too complex. Since EvalPro offers a good platform for managing assignments, professors used it very often for programming as well as non-programming assignments. EvalPro needs a little tuning to make it compatible with non-programming assignments. Following are the problems that instructors faced with non-programming assignments in EvalPro

- There are too many fields in assignment creation which are unrelated to non-programming assignments and put a tight constraints in creating assignments which professors found annoying. These are field like programming language choice, execution time calculation, advance correctness checker, indentation calculation, bulk upload of test case etc.
- Also from students perspective, the file name validation is unnecessary in non-programming assignments.

### 5.14.1 Solution & Implementation

- We have created a boolean flag in assignment create page that hides all unnecessary fields using javascript in respect to non-programming assignments, when checked. The fields hidden are programming language choice, execution time calculation, advance correctness checker, indentation calculation, file name that student must submit, bulk upload of test case etc.

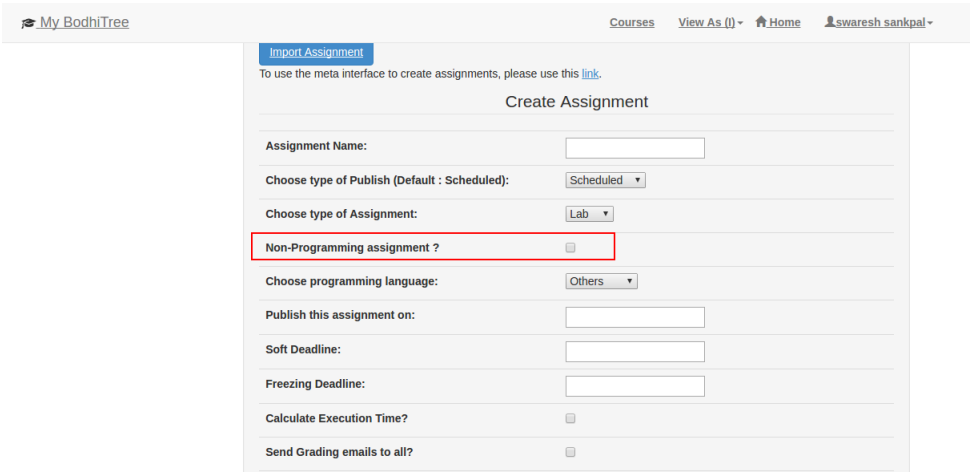



Figure 5.48: Non-Programming Assignment flag in assignment create page

- No validation check is done on file name that student submits if the assignment is marked as non-programming assignment. Student can submit file with any name.
- No option to create sections, run/ evaluate files if assignment is marked as non-programming.

## 5.15 Miscellaneous

Fixing bugs/issues of EvalPro's existing features is as important as adding new features to EvalPro's functionality. We have fixed following bugs of EvalPro -

### 5.15.1 Unique Test case file name

While evaluating submissions of a lab, it was found out that EvalPro is giving inconsistent marks for the same submission when evaluated multiple times. Later, after a little study on this issue it was found out that the problem arose due to same test case file names in different section of same type. For example, if there are two test cases with same names but under different sections of practice or evaluate, it caused issue. Let's consider following example -

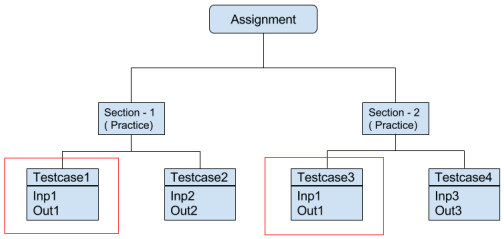



Figure 5.49: Bug with same test case file names

In above figure, Testcase1 and Testcase3 have input and output files with same names and both are under same type of section i.e practice section. When someone clicks run practice test for assignment given in above figure, EvalPro dumps all test case files that are under section of type 'Practice' into a grading directory. Hence when a submission is executed against these test case, random behaviour is seen since multiple files exist with same file names. This was the reason behind observed irregular behaviour of EvalPro.

Solution to this problem is to give hierarchical name to test case files, so that no two files in different sections conflict. We have fixed the bug and input, output file names looks like these now -

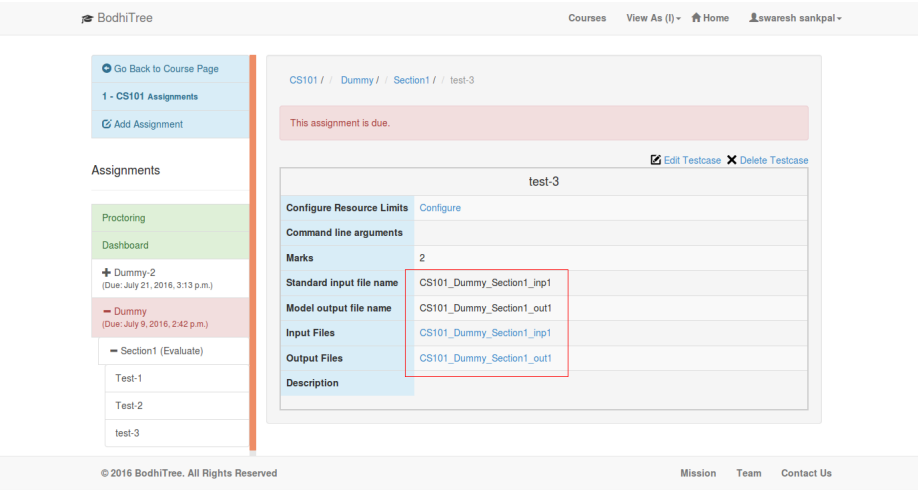



Figure 5.50: Test case file names after bug fix

### 5.15.2 Only Individual cribs should be shown

A small bug was found in data structures lab offered in Autumn 2016 at IIT Bombay. When students started using the crib feature of EvalPro to post their cribs related to a particular assignment, it was observed that all students were able to see cribs made by other students for the same assignment. We have fixed is issue. Now only the user can see the cribs that has been created by him but not others. Following are the screen shots that shows the cribs before the bug was fixed and after the bug was fixed.

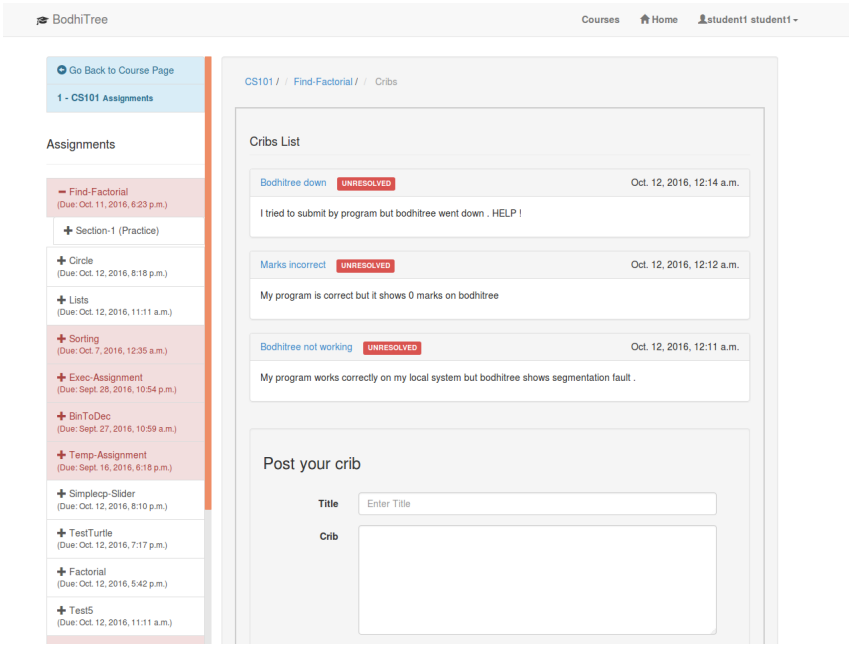



Figure 5.51: Bug in Cribs : A student is able to see cribs of all other students

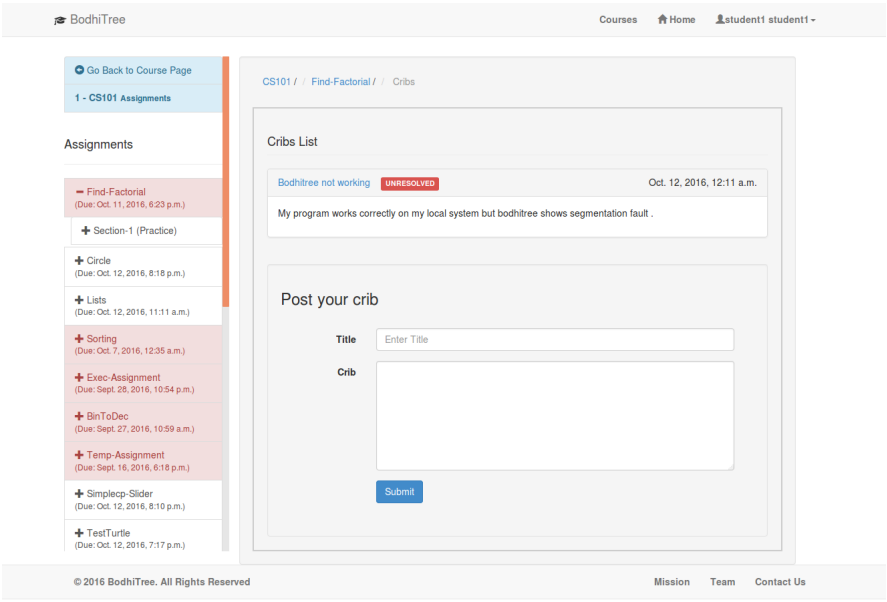



Figure 5.52: Cribs after bug fix

### 5.15.3 Create/Edit Assignment page modifications

Recently exam feature was added to EvalPro which requires parameters like IP address of students machines, duration of exam, exam group id and late duration which needs to be configured configured while creating an assignment for exam. So, these four form fields were added to create and edit assignment page to configure exam. There is a dropdown in create,edit assignment page to select the type of assignment that has to be created/edited i.e lab or exam. Irrespective of type of lab selected all these fields related to exam are always shown, which is not a good design. We have added a javascript function that hides/shows form fiels based on selection of type of lab. If exam is selected then only the form fields related to it i.e all four mentioned above will be displayed or else will be hidden. Also the if the instructor selects Non-programming check box, fields like programming language choice, execution time calculation, advance correctness checker, indentation calculation, bulk upload of test case etc are hidden using javascript. Also, the help text in form fields that was visible by default before, is hidden now and is shown once the particular field is hover as shown in second figure.

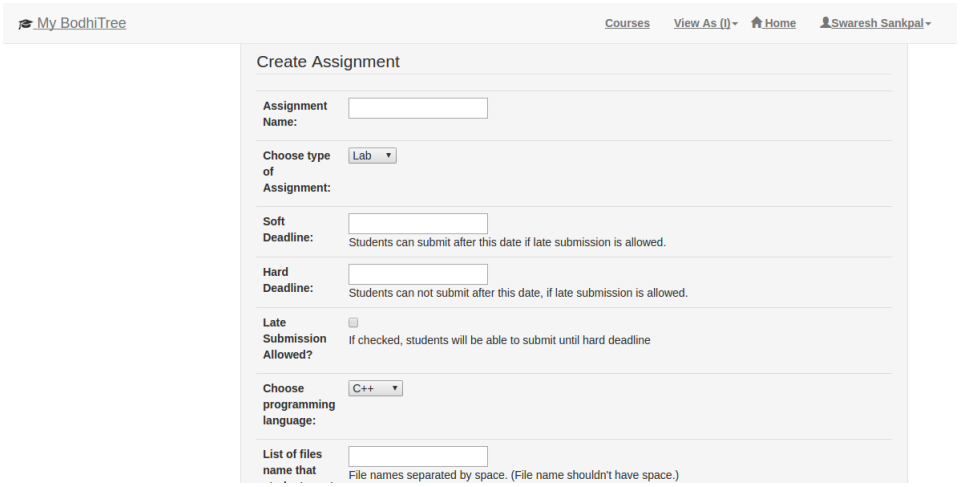



Figure 5.53: Create Assignment Page before modifications

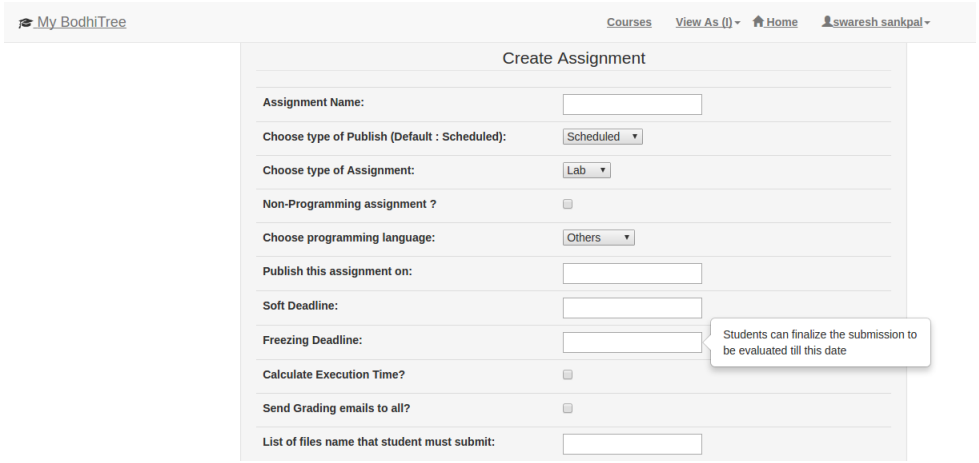



Figure 5.54: Create Assignment Page after modifications

### 5.15.4 Deadline alert feature

In EvalPro, every assignment has a deadline associated with it. It is observed that students tend to forget about the deadline and are busy with debugging their program. EvalPro does not allow students to upload their programs once deadline is over. Hence, it is very important to warn students when deadline is near. We have implemented a feature that warns student before 5 minutes of deadline. It informs the student which assignments are going to expire in next 5 minutes.

Implementation

We have written a middleware class which is called for every request. We have

written a function in this class which checks if there are any assignments that are going to expire in next 5 minutes. If such assignments are found a message is shown on current page saying that which all assignments are expiring in next 5 minutes.

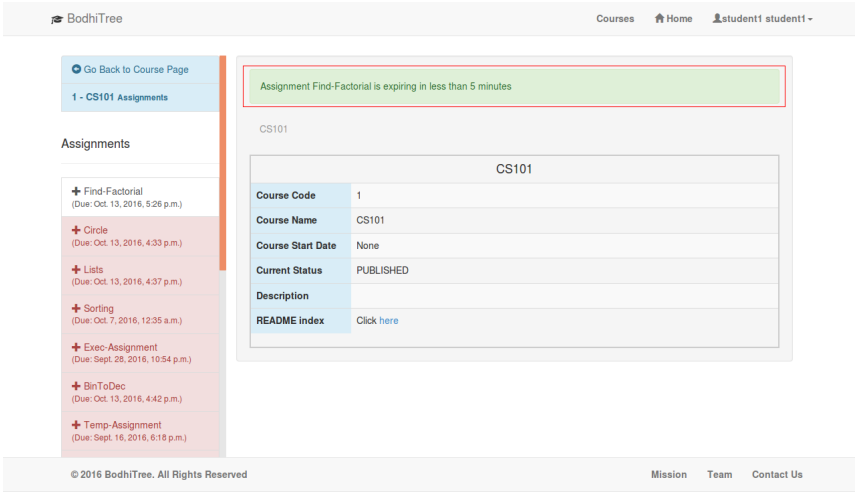



Figure 5.55: Deadline alert feature

# Chapter 6

# Future Work

The goal of our project is to add features to EvalPro such that the manual work of instructor and TAs is reduced, usability of students is increased and evaluation method becomes smarter and efficient. We have added some features that adds to EvalPro's functionality. In this section, we'll discuss about the work, we are going to do in next phase of this thesis project to make EvalPro a better auto evaluation tool than it is currently.

- Time complexity based evaluation
  Time complexity is a very big concern in advance computer science courses like Data Structures. For example, instructors asks students to implement merge sort in a lab, but some students since didn't knew or didn't want to implement merge sort and implemented bubble sorted instead. Execution time feature can be extended or some new logic has to be implemented in EvalPro to make it do evaluation based on time complexity of a program similar to what hacker rank has implemented.

- In-browser Editor improvements
  We have recently added code editor to EvalPro which has lot of advantages but it can be made better by making some improvements like

  - Displaying compilation error/ warnings alongside with code for faster debugging
  - Compilation of program should be independent from evaluate/ run practice test functionality. Because there will be a lot of compilation errors if students use in-browser editor and for every uncompiled code new file and object will be saved which is overhead in terms of disk space.

  - Only once compilation is successful, evaluate/ run practice test option should be visible
  - Option to provide custom input to program
  - Multiple language option to write a single program

- Exam Feature
  - Time duration of a student should start when he/she actually views the assignment and not the publish time given by bodhitree.
  - Timer running on Proctoring page for all students

- Adding one more hierarchy over assignments as LAB.
  In RA selections test 2017, there were 12 assignments created for programming test. The assignment deadlines were set before the exam started, but exam started late than the expected time and the deadlines were required to be adjusted for all assignments. As per current structure of assignment,instructor had to change the deadlines of all 12 assignments one by one which is a tedious work. There should a hierarchy over assignments which wouldclub them together and have common fields like deadline, publish time etc.

- Multiple language support in single assignment
  As per current architecture of Evalpro, one assignment supports a single programming language and if we want to give students multiple programming language option we have to create an assignment for each programming language. A single assignment should support multiple programming language

- Better User Interface
  EvalPro has improved a lot functionality wise, but the UI has remained the same. User interface is also a very important part of a Web application and hence has to be looked upon.

# Bibliography


[1] Mike Joy, *The boss online submission and assessment system.*. Journal on Educational Resources in Computing (JERIC),Volume 5 Issue 3, September 2005.

[2] Guillaume Derval, Anthony Gego, Pierre Reinbold, Benjamin Frantzen and Peter Van Roy. *Automatic grading of programming exercises in a MOOC using the INGInious platform.* Proceedings of the European MOOC Stakeholder Summit 2015

[3] D. M. de Souza,J. C. Maldonado and E. F. Barbosa *PROGTEST: An Environment for the Submission and Evaluation of Programming Assignments based on Testing Activities.* Software Engineering Education and Training (CSEE&T), 2011 24th IEEE-CS Conference.

[4] Codeboard
`https://codeboard.io/`

[5] X Window System
`https://en.wikipedia.org/wiki/X_Window_System`

[6] Safeexec : Safe Execution Environment
`https://github.com/ochko/safeexec`

[7] Codemirror
`https://codemirror.net`

[8] Difflib
`https://docs.python.org/2/library/difflib.html`

[9] Beautiful Soup
`https://www.crummy.com/software/BeautifulSoup/bs4/doc/`

[10] Celery
`http://docs.celeryproject.org/en/latest/getting-started/first-steps-with-celery.`

# Chapter 7

# Appendix

1. Execution and display of graphics programs through web application

   - Database
     assignments/models.py
     - graphics_program
   - Form
     assignments/forms.py
     - graphics_program
   - Templates
     assignments/templates/assignments/showTurtleTestcase.html (New)
     assignments/templates/assignments/details.html (Modified)
     upload/templates/upload/my_submissions.html (Modified)
     upload/templates/upload/allSubmissions.html (Modified)
   - Urls
     assignments/urls.py
     - evaluate/testcases/<submissionID>
     - evaluate/<submissionID>/<testcaseID>
   - Views
     evaluate/utils/evaluator.py
     - run_turtle
     - getResults_turtle

     evaluate/utils/executor.py
     - safe_execute_turtle
     - run_turtle

evaluate/views.py
- evaluate_turtle
- showTurtleTestcases
- evaluate_turtle_testcase

2. Execution time calculation of programs

- Database
  assignments/models.py
  - execution_time_allowed
  evaluate/models.py
  - AssignmentResults
    - get_execution_time (function)
  - ProgramResults
    - get_execution_time (function)
  - TestcaseResult
    - executiontime
  upload/models.py
  - executiontime
- Forms
  assignments/forms.py
  - execution_time_allowed
- Templates
  evaluate/templates/evaluate/practice_run_messages.html
  evaluate/templates/evaluate/test_run_messages.html
  upload/templates/upload/allSubmissions.html
- Views
  evaluate/utils/evaluator.py (Modified)
  sandbox/safeexec.c (Modified)

3. Assignment Trash

- Database
  assignments/models.py
  - trash
- Templates
  assignments/templates/assignments/assignments_base.html
  assignments/templates/assignments/showAssignmentsTrash.html

- Urls
  assignments/urls.py
    - assignments/trash/<courseID>
    - assignments/restore/<assignmentID>
- Views
  assignments/views.py
  upload/views.py

4. Assignment Repository

- Database
  assignments/models.py
    - backup
- Templates
  assignments/templates/assignments/assignments_base.html
  assignments/templates/assignments/createAssignment.html
  assignments/templates/assignments/details.html
  assignments/templates/assignments/selectAssignments.html
  assignments/templates/assignments/showRepository.html
  assignments/templates/assignments/viewAssignmentDetails.html
- Urls
  assignments/urls.py
  assignments/save/<assignmentID>
  assignments/select/<courseID>
  assignments/paste/<assignmentID>/<courseID>
  assignments/repository/<courseID>
  assignments/view/details/<assignmentID>
- Views
  assignments/views.py
    - paste_assignment
    - view_assignmentdetails
    - show_repository
    - select_assignment
    - save_assignment

5. Manual grading of an submission

- Database
  upload/models.py
    - comments
    - final_marks
- Templates
  upload/templates/upload/allSubmissions.html
  courseware/templates/student/pastsubmissions.html
  elearning_academy/templates/base.html
  elearning_academy/templates/elearning_academy/logged_in.html
- Urls
  upload/urls.py - edit/marks
    - edit/comments
  courseware/urls.py
    - pastsubmission/<studentID>
- Views
  upload/views.py
    - edit_submission_comments
    - edit_submission_marks
  courseware/views.py
    - pastsubmissions

6. Feature to map students and TA's together

- Database
  assignments/models.py
    - type_of_allocation
    - ta_allocation_document
    - previous_allocation_policy
    - TA_allocation (Class)
      - assignment
      - student
      - assistant
- Forms
  assignments/forms.py
    - isenrolledin_course (Function)
    - AssignmentModelChoiceField (Class)

- AssignmentForm
    - type_of_allocation
    - ta_allocation_document
    - previous_allocation_policy
    - check_ta_allocation_document (Function)

- Templates
  assignments/templates/assignments/createAssignment.html
  assignments/templates/assignments/details.html
  assignments/templates/assignments/edit.html
  upload/templates/upload/allSubmissions.html
- Urls
  upload/urls.py
- Views
  assignments/views.py
    - perform_ta_allocation

  crib_management/views.py
  upload/views.py

7. Archive student's intermediate submissions

- Database
  assignments/models.py
    - Assignment
    - freezing_deadline
    - hard_deadline (Deleted)
    - late_submission_allowed (Deleted)
    - deletesubmissions_task_id

  upload/models.py
    - LatestSubmission
        - assignment
        - owner
        - submission
- Forms
  assignments/forms.py
    - AssignmentForm
        - freezing_deadline

- hard_deadline (Deleted)
- late_submission_allowed (Deleted)

- Templates
  assignments/templates/assignments/createAssignment.html
  assignments/templates/assignments/details.html
  assignments/templates/assignments/edit.html
  assignments/templates/assignments/readmeAssignmentMetaInterface.html
  upload/templates/upload/my_submissions.html

- Urls
  upload/urls.py
  - mysubmissions/<assignmentID>
  - freeze/<submissionID>

- Views
  assignments/tasks.py
  - delete_redundant_files
  - deleteSubmission

  assignments/views.py
  evaluate/views.py
  upload/views.py
  - freeze_submission

8. In-Browser Code Editor

- Forms
  assignments/forms.py
  - class EditorForm

- Templates
  assignments/templates/assignments/details.html
  assignments/templates/assignments/editor.html
  elearning_academy/static/css/eclipse.css
  elearning_academy/static/css/mdn-like.css

- Urls
  assignments/urls.py
  - publish/<assignmentID>
  - editor/<assignmentID>

- Views
  assignments/views.py
  - online_editor
  elearning_academy/settings.py
  - codemirror2
  requirements.txt

9. Feature to compare output files visually

   - Forms
     assignments/forms.py
   - Templates
     assignments/templates/assignments/details.html
     evaluate/templates/evaluate/fileComparison.html
     evaluate/templates/evaluate/practice_results.html
     evaluate/templates/evaluate/results.html
   - Urls
     evaluate/urls.py
     - compare/<testcaseresultID>
   - Views
     assignments/views.py
     evaluate/views.py
     - compare_files

10. Feature to calculate indentation percentage of a program

    - Database
      upload/models.py
      - indentation_calculated
      - indentation
      assignments/models.py
      - indentation
    - Forms
      assignments/forms.py
      - indentation
    - Templates
      evaluate/templates/evaluate/result_table.html
      upload/templates/upload/allSubmissions.html

- Views
  upload/views.py
    - get_indentation
    - indentation_helper
    - remove_comments

11. Bulk Download

    - Templates
      assignments/templates/assignments/index.html
      assignments/templates/assignments/details.html
    - Urls
      download/urls.py
        - assignment/data/<assignmentID>
        - course/data/<courseID>
    - Views
      download/views.py
        - download_assignment_files
        - download_course_files

12. Feature to publish/ unpublish solution code on demand

    - Database
      assignments/models.py
        - model_solution_published
    - Forms
      assignments/forms.py
    - Templates
      assignments/templates/assignments/details.html
    - Urls
      assignments/urls.py
        - publish/<asignmentID>
    - Views
      assignments/views.py
        - modelsolution_changestatus

13. Feature to extend EvalPro's functionality for Non-Programming assignments

    - Database
      assignments/models.py
          - only_uploads
    - Forms
      assignments/forms.py
          - only_uploads
      upload/forms.py
    - Templates
      assignments/templates/assignments/createAssignment.html
      assignments/templates/assignments/details.html
      assignments/templates/assignments/edit.html
      upload/templates/upload/allSubmissions.html
    - Views
      assignments/views.py
      utils/filetypes.py