

Spring 2023

## CodEval

Aditi Agrawal  
*San Jose State University*

Follow this and additional works at: [https://scholarworks.sjsu.edu/etd\\_projects](https://scholarworks.sjsu.edu/etd_projects)



Part of the [Artificial Intelligence and Robotics Commons](#), and the [Other Computer Sciences Commons](#)

---

### Recommended Citation

Agrawal, Aditi, "CodEval" (2023). *Master's Projects*. 1224.

DOI: <https://doi.org/10.31979/etd.kj7h-7xxh>

[https://scholarworks.sjsu.edu/etd\\_projects/1224](https://scholarworks.sjsu.edu/etd_projects/1224)

This Master's Project is brought to you for free and open access by the Master's Theses and Graduate Research at SJSU ScholarWorks. It has been accepted for inclusion in Master's Projects by an authorized administrator of SJSU ScholarWorks. For more information, please contact [scholarworks@sjsu.edu](mailto:scholarworks@sjsu.edu).

# CodEval

A Project Report

Presented to

The Department of Computer Science

San José State University

In Partial Fulfillment

of the Requirements of the Class

CS 298

By

Aditi Agrawal

May 2023

© 2023

Aditi Agrawal

ALL RIGHTS RESERVED

The Designated Project Committee Approves the Project Titled

CodEval

by

Aditi Agrawal

Approved for the Department of Computer Science

San Jose State University

May 2023

Dr. Benjamin Reed

Department of Computer Science

Dr. Faranak Abri

Department of Computer Science

Dr. Thomas Austin

Department of Computer Science

## **ABSTRACT**

Grading coding assignments call for a lot of work. There are numerous aspects of the code that need to be checked, such as compilation errors, runtime errors, the number of test cases passed or failed, and plagiarism. Automated grading tools for programming assignments can be used to help instructors and graders in evaluating the programming assignments quickly and easily. Creating the assignment on Canvas is again a time taking process and can be automated. We developed CodEval, which instantly grades the student assignment submitted on Canvas and provides feedback to the students. It also uploads, creates, and edits assignments, thereby making the whole experience streamlined and quick for instructors and students. It is simple to use, easily integrated with the learning management system, and has a low learning curve. This report shows the background, implementation, and results of using CodEval for programming courses.

## **ACKNOWLEDGMENT**

I extend my gratitude to Professor Reed for consistently guiding and being part of this project. Archit Jain for his contribution to this project, and finally Department of Computer Science for their resources and support for the opportunities that came up with this project.

I would also like to thank the committee member, Dr. Faranak Abri and Dr. Thomas Austin, for their guidance.

# TABLE OF CONTENTS

<i>ABSTRACT</i> .....	<i>i</i>
<i>ACKNOWLEDGMENT</i> .....	<i>ii</i>
<i>I. INTRODUCTION</i> .....	<i>1</i>
<i>II. RELATED WORK</i> .....	<i>4</i>
<i>III. SYSTEM DESIGN</i> .....	<i>11</i>
3.1 Grading Submission .....	12
3.1.1 Specification Files.....	13
3.1.2 CodEval Configurations .....	15
3.1.3 Running CodEval .....	16
3.1.4 Feedback Method .....	17
3.1.5 Supporting Multiple Languages .....	19
3.2 Creating Assignments .....	19
3.2.1 Uploading Files.....	20
3.2.2 Converting Text to Markdown .....	22
3.2.3 Creating and Editing Assignments .....	24
3.2.4 Creating Discussion Topic and Linking .....	25
3.2.5 Command Line Arguments .....	26
<i>IV. RESULTS</i> .....	<i>29</i>
4.1 Evaluation of Submissions .....	29

4.1.2 Student Evaluations .....	30
4.2 Assignment Creation .....	32
V. FUTURE WORK .....	36
5.1 Adding Support for Python .....	36
5.2 Enhancing Readability of Output .....	36
5.3 Convert the bash code to Python .....	37
5.4 Add Support for Plagiarism Check .....	37
5.5 Integrating CodEval with other LMS .....	37
VI. CONCLUSION .....	38
REFERENCES .....	39



## LIST OF TABLES

Table 1: CodEval Specification Tags .....	15
Table 2: Survey Results .....	30
Table 3: Comparing assignments with and without usage of CodEval.....	32

## LIST OF FIGURES

Figure 1: System Architecture.....	11
Figure 2: CodEval Specification file .....	13
Figure 3: CodEval Configuration File .....	16
Figure 4: parsediff Output.....	18
Figure 5: parsevalgrind Output.....	18
Figure 6: Addition of HINT tag in specification file .....	19
Figure 7: Uploading the files manually .....	20
Figure 8: assignmentFiles folder and required files in it.....	21
Figure 9: Files uploaded on Canvas.....	21
Figure 10: CRT_HW START tag in the specification file .....	22
Figure 11: CRT_HW END tag in the specification file.....	22
Figure 12: EXMPLS tag in the specification file.....	23
Figure 13: URL_OF_HW in the specification file .....	24
Figure 14: DISCSN_URL tag in the specification file.....	26
Figure 15: Example of creating commands using click [63].....	27
Figure 16: Command line arguments options in CodEval.....	28
Figure 17: grade-submissions command's options .....	28
Figure 18: Example of a submission evaluation by CodEval .....	29
Figure 19: Html format of the description.....	32
Figure 20: URL_OF_HW tag has been replaced with the file URL .....	33
Figure 21: Created Assignment on Canvas .....	33
Figure 22: Examples in the description.....	34

Figure 23: Referenced files and discussion topic.....35

Figure 24: Created Discussion topic, with reference to the assignment.....35

## LIST OF ABBREVIATIONS

LMS – Learning Management Systems

UML – Unified Modelling Language

API – Application Programming Interface

XML – Extensible Markup Language

RPC – Remote Procedure Call

SQL – Structured Query Language

REST – Representation State Transfer

HTML – HyperText Markup Language

XML – Extensible Markup Language

JSON – JavaScript Object Notation

PEML – Programming Exercise Markup Language

AML – ArchieML

YAML – YAML Ain't Markup Language

CMS – Content Management System

SGML – Standard Generalized Markup Language

GFM – GitHub Flavoured Markdown

URL – Unified Resource Locator

## I. INTRODUCTION

Advancements in technology, easy access to the internet, and the pandemic have led to the shift to virtual platforms in academia. Streaming classes, taking exams, submitting assignments, and providing grades all take place through learning management systems (LMS) such as Canvas [1], Moodle [2], and Codelab [3]. Creating content for the classes is already a laborious task and on top of it evaluating the assignments, especially coding assignments for class sizes varying from 20-80, and one assignment per week for a four-month long semester is cumbersome.

Grading the coding assignments can be done by leveraging automated grading tools for programming assignments, thereby reducing the workload on an instructor or grader. There are a number of tools in the market, to name a few, ASSYST [4], JavaAssess [5], MarkUs [6], and Submittly [7]. These tools are either standalone or integrated with LMS, either way, an assignment is submitted on these tools and evaluated based on numerous factors such as compilation errors, run-time errors, the number of test cases passed or failed so on and so forth and the result is either graded and displayed to the user on the tool or is used by the grader or instructor to grade the assignment manually and later provide the final grades to the student.

The advantage of using such tools is that they provide instant feedback to the student and perform much of the quantitative check of the student code which in turn reduces the burden on the instructor or grader. Another benefit is that when compared with manual grading of programming assignments, they are not error-prone and consume much less time to evaluate the assignment thoroughly.

CodEval [8] is one such automated grading tool for programming assignments that are integrated with LMS Canvas [1]. It provides instant feedback to the students, and supports multiple languages such as Java, and C. It has its own specification format which is simple and has a low learning curve and supports multiple programming languages with minimal changes and can be used for creating an assignment as well as evaluating the assignment on Canvas[1] or any other LMS.

San Jose State University uses Canvas to manage assignments and grade them. For programming assignments, a student must upload a zip file of the code on Canvas [1] and wait for it to be evaluated. Grader usually waits until after the deadline to start the grading. The grader or instructor must download the zip file, unzip it on their own environment, setup the environment, compile the code which can be malicious due to a malicious code, run it against the test cases, check for additional requirements for the assignment and then grade the assignment and upload the grades and feedback of each student on Canvas.

As one can see, this is time taking, and this aspect of programming assignment can be automated therefore CodEval has been developed. CodEval [8], automates downloading the zip file, running the code on the Docker instance, running it against the specification file, and providing instant feedback to the student in the comment section of Canvas. In this way the initial evaluation of the code is done by the tool and the grader or instructor can go through the feedback provided by CodEval and provide final grades to the student.

The first implementation was finished in the Spring of 2022, which included the features mentioned above. As part of the fall of 2022, a couple of enhancements features such as providing hints for failed test cases and providing more readable feedback of failures to students in the comment section have been implemented. Other enhancement features

such as checking for plagiarism and providing support for using the specification file for creating the assignment on Canvas will be implemented.

CodEval [8] is already being used in Operating System and Algorithm undergrad classes at SJSU and has shown positive results by catching the errors at an early stage and giving students a chance to rework their code and submit it multiple times to get it all right by the deadline. Student's feedback has also been positive showing improvement in coding skills and confidence.

Section 2 will go over the related work and background. Section 3 describes the system design for CodEval. Section 5 displays the results and Section 6 concludes the report followed by references.

## II. RELATED WORK

Although C++ and Java are supported by most tools, other languages are now being supported by some tools recently [9]. The most recent automated grading tools on the market have been compared and examined by Aldriye et al. [10] and J Caiza et al. in [11]. They emphasize the distinctions, benefits, and advantages of the various systems and have paved the road for potential future developments in automated grading tools. Assignments in the programming languages Java, C++, and Scala are graded using a variety of tools, including CourseMaker created by Nottingham University[12], JavaBrat created at San Jose State University[13], WebCat, and RoboLift(an incremental work on WebCat).

CourseMaker [12] verifies the student code against specified test cases and examines typographic consistency as part of their evaluation criteria (indentations, comments, layout of the file). In contrast to JavaBrat[13] and Virtual Programming Lab[15], which grade based on code correctness, which is determined by validating the output against the output from test cases, WebCat[14] checks code correctness (how many tests pass), test completeness (check which parts of the code are actually executed), and test validity (test accurate-consistent with the assignment). While some programs merely consider whether test cases passed or failed, others, like CourseMaker, also consider extra factors like the soundness of the code design.

CodEval uses the second approach and has additional comprehensive checks for the code. The effects of automated grading technologies on students' opinions of them as well as their performance were examined by A. Gordillo [16]. According to the results, adding the automated evaluation tool to the curriculum helped students by strengthening their



motivation, the standard of their work, and their practical programming abilities. We'll demonstrate how CodEval yields comparable outcomes.

In [11], the author notes that while most automated grading solutions, such as CourseMaker [12], WebCat [14], and Marmoset [17], are freestanding, the next step for these technologies would be integration with LMSs. For instance, in order to utilize CourseMaker, users must install the client on their computer. Similar to CodEval in terms of integration with LMSs, there are other tools as well, like Virtual Programming Lab, JAssess, and JavaBrat that can be connected with LMSs or offer plugins with platforms like Moodle.

The Magdeburg University's grading tool [18] is independent of any specific LMS. In order to evaluate a programming language, it takes into account three servers: the front-end, an LMS system, and the spooler server, which manages the request, submissions queues, and back-end calls. XML-RPC (Remote Procedure Calls) has been utilized to talk to the servers. It's a good idea to make the automatic grading tool independent of an LMS. It is also possible to utilize CodEval independently with different LMSs. It takes advantage of the LMS's APIs rather than any explicit procedure calls to communicate. Although CodEval makes use of pre-existing Docker containers rather than needing the setup of individual servers, it nevertheless supports a variety of languages and environments.

A software product's security is an essential component. The preservation of the submissions and safeguarding the host server from any malicious attacks from any submission are essential in automated grading solutions. Marmoset [17] runs its J2EE web server and SQL database in a separate environment to guard against the effects of malicious programming. WebCat [14] has modified architectural concepts to address authentication issues and illegal activity.

Using sandbox settings to test security measures is a more comprehensive strategy. CourseWork runs its code in a sandbox environment. The use of containers offers two benefits. They give the host server security and a tailored, spotless environment in which to execute the tests. For its container environment, CodEval uses Docker, however, other environments can be utilized by altering the setup options.

Instead of providing comments with failing test cases, Liu et al. AutoGrader's program [19] looks for semantically distinct execution paths between a student's submission and the reference implementation. To completely automate the process of analyzing a submission based on semantics, the goal of this study is to decrease the amount of manual labor necessary to generate test cases. By defining unique test instructions in the evaluation specification, CodEval enables the integration of these tools for evaluation.

Most studies on automated grading tools for programming assignments conducted to date have divided and contrasted the tools into categories such as programming languages, dynamic analysis, and static analysis. K.A. Mukta conducted a market analysis of the automated grading technologies available up till 2005 [20]. She researched and listed the characteristics that these programming tools could use to analyze automatically. The use of a sandbox environment for security purposes, testing the code's functionality and efficiency against predefined test cases, catching programming errors like runtime and compilation errors, running student-defined test cases, coding style, programming language-specific issues like memory leak check for C, and special features were some of the features mentioned (certain libraries are used or not).

Building on similar work, P. Ihantola et al. [21] and Lian et al. [22] covered the automated grading tools until 2010. Other than evaluating them on the same features as done

in [20], [22] has also compared the tools based on the platforms they are offered on, such as whether the tool is a standalone tool or integrated with LMS such as CodEval[8], AutoGrader [23], and on resubmission criteria which includes limiting the number of submissions, penalty on late submission, limiting the amount of feedback provided for instance in [8], and hybrid approaches such as in Marmoset[17], where the limit of submissions is optional and set by the instructor.

Like the surveys previously mentioned, the authors in [24] evaluate the tools until 2021 based on additional criteria including the CS domains supported and the data provided to a teacher to enhance student learning. They thoroughly divided the tools into various computing-related fields, including visual programming, system administration (LINSIM[25]), formal language and automata (JFLAP[26]), web development, software modeling (by comparing class diagrams and other UML elements), parallel computing (SAUCE[27]), and software testing, which includes assessing the code base against test cases.

Some surveys evaluated the tools based on the instructional element in addition to evaluating the technical distinctions and concepts. [20],[28],[29] have contrasted and compared grading tools for automatic assessment (GAME[30]), semi-automatic assessment (JACKSON[31]), and hybrid assessment, where some static and dynamic analysis is performed by the tool and the qualitative analysis, such as code style, and deciding which grade to award is performed by the instructor or the grader. (Webcat[32] and MOSSHAK[33]).

Separating the tools based on a formative versus summative approach was another criterion utilized in [20]. In CourseMaker[34] and Codeval [8], the author explains the formative technique as one in which the student can submit the assignment more than once

to assist them in developing their programming skills. In contrast, a summative approach—like BOSS[35,36]—allows the learner to submit the code just once.

There are surveys on various aspects of automated grading technologies in addition to technical and pedagogical evaluation. J.R. Alamo experimented with his students in [37] by introducing a variety of online assessment tools (Programmr [38], Code Step by Step [39]) each fall semester. He then gathered input from the students to determine whether the tools had improved their grades.

R. Queiros and J.P. Leal examined an intriguing element of automated tools in [40], where they assessed the tools' capacity for interoperability. They examined the interoperability of 15 tools in terms of programming exercises, users, and test outcomes. They came to the conclusion that more than 50% of these instruments lacked sophisticated interoperability, thus attention must be paid to this market. Despite the quality of the literature and surveys on automated grading tools, none of them have thoroughly examined the assessment criteria of the tools now on the market.

To format a document, assignment, or web page there are a number of markup languages and respective parsers used on websites and desktop applications. Some of the most popular markup languages are Hyper Text Markup Languages (HTML) [41], Extensible Markup Language (XML) [42], and JavaScript Object Notation (JSON) [43]. HTML and XML are used to display the data in a standard format on web pages and JSON is a data-interchange format used for programs. Some of the other data-oriented languages are YAML Ain't Markup Language (YAML) [44], Programming Exercise Markup Language (PEML) [45], and ArchieML (AML) [46]. YAML is used in applications where data is stored and transmitted, in configuration files, and is like JSON.

Programming Exercise Markup Language (PEML), which is a specific data format for creating coding assignments. In [45], the author claims that the language is simple and has less learning curve compared to YAML or JSON and therefore is better than to be used for programming assignments. Another example is ArchieML, which also has a simpler format to edit and structure a document to be rendered on the web. In [46], explains the subtle difference of AML when compared to YAML or JSON, which gives less significance to whitespace, and syntax that can be learned and used by non-programmers.

As mentioned in Standard Generalized Markup language (SGML) [47], a markup language contains the format of how the document should look like and not the process of creating the document, therefore parsers are required to convert the data from these markup languages to entities that can be either rendered as a web page or can be read as input to a program. A number of desktop applications and websites support the usage of these data formats by having parsers or plugins to translate the data such as GitHub [48], Kattis, and Canvas.

Programming languages such as Python, Ruby, R, Java, CPP, and C# have libraries to parse these markup language data such as the 'json' library in CPP to read JSON objects or the 'XML' library in Python to read XML objects. Yet another markup language whose name came be misleading is Markdown language, which is a markup language to create documents and is different from other markup languages as it does not contain tags and is simple to read and edit.

In [49], the author mentions the advantages of using the Markdown language, some of which are, it can be used to create documents, email, websites, and technical documentation, it can be opened in any application, it can be created on any Operating

system, and even if the application using it ceases to exist, the Markdown-formatted document can be read by any other text editor applications as well.

Markdown is popular and is supported by multiple websites such as GitHub, Discord, Microsoft Teams, and RStudio [50]. The parser for Markdown is a Perl script named `markdown.pl` [49], which converts the data in a markdown file to HTML or XHTML, which can be rendered as web pages. There are third-party parsers that convert the Markdown formatted data to HTML or pdf, some of them are `iaWriter` [51], `ghostwriter` [52], `Markdown Monster` [53], `ReText` [54], and `StackEdit` [55].

There are other flavors of Markdown as well in the market. GitHub-flavored Markdown (GFM) [56] supports table contents, nesting block contents under lists, and some GitHub features such as references to commits, on top of the basic Markdown features. Similarly, `Markdown Extra` is implemented in Python and used by content management systems (CMS) such as `Drupal` [57] and `TYPO3` [58]. The additional features are supported by `Markdown Extra` are tables, footnotes, abbreviations, and multiple lines of code. Due to the simplicity of Markdown, it is used in `CodEval` to create assignments.

### III. SYSTEM DESIGN

CodEval automates grading the coding submissions and creating the submission on Canvas. Grading the submission comprises of downloading the assignment from Canvas, starting the Docker instance, running each submission against the specification file and displaying the feedback in the comments section on Canvas. On the other hand, creating the assignment comprises uploading the required files, converting the text to HTML, creating the assignment, and creating a discussion topic. Other than providing instant feedback and automatic creation of assignments, another design advantage of using CodEval is that, all the states of the CodEval reside on Canvas and none on the instructor's server. This design advantage makes installing and shifting CodEval to other servers seamless and effortless.

Figure 1, depicts the system architecture of CodEval.

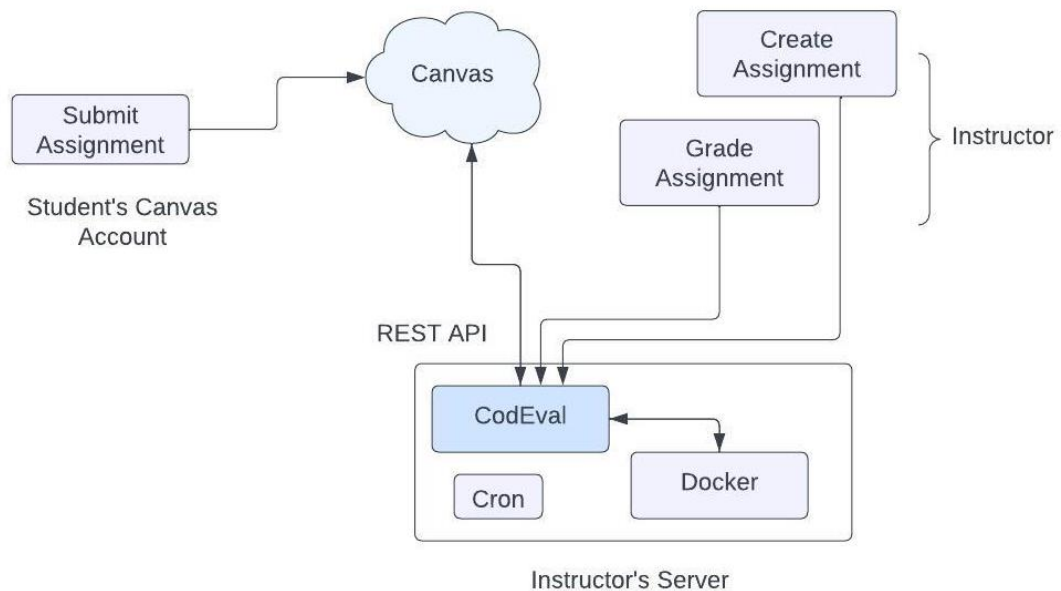


Figure 1: System Architecture

### **3.1 Grading Submission**

CodEval connects to Canvas using the Canvas REST API while running locally on a computer. CodEval can download contributions and annotate them using this API. Additionally, we keep specification files and other supporting files for testing and reviewing the submissions using the Canvas File service. CodEval will only be able to access courses and content that the user running CodEval is authorized to view since Canvas employs API tokens to restrict access to Canvas features. This makes sure that a student cannot access the specifications and submissions of other students.

CodEval leverages Docker to execute the compilation and test cases in a container to isolate the student code from the machine conducting the test after downloading a submission and the specification files that specify how to assess the submission. CodEval can perform the evaluations outside of Docker, however, doing so is not advised to prevent malicious or buggy submissions from harming the computer running CodEval. After the code has been assessed, CodEval will use the Canvas REST API to submit the result as a remark to the submission in Canvas.

CodEval does not require any special Canvas configuration to work. It uses the REST API that instructors can enable by creating an access token. Instructors upload all the specification files and support files to Canvas so that they only need to interact with the Canvas interface once while creating CodEval assignments. Once CodEval is running on a server using the provided tokens, instructors, students, and graders do everything through Canvas.



### 3.1.1 Specification Files

The CodEval specs files were made to be simple to produce and read. Additionally, we needed them to be adaptable enough to take both our straightforward programming tasks and our more complicated ones. The tag, a space, and the data associated with the tag are the first three characters on each line of the file. For instance, Figure 2 displays a very basic configuration file that executes the well-known 'hello world' Java application. In that illustration, the script to use to analyze the specification file is specified on the first line that begins with the RUN tag.

```
RUN evaluate.sh
C javac edu/sjsu/CS001/Hello.java

T java edu.sjsu.CS001.Hello ben
X 0
O Hello ben

T java edu.sjsu.CS001.Hello
X 1
O USAGE: edu.sjsu.CS001.Hello name

HT java edu.sjsu.CS001.Hello "long name here!"
X 0
O Hello long name here!
```

Figure 2: CodEval Specification file

The tag C tells CodEval to run the compilation command adjacent to it. T tag stands for the test cases that need to be run, and O is the expected output of that test case which is compared with the student's output for that test case. X tag is for the exit code of the code while executing that particular test case. If the output is of multiple lines then each output

line is prefixed with O. The last test case in the figure starts with HT which stands for the hidden test case. If this test case fails then the input of the test case is not displayed, but a hint is still provided. Normal test cases will show hints as well as the input of that test case in case of failures. Table 1 lists all the tags supported by the CodEval specification file.

Tag	Meaning	Function
RUN	Run Script	Specifies the script to use to evaluate the specification file. Defaults to evaluate.sh.
Z	Download zip	Will be followed by zip files to download from Canvas to use when running the test cases.
CF	Check Function	Will be followed by a function name and a list of files to check to ensure that the function is used by one of those files.
CMD/TCMD	Run command	Will be followed by a command to run. The TCMD will cause the evaluation to fail if the command exits with an error.
CMP	Compare	Will be followed by two files to compare.
T/HT	Test Case	Will be followed by the command to run to test the submission.
I/IF	Supply Input	Specifies the input for a test case. The IF version will read the input from a file.
O/OF	Check output	Specifies the expected output for a test case. The OF version will read from a file.
E	Check error	Specifies the expected error output for a test case.
TO	Timeout	Specifies the time limit in seconds for a test case to run. Defaults to 20 seconds.
X	Exit code	Specifies the expected exit code for a test case. Defaults to zero.
HINT	Hint	Provides hints to the hidden test cases.
CRT_HW START	Start description	Depicts the beginning of the description of the assignment. It is followed by the assignment name.
CRT_HW END	End description	Depicts the end of the description of the assignment.

EXMPLS	Examples	Examples in the description
URL_OF_HW	File URL	URL of the file is linked.
DISCSN_URL	Discussion Topic URL	URL of the Discussion Topic is linked.

Table 1: CodEval Specification Tags [71].

One of the evaluation criteria that CodEval performs is checking if the required function is used or not in the code. The grader eventually must verify if the function was correctly used or not, but the check failing or passing during CodEval evaluation can help remind the student that he/she has missed using the function or has used it although it was not required to be used. For instance, the second assignment in the Operating Systems course focused on using `fork()`, therefore checking submissions for `fork()` served as a reminder to students who had forgotten to include `fork()`. It also gave the grader a hint about a potential issue.

Note that the grader still needed to ensure that `fork()` was used appropriately even if the submission did use it. The `grep` command was not as simple as just `grep fork main.c`, so we added the CF tag, which allows us to specify `CF fork main.c`, to check for the use of functions at first. We started using CodEval before the CF, TCMD, IF, and OF tags were included. In each instance, we only needed a few lines of script and a few minutes to add the tag and get it working. CF command has only four lines of code.

### 3.1.2 CodEval Configurations

While configuring CodEval, Canvas and Docker both are configured. Figure 3 below shows the Canvas configuration. The SERVER section has the URL details of the Canvas

instance running on the SJSU server. URL for SJSU: <https://sjsu.instructure.com>. Following this tag it contains token which contains Canvas API token corresponding to the user running CodEval. The RUN section contains the commands to run the evaluation of the submission. *SUBMISSIONS* and *EVALUATION* are substituted with the path of the downloaded submission and the command to evaluate the submission. The command starts with docker as CodEval starts a *docker* instance, sets up the environment in docker, and then runs the code in that instance.

```
[SERVER]
url=https://sjsu.instructure.com

token=XXXX
[RUN]
precommand=
command=docker run -i -v SUBMISSIONS:/submissions jimg bash -c "cd /submissions; EVALUATE"
```

Figure 3: CodEval Configuration File

### 3.1.3 Running CodEval

The name of the course to be evaluated is passed as a runtime parameter to CodEval. It will look for any contributions that lack a CodEval remark by scanning all the assignments with related specification files in the Canvas Files portion of Canvas. These submissions will be downloaded and evaluated in accordance with the CodEval specification. We utilize the Linux *cron* function to execute CodEval every five minutes since we need submissions to be assessed within five minutes after submission.

The CodEval Python script connects to Canvas using the Canvas REST APIs, creating the temporary folder, downloading the submissions, and storing the submissions in the temporary folder. CodEval builds a Docker instance for each student's every submission,

compiles and runs the student's code, and compares the student's output with the expected output mentioned in the specification file. The output from the comparison is sent back as feedback to the comment section on Canvas using Canvas REST API.

### 3.1.4 Feedback Method

Both static and dynamic analyses of code are performed by CodEval. Static analysis ensures the software compiles correctly and confirms the presence of particular functions or commands. The *CMD* or *TCMD* commands can be used to carry out ad-hoc static analysis. The specification file's test cases are executed against the compiled code as part of the dynamic analysis, which then shows whether the test case passed or failed. The command for the failed test case will be displayed if the test case has failed. The *diff* tool is used to compile errors brought on by unexpected output. Dynamic analysis can also use runtime analyzers to verify the accuracy, such as *valgrind* [59].

After receiving feedback from students on CodEval, the readability experience of students was improved. As part of this step, the result from the *diff* tool and *valgrind* [59] were further processed by new parsers namely *parsediff* and *parsevalgrind* to parse the result from *diff* and *valgrind* and convert it to a more easily comprehensible format.

*parsediff* parses the output of *diff* of expected output and student's output and displays the lines that are needed to be added in the student's output and also the lines that need to be not present in the student's output. These lines are prefixed with '+' and '-' respectively. The parsed output is sent to the *logOfDiff* file in the *evaluationLogs* folder. The snippet of the output is given below in Figure 4.

```

[aditi@cs-reed-02 CodEval]$ [aditi@cs-reed-02 CodEval]$ ./evaluate.sh
used strtol PASSED

Test Case 1 of 10: FAILED
  Command ran: ./addup 17 3 42 8
The lines prefixed with '-' are wrong output lines present in your output and not present in the expected output.
The lines prefixed with '+' are present in the expected output and should be present in your output.
Your output file: ./youroutput^
Expected output file: ./expectedoutput^
-70 output$
+70$

```

Figure 4: parsediff Output

Similarly, *parsevalgrind* takes the XML formatted output of *valgrind* and parses its respective tags to read the exact error statement, the hierarchy of functions where the error has occurred, the line in each function, and the type of error for each error in the code and print it out so that students can better understand where the memory leak has taken place. *parsevalgrind* sends its output to the log file *log\_of\_valgrind* in the evaluation folder. *codeval.py* file reads the logs from this folder and displays them in the comment section on Canvas. The output of the parser is given below in figure 5.

```

FAIL!!!! YOU HAVE A MEMORY PROBLEM
FAILED
In file: vg_replace_malloc.c .
In function: malloc at line number: 309
In function: main at line number: 309
Type of Error: Leak_DefinitelyLost
Error: 40bytes in 1 blocks are definitely lost in loss record 1 of 1

```

Figure 5: parsevalgrind Output

Another enhancement to CodEval is adding the hints feature for failed test cases. The HINT tag has been added in the specification format which is present for hidden test cases and when these test cases failed, it displays the corresponding hint for that test case. This tag is not added for the other test cases as for those, the input is displayed when the specific test case fails. Adding hints has helped students debug the code faster and thereby increasing

their motivation to attempt the assignment. An example of the specification file is presented below in figure 6.

```
T ./addup 1 20 300 4000 50000 600000 7000000
O 7654321

HT ./addup -3 4 -8
HINT try some negativity.
O -7

HT ./addup 0 -1 2 -3 2
HINT sometimes you are negative, sometimes you are positive.
O 0

HT ./addup 0
HINT 0+0=?
O 0
```

Figure 6: Addition of HINT tag in specification file

### 3.1.5 Supporting Multiple Languages

As of now CodEval supports C and Java but is able to support other languages such as Python. In order to support multiple languages two things, need to be taken care of, firstly installing the environment for code compilation in the Docker instance, which is a one-time work, and secondly, changing the command tag 'C' in the specification file suiting the specific language. For instance, for C language assignments the value for tag 'C' is GCC keyword followed by flags and the *file\_name.c*. Similarly, the java assignments have the command as javac followed by file name.

### 3.2 Creating Assignments

The second phase of the project is automating the creation of coding assignments on Canvas. Creating an assignment consists of multiple steps, such as adding the description, formatting the description, uploading the relevant files, linking those files in the description,

and modifying the attributes of the assignment such as due date, publish, rubric, etc. In order to streamline the process, this feature has been added. Manually creating the assignment on Canvas is time-consuming and tedious as it involves all the steps mentioned above. Creating the assignments each semester can be monotonous and with these many steps involved, chances of errors in the format of the assignment are high.

Adding examples in each assignment and formatting is again a tedious job. In order to make an instructor's job easier and provide a good user experience to students, this feature has been added. In the following subsections, we will go through each of them.

### 3.2.1 Uploading Files

Firstly, as part of this feature, CodEval requires certain files to be present on Canvas. Some of the files required are specification file and other files that students would need to download for that particular assignment. Manually uploading the files can be frustrating as you will have to upload one file at a time on Canvas using the upload button as shown below.

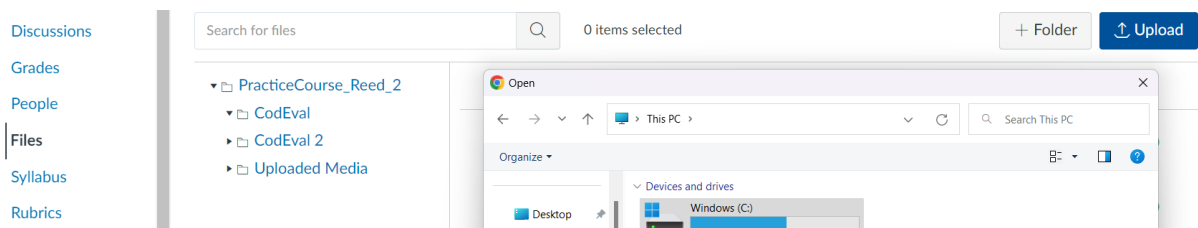


Figure 7: Uploading the files manually

In order to simplify this process Canvas API [61] 'upload' is used. The instructor or grader will have to create a folder called *assignmentFiles* on their server which would contain all the files that need to be uploaded. Canvas supports varieties of file extensions such as zip, docx, pdf, xls, py, etc. The below figure shows the *assignmentFiles* folder containing all the required files.



```

aditi@cs-reed-07:~/CodEval$ ls
assignmentFiles  commons.py      distributed  evaluate.sh   __main__.py  parsediff
codeval.py      convertMD2Html.py Dockerfile    file_utils.py make_zipapp.sh parsevalgrind
aditi@cs-reed-07:~/CodEval$ cd assignmentFiles/
aditi@cs-reed-07:~/CodEval/assignmentFiles$ ls
a_big_bag_of_strings.txt  a_big_bag_of_strings.txt.html  Absenteeism_at_work.xls

```

Figure 8: assignmentFiles folder and required files in it

Once the files are present in the *assignmentFiles* folder, CodEval will pick each file and upload it on Canvas in the course mentioned in the command line arguments. If the command line option ‘—dry-run’ is provided then it will not upload the files and will print and information statement instead. If the file upload fails, then it will throw an exception and exit. In the below figure you can see that under the Files section under the CodEval folder the files are uploaded.

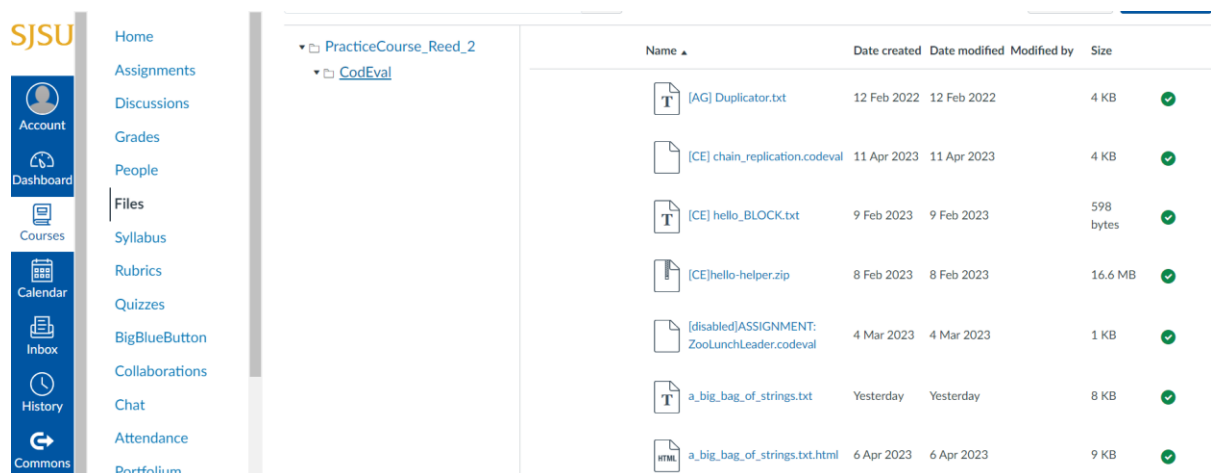
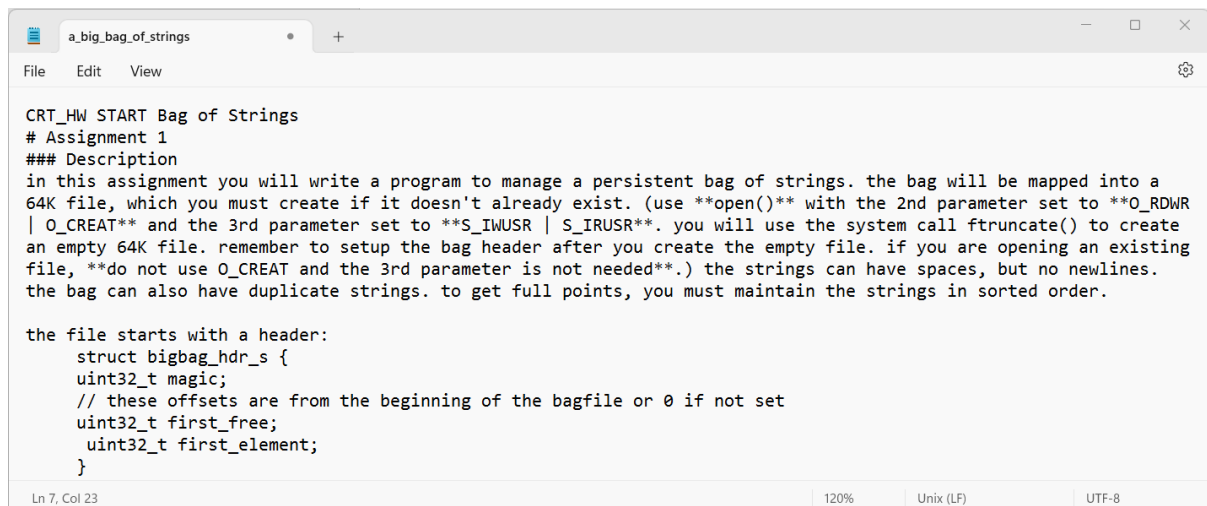


Figure 9: Files uploaded on Canvas

After the successful upload of the files, CodEval will update a dictionary that contains the file names as the key and the Url of the corresponding file uploaded in the Files section of the course on Canvas as the value. This is later used by CodEval to update a tag as described in the next section.

### 3.2.2 Converting Text to Markdown

The description of the assignment is present in the specification file. CodEval parses the description from the specification file which is in the markdown format [62] and converts it into HTML so that it can be uploaded on Canvas. The description is added between the tags `CRT_HW START` and `CRT_HW END`. The tag `CRT_HW START` is followed by the name of the assignment which will be the heading of the assignment on Canvas. The `CRT_HW END` tag will not have any value and is purely to demarcate the creation and evaluation parts of the assignment in the specification file. Figures 10 and 11 show the tags in the specification file.

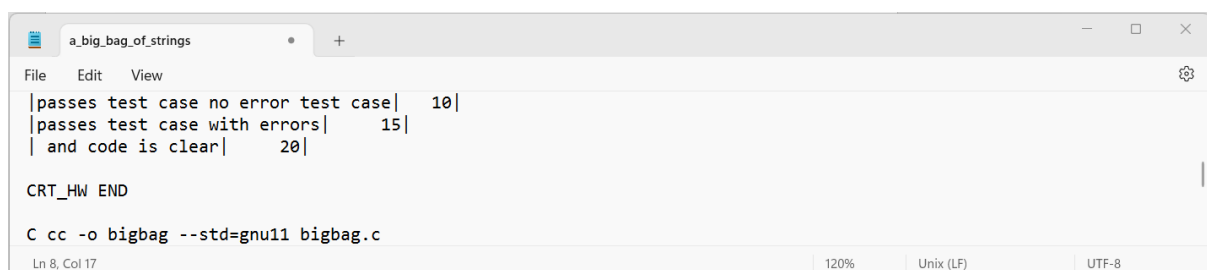


```
CRT_HW START Bag of Strings
# Assignment 1
### Description
in this assignment you will write a program to manage a persistent bag of strings. the bag will be mapped into a
64K file, which you must create if it doesn't already exist. (use **open(** with the 2nd parameter set to **O_RDWR
| O_CREAT** and the 3rd parameter set to **S_IWUSR | S_IRUSR**. you will use the system call ftruncate() to create
an empty 64K file. remember to setup the bag header after you create the empty file. if you are opening an existing
file, **do not use O_CREAT and the 3rd parameter is not needed**.) the strings can have spaces, but no newlines.
the bag can also have duplicate strings. to get full points, you must maintain the strings in sorted order.

the file starts with a header:
    struct bigbag_hdr_s {
        uint32_t magic;
        // these offsets are from the beginning of the bagfile or 0 if not set
        uint32_t first_free;
        uint32_t first_element;
    }
}

Ln 7, Col 23 | 120% | Unix (LF) | UTF-8
```

Figure 10: CRT\_HW START tag in the specification file



```
|passes test case no error test case| 10|
|passes test case with errors| 15|
| and code is clear| 20|

CRT_HW END

C cc -o bigbag --std=gnu11 bigbag.c

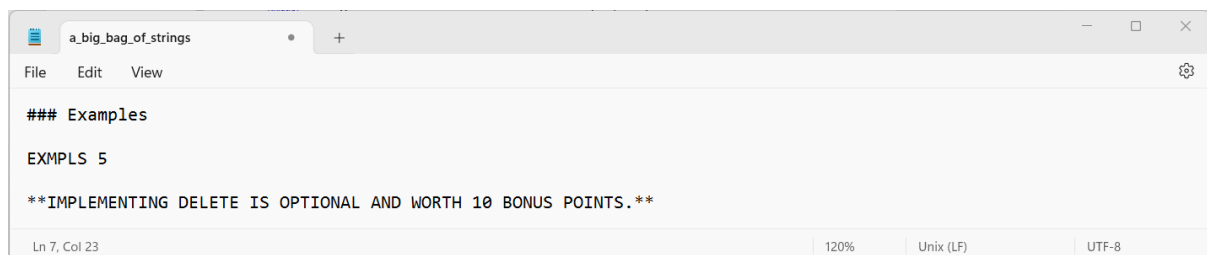
Ln 8, Col 17 | 120% | Unix (LF) | UTF-8
```

Figure 11: CRT\_HW END tag in the specification file

CodEval parses each line from the beginning of the specification file until the end. It will parse each line, modify it if required, and add it to a variable which will be converted to HTML using the markdown function. There are other tags that have been added which include *EXMPLS*, *URL\_OF\_HW*, and *DISCSN\_URL*.

*EXMPLS* tag is followed by a number, which is the number of test cases the instructor wants to print as part of the examples. The maximum value of this could be the number of non-hidden test cases and a minimum of 1. CodEval replaces this tag with the non-hidden test cases in the description of the assignment. In the Results section, the example shows tag *EXMPLS 5* has been replaced with five non-hidden test cases.

It is doing it by parsing the test cases after the *CRT\_HW\_END* tag, editing them to have different colors for input, output, and exit for easier readability, and finally replacing the tag with them. Figure 12 below, shows the *EXMPLS* tag in the specification file.



The image shows a code editor window with the following content:

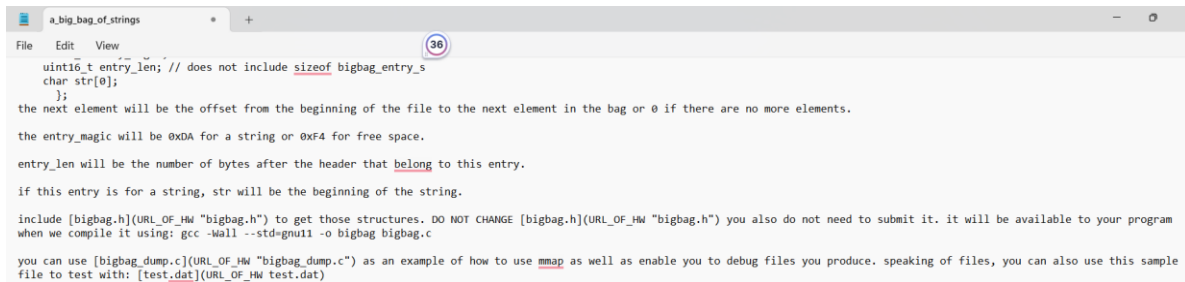
```
### Examples
EXMPLS 5
**IMPLEMENTING DELETE IS OPTIONAL AND WORTH 10 BONUS POINTS.**
```

The editor interface includes a menu bar (File, Edit, View), a status bar at the bottom left showing "Ln 7, Col 23", and a status bar at the bottom right showing "120%", "Unix (LF)", and "UTF-8".

Figure 12: EXMPLS tag in the specification file.

Another new tag that has been introduced is *URL\_OF\_HW* which is followed by the name of the file that needs to be linked and is mentioned in double quotes. CodEval reads the filename and checks if this file name is present in the dictionary, if it is then it will update

the specification file by replacing the tag with the file URL obtained from the dictionary. If the expected file is not present, then an exception is thrown. Figure 13 shows the `URL_OF_HW` tag followed by the file name in the specification file.

A screenshot of a code editor window titled 'a\_big\_bag\_of\_strings'. The editor shows a C code snippet with a comment: 'uint16\_t entry\_len; // does not include sizeof bigbag\_entry\_s'. Below the code, there are several explanatory lines: 'the next element will be the offset from the beginning of the file to the next element in the bag or 0 if there are no more elements.', 'the entry\_magic will be 0xDA for a string or 0xF4 for free space.', 'entry\_len will be the number of bytes after the header that belong to this entry.', and 'if this entry is for a string, str will be the beginning of the string.'. There are also instructions on how to include the header file and compile the program. A red circle highlights the number '36' in the top right corner of the editor window.

```
uint16_t entry_len; // does not include sizeof bigbag_entry_s
char str[0];
};
the next element will be the offset from the beginning of the file to the next element in the bag or 0 if there are no more elements.
the entry_magic will be 0xDA for a string or 0xF4 for free space.
entry_len will be the number of bytes after the header that belong to this entry.
if this entry is for a string, str will be the beginning of the string.
include [bigbag.h](URL_OF_HW "bigbag.h") to get those structures. DO NOT CHANGE [bigbag.h](URL_OF_HW "bigbag.h") you also do not need to submit it. it will be available to your program
when we compile it using: gcc -Wall --std=gnu11 -o bigbag bigbag.c
you can use [bigbag_dump.c](URL_OF_HW "bigbag_dump.c") as an example of how to use mmap as well as enable you to debug files you produce. speaking of files, you can also use this sample
file to test with: [test.dat](URL_OF_HW test.dat)
```

Figure 13: `URL_OF_HW` in the specification file

The last tag introduced is `DISCSN_URL` which does not have any value. This tag is related to the discussion topic corresponding to the assignment. Processing this tag is taken care of in creating the assignment section and therefore it will be discussed there.

### 3.2.3 Creating and Editing Assignments

Once the required files are uploaded and the description text from the specification file has been converted from markdown to HTML, the next step is to create or edit the assignment. It will first get the list of groups in the course by using the `get_assignment_groups` Canvas API [61] and then it will get the group id by comparing the group name mentioned in the command line with the one present in the list.

After getting the group id, CodeEval will first get the list of assignments in the course using `get_assignments` Canvas API [61] and then it will check if the assignment name present in the specification file is present in this list, if it is then the assignment is edited, otherwise it

will be created. If the ‘—dry-run’ option is provided, then neither the assignment is created nor edited.

While editing the assignment, some of the attributes that can be changed are as follows:

- Description
- Assignment name
- Group id
- Points possible
- Published
- Allowed extensions

Canvas API used to edit the assignment is ‘edit’ [61], and in case of a failure, while using the API, an exception is thrown. If the assignment name is not present in the list of assignments, then the Canvas API ‘create’ [61], is called to create the assignment using the above-mentioned attributes. In case of failure, the exception is thrown otherwise a successful information message is printed.

### **3.2.4 Creating Discussion Topic and Linking**

Another problem that has been addressed by CodEval is creating a discussion topic for an assignment. It is an erroneous task to create it, as first we would have to create the assignment, then create the discussion topic related to the assignment and add the link of the corresponding assignment to it, and then again add the link of the discussion topic in the assignment description.

To tackle this, before the assignment is created, a discussion topic is created for that assignment using the Canvas API *create\_discussion\_topic* [61], and the message attribute in

the API call is kept empty. The created topic object contains *html\_url* attribute which is stored into a variable to be later used by *create\_assignment* API [61].

Once the discussion topic is created, the Canvas API *create\_assignment* is called to create the assignment with the attributes mentioned in the previous sub section. In the *description* attribute the html text is modified by replacing the *DISCSN\_URL* tag with the URL of the discussion topic just created. After which the above-created discussion topic is updated by calling *update* API [61], and the message is updated with some text and the corresponding assignment's URL. Figure 14 depicts the tag in the specification file.

```
### Discussion
for questions and clarifications, use this discussion group: DISCSN_URL

### Suggested steps
- write your program that check for the -t command line option and the filename. make sure the
```

Figure 14: DISCSN\_URL tag in the specification file

On successful creation of the assignment and discussion topic, an informative message is printed otherwise an exception is thrown. As one can see since this whole circular process is taken care of by CodEval, it eliminates the chances of error or provides the required debug logs to find the exact problem in case of any error thrown.

### 3.2.5 Command Line Arguments

User experience is an important part of any software development and due to this reason, the Click [63] library has been used to not only provide a good user experience but also to simplify the developer's job. Its built-in methods help in adding options to the command line arguments in a simple and easy way.

CodEval uses the decorators such as 'group' or 'command' to create a command. For instance, adding `@click.command` right before a function definition will create the command and it will perform the tasks mentioned in the function. An example of the same is mentioned in Figure 15 [63], below.

```
import click

@click.command()
@click.option('--count', default=1, help='Number of greetings.')
@click.option('--name', prompt='Your name',
              help='The person to greet.')
def hello(count, name):
    """Simple program that greets NAME for a total of COUNT times."""
    for x in range(count):
        click.echo(f"Hello {name}!")

if __name__ == '__main__':
    hello()
```

And what it looks like when run:

```
$ python hello.py --count=3
Your name: John
Hello John!
Hello John!
Hello John!
```

It automatically generates nicely formatted help pages:

```
$ python hello.py --help
Usage: hello.py [OPTIONS]

    Simple program that greets NAME for a total of COUNT times.

Options:
  --count INTEGER  Number of greetings.
  --name TEXT      The person to greet.
  --help           Show this message and exit.
```

Figure 15: Example of creating commands using click [63]

In CodEval, there are two commands:

- `create_assignment` : To create the assignment.
- `grade_assignment` : To grade the assignment.

Each of these commands is followed by their respective arguments such as course name or spec name and then followed by the optional arguments such as:

- *dry run/no\_dry\_run*: This command will run either of the processes and will perform all the steps in the process but will not update Canvas.
- *verbose/no\_verbose*: This command provides detailed logs and information about each step in the process.
- *force/no\_force*: This command is available with *grade\_submissions* command only. This sub-command will grade the submissions even though they are already graded.
- *group id*: This command is only available with the *create\_assignment* command. It is an optional command where the user can mention the name of the group in which the assignment needs to be created.

Figures 16 and 17 show examples of these commands with their options.

```

aditi@cs-reed-07: ~/CodEval
(venv) aditi@cs-reed-07:~/CodEval$ python3 codeval.py --help
Usage: codeval.py [OPTIONS] COMMAND [ARGS]...

Options:
  --help  Show this message and exit.

Commands:
  create-assignment  Create the assignment in the given course.
  grade-submissions  Grade unsubmitted graded submission in the given...
(venv) aditi@cs-reed-07:~/CodEval$

```

Figure 16: Command line arguments options in CodEval

```

aditi@cs-reed-07: ~/CodEval
(venv) aditi@cs-reed-07:~/CodEval$ python3 codeval.py grade-submissions "PracticeCourse_Reed_2" --help
2023-04-29 00:13:07 I connected to canvas as Aditi Agrawal (4514571)
Usage: codeval.py grade-submissions [OPTIONS] COURSE_NAME

  Grade unsubmitted graded submission in the given course.

Options:
  --dry-run / --no-dry-run      Grade submissions but don't update canvas
                                [default: dry-run]
  --verbose / --no-verbose     Verbose actions [default: no-verbose]
  --force / --no-force         Grade submissions even if already graded
                                [default: no-force]
  --copytmpdir / --no-copytmpdir  copy tmpdirs to current directory [default:
                                no-copytmpdir]
  --help                        Show this message and exit.
(venv) aditi@cs-reed-07:~/CodEval$

```

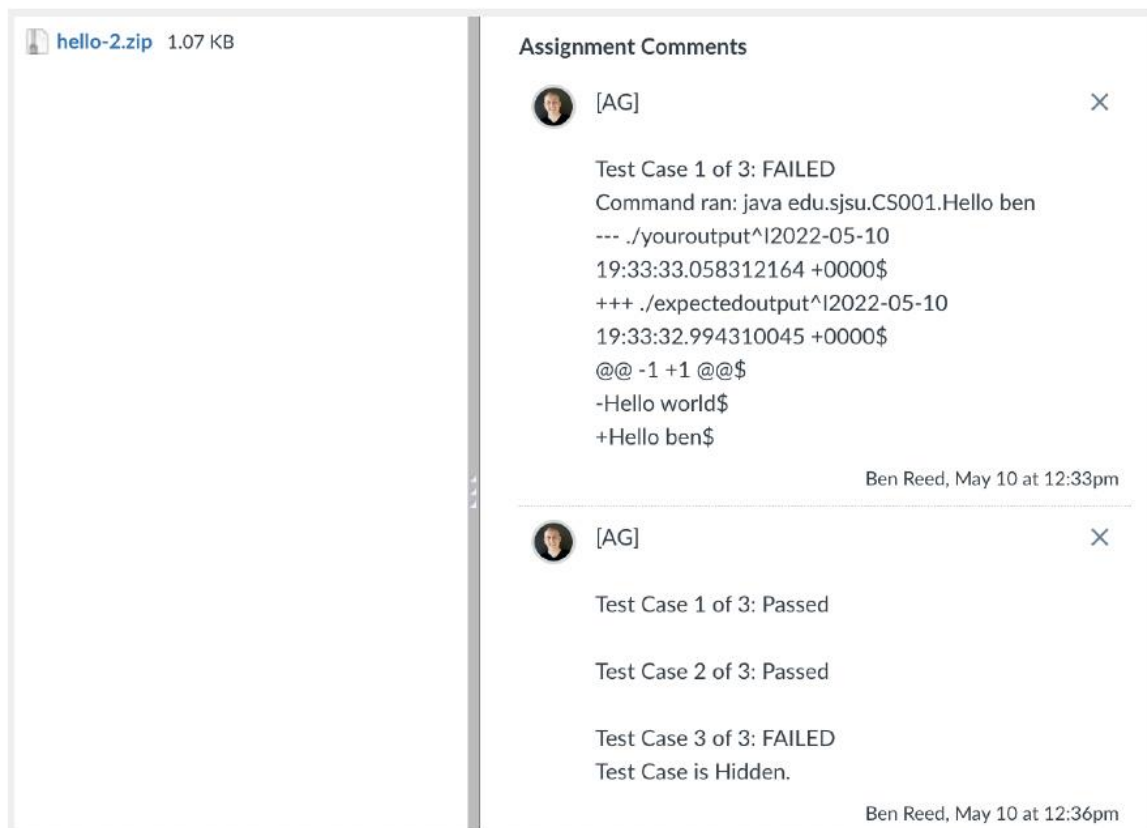
Figure 17: grade-submissions command's options



## IV. RESULTS

### 4.1 Evaluation of Submissions

CodEval seamlessly creates edits and grades coding assignments. A sample submission evaluation for an assignment is shown in Figure 18. The assignment requires the program to output "hello" followed by the name supplied on the command line, but the submission is the traditional "hello world" example that outputs "hello world." CodEval discovered the error and informed the student. The second submission passed the first test because "Hello Ben" was printed on it. The submission passes the usage statement for the second test as well; however, it fails the third test because it consistently outputs "Hello Ben."



The screenshot displays the CodEval submission evaluation interface. On the left, a file named "hello-2.zip" (1.07 KB) is shown. The right panel, titled "Assignment Comments", contains two entries from user [AG].

**First Comment:**

- Test Case 1 of 3: FAILED
- Command ran: java edu.sjsu.CS001.Hello ben
- ./youroutput^|2022-05-10 19:33:33.058312164 +0000\$
- +++ ./expectedoutput^|2022-05-10 19:33:32.994310045 +0000\$
- @@ -1 +1 @@\$
- Hello world\$
- +Hello ben\$

Ben Reed, May 10 at 12:33pm

**Second Comment:**

- Test Case 1 of 3: Passed
- Test Case 2 of 3: Passed
- Test Case 3 of 3: FAILED
- Test Case is Hidden.

Ben Reed, May 10 at 12:36pm

Figure 18: Example of a submission evaluation by CodEval

#### 4.1.2 Student Evaluations

Our hypothesis was that the students would begin their assignments sooner because they could receive quick feedback on them from CodEval at the beginning of the semester, but this did not appear to be the case. In comparison to previous semesters, there was no difference in the need for extensions or the absence of late submissions. Few students, as seen in the student feedback, thought CodEval improved early starts to the assignments. The outcomes show that students believed CodEval was beneficial in helping them polish their work and complete the projects.

Metric	Strongly Disagree	Somewhat Disagree	Neither Agree or Disagree	Somewhat Agree	Strongly Agree
Start assignment early	0	6	12	6	4
Improved code quality	0	1	0	14	13
Learn more from assignment	0	4	6	12	6
Improved overall success	0	0	3	12	13

Table 2: Survey Results

Another observation is that no one expressed dissatisfaction with their mark after the assignment had been graded. This is true even if the assignments aren't really graded by CodEval; instead, the grader is responsible for determining whether the submission was made correctly. Students were also asked to remark on how CodEval helped them and any changes they would like to see, in addition to scoring those four specified dimensions. The students through their responses stated the following about the impact of CodEval:

- Immediate feedback was very valuable
- They were able to catch errors that they wouldn't have otherwise
- They were motivated to fix the errors in time
- They were able to better understand and handle test cases
- The tool helped their overall motivation
- They were more confident about their submission
- They could predict how well they did on an assignment before grades were awarded
- Helped them catch small bugs

The improvements suggested by students are:

- Better readability of the result.
- Hints for hidden test cases.

Four assignments were used in an earlier class that took place two semesters before. Back then, CodEval did not exist, so students did not receive feedback until after the assignments had been fully graded. There were two sections of the class in the fall of 2020. Table 3 shows that for the first three assignments, the average scores for Spring 2022 were close to or in the middle of the averages for the two parts. This shows that for the early assignments, students' perceptions of their accomplishment and the quality of their code were more accurate than their actual experiences. However, CodEval clearly made improvements to the final assignment, which was extremely challenging and complex.

It's critical to keep in mind that we used different graders for the fall of 2020 and the spring of 2022 when making this comparison. The testing of CodEval, which is more thorough

than the ad-hoc scripts that graders in Fall 2020 utilize, was also not available to the Fall 2020 graders. The scores in Fall 2020 may have been exaggerated due to a lack of testing rigor.

Assignments	Difficulty	Fall 2020 Sec 2	Fall 2020 Sec 3	Spring 2022 Sec 1
count the bits	easy	94.94871795	91.02702703	92.6125
count the bits faster	moderate	88.76923077	93.75675676	87.6625
flaky tests	difficult	77.25641026	81.43243243	78.4125
a big bag of strings	very difficult	76.46153846	79.62162162	86.35

Table 3: Comparing assignments from Fall 2020 where CodEval was not used with Spring 2022, where it has been used

## 4.2 Assignment Creation

Figure 19 shows the generated HTML of the description which will be used to create the assignment. As one can see, the # has been replaced with `<h1></h1>` tags and the paragraph has been prefixed and suffixed with `<p></p>` tag.

```

aditi@cs-reed-07: ~/CodEval/
<h1>Assignment 1</h1>
<h3>Description</h3>
<p>in this assignment you will write a program to manage a persistent bag of strings. the bag will be mapped into a 64K file, which you must create if it doesn't already exist. (use <strong>open()</strong> with the 2nd parameter set to <strong>O_RDONLY | O_CREAT</strong> and the 3rd parameter set to <strong>S_IWUSR | S_IRUSR</strong>. you will use the system call ftruncate() to create an empty 64K file. remember to setup the bag header after you create the empty file. if you are opening an existing file, <strong>do not use O_CREAT and the 3rd parameter is not needed</strong>.) the strings can have spaces, but no newlines. the bag can also have duplicate strings. to get full points, you must maintain the strings in sorted order.</p>
<p>the file starts with a header:
struct bigbag_hdr_s {
    uint32_t magic;
    // these offsets are from the beginning of the bagfile or 0 if not set
    uint32_t first_free;
    uint32_t first_element;
}</p>
1,1 Top

```

Figure 19: Html format of the description

Figure 20 shows how the reference link has been replaced in place of the `URL_OF_HW` tag in Figure 13. This link is accessed after the files have been uploaded on Canvas.

```

aditi@cs-reed-07: ~/CodEval/
<p>the entry_magic will be 0xDA for a string or 0xF4 for free space.</p>
<p>entry_len will be the number of bytes after the header that belong to this entry.</p>
<p>if this entry is for a string, str will be the beginning of the string.</p>
<p>include <a href="https://sjsu.instructure.com/files/72851569/download?download_frd=1&verifier=Q7h0q1tTI3W2ZRLP3stLCrHexul012V9741fxXRx">bigbag.h</a> to get those structures. DO NOT CHANGE <a href="https://sjsu.instructure.com/files/72851569/download?download_frd=1&verifier=Q7h0q1tTI3W2ZRLP3stLCrHexul012V9741fxXRx">bigbag.h</a> you also do not need to submit it. it will be available to your program when we compile it using: gcc -Wall --std=gnu11 -o bigbag bigbag.c</p>
<p>you can use <a href="https://sjsu.instructure.com/files/72851569/download?download_frd=1&verifier=Q7h0q1tTI3W2ZRLP3stLCrHexul012V9741fxXRx">bigbag_dump.c</a> as an example of how to use mmap as well as enable you to debug files you produce. speaking of files, you can also use this sample file to test with: <a href="https://nodejs.org/">test.dat</a></p>
<p>your program will memory map the bag file with <strong>mmap()</strong> to access the file. all file changes will be through memory accesses.</p>
<h3>Usage statement</h3>
27,1 15%

```

Figure 20: URL\_OF\_HW tag has been replaced with the file URL

Figure 21 shows that created assignment from the html format on Canvas. As one can see the name of the assignment 'Bag of Strings' which was the value for the tag *CRT\_HW START* is right at the top of the assignment. The headings which are surrounded by *<h1>*, *<h3>* tags for example assignment, and description is highlighted in this figure.

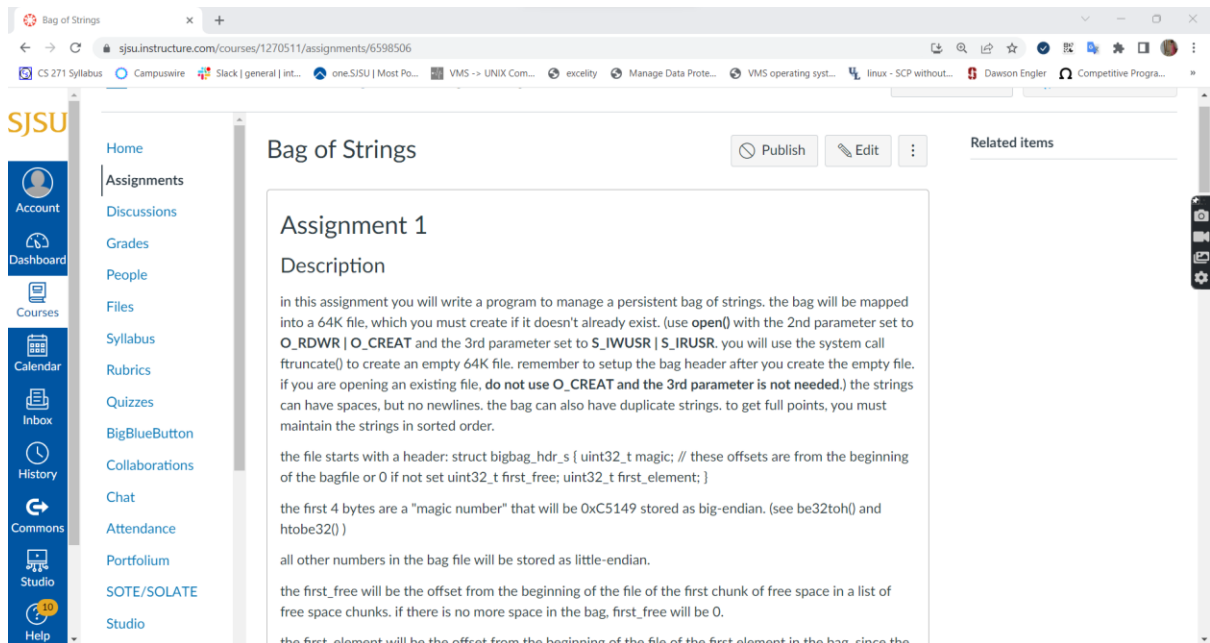


Figure 21: Created Assignment on Canvas

Figure 22 shows the examples in the description. As shown in Figure 12, the *EXMPLS* tag is replaced with 5 test cases. The examples are color coded for enhanced user readability. Figure 23 shows the highlighted file name which when clicked will either download the file or will direct it to the corresponding URL. Not only files but as you can see under the discussion topic, the *DISCSN\_URL* tag shown in Figure 14, is replaced with the link to the corresponding assignment.

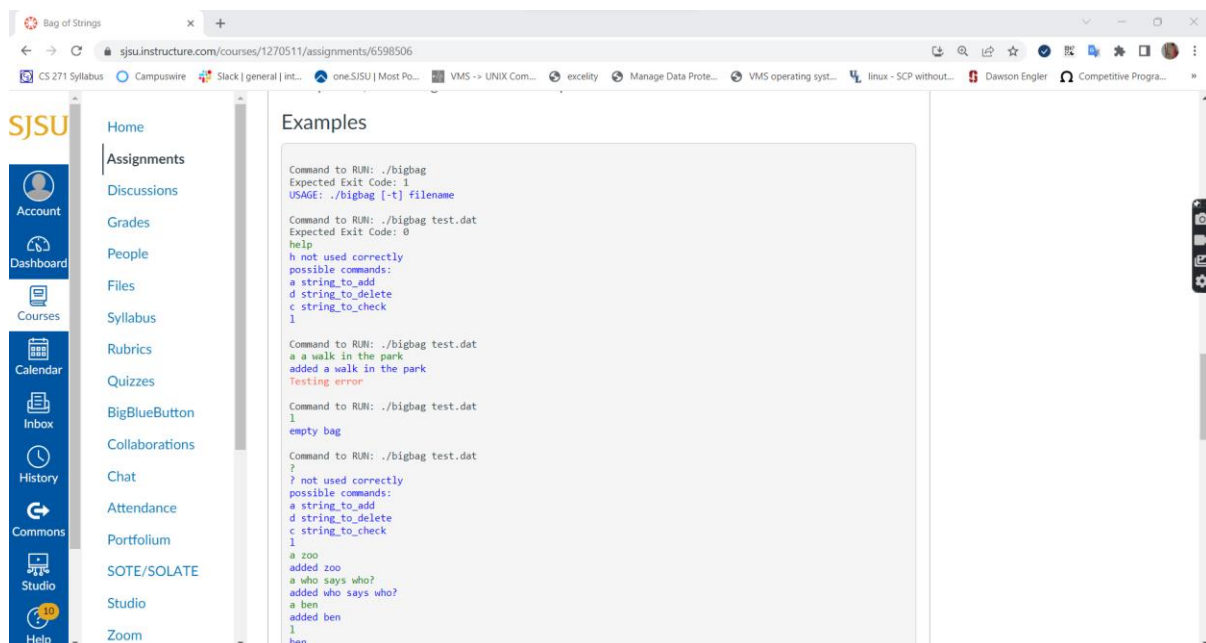


Figure 22: Examples in the description

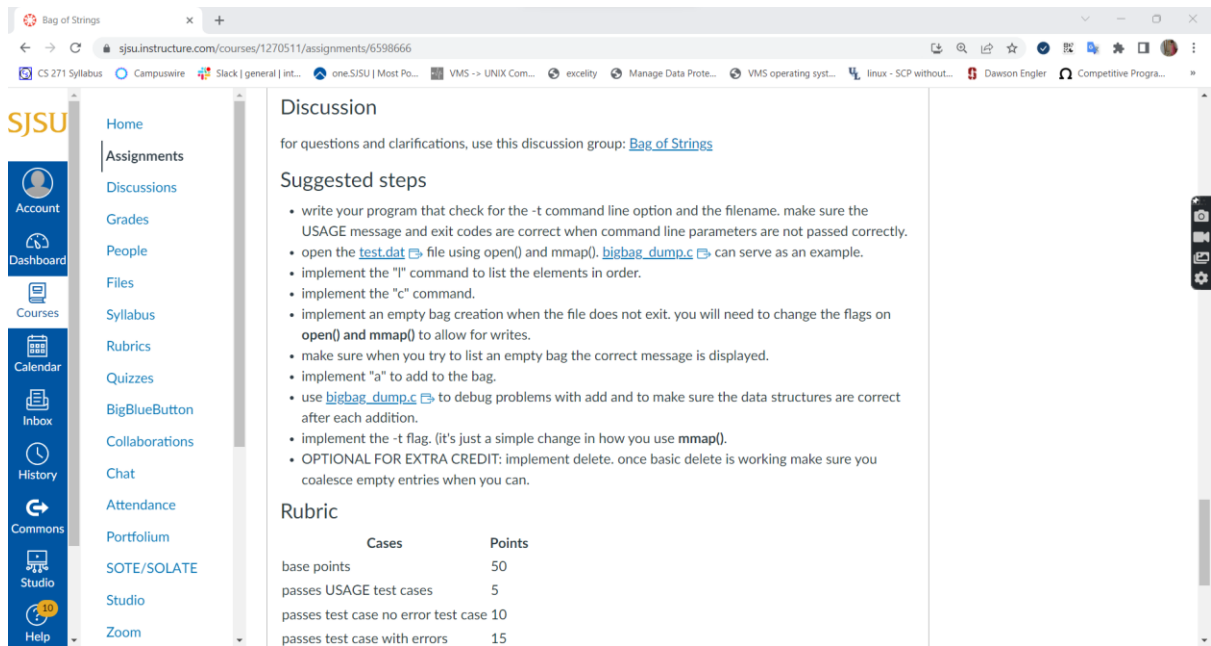


Figure 23: Referenced files and discussion topic

Figure 24 shows the successful creation of a discussion topic under the Discussions tab, and in the description of the topic, you can see the assignment name which is linked to the corresponding assignment.

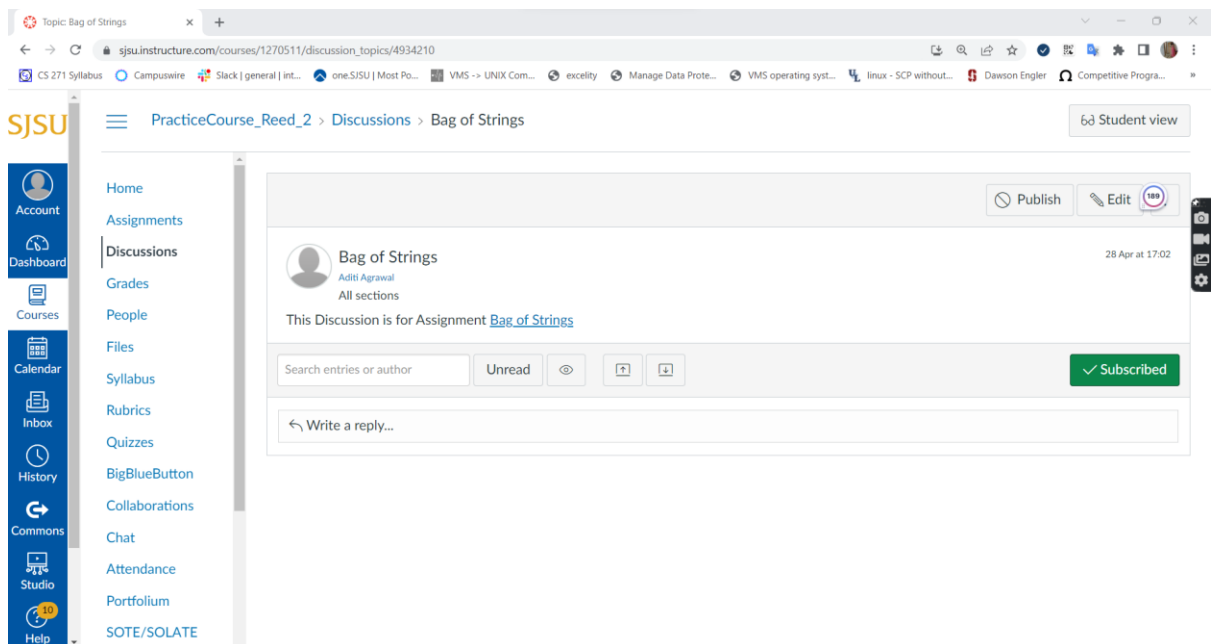


Figure 24: Created Discussion topic, with reference to the assignment.

## V. FUTURE WORK

CodEval's design is not only simple but also extensible and therefore it can be used in various programming courses extending across various programming languages. Other than its flexibility to be incorporated in different programming courses and environments, the next phase could involve some of the below-mentioned enhancements.

### 5.1 Adding Support for Python

As of now, CodEval supports C and Java languages. So the next step would be to incorporate the support for Python. As mentioned earlier, the difficulty with Python is that in most courses Jupyter notebooks are used and the files exported from there are not in `.py` format but in `.ipynb` format therefore there would be some extra work done in converting the code into `.py` format. In the case of code already written in `.py` format files, it would be straightforward as using the existing Java or C code with the addition of a Python translator in Docker.

### 5.2 Enhancing Readability of Output

After the last modification to enhance the readability of the description sent by CodEval in the comments section when a test case fails, there is still some room for improvement. The problem with failure explanation is twofold.

- The output from the diff tool is complicated and confusing.
- The size of the comment section makes it difficult to read the feedback from CodEval.

The last enhancement solved the first problem mentioned above by providing a legible format that a student can understand. To solve the second problem, the feedback from CodEval can be directed into a file, one for output differences and another for stderr



differences. The files can be either downloaded from the comment section or a hyperlink can be provided to open the file on the web browser.

### **5.3 Convert the bash code to Python**

As of now a bash script is used to parse the specification file and it has been used to its maximum capacity. In the future, so that the developer can add more functionality and do it easily, the bash code can be converted to Python code.

### **5.4 Add Support for Plagiarism Check**

It is already easy to copy the code from classmates and websites like GeeksforGeeks [64] and Stackoverflow [65], but with the introduction of ChatGPT [66] and Bard [67], it is more crucial to validate the authenticity of the work submitted.

To achieve this, either a plagiarism check logic must be written and added in CodEval, or a plagiarism tool can be integrated. The problem with the former option is that it is time-consuming and a whole other project on its own. The latter option of integrating third-party tools for example MOSS [68] seems more feasible and apt.

### **5.5 Integrating CodEval with other LMS**

Another fascinating feature to add in CodEval is supporting and testing the integration with other LMS such as BlackBoard [69], Moodle [70] , etc. This will increase the user base for CodEval. This will also lead to the creation a general and simplified template to integrate CodEval in LMSs.

## VI. CONCLUSION

As learning management systems such as Canvas becomes more and more popular, tools such as CodEval become more crucial for courses related to programming. The aim of this project was to create a system that provides instant feedback to students, in turn, relieves the instructor or grader from the tedious aspects of grading the assignment and creates assignments quickly. All three of these goals have been achieved with CodEval.

CodEval was simple and easy to use in the Operating systems and Networking classes. The design was simple to make enhancements on it and as a result, it has been extended to be used in Distributed Systems class as well where the specification file's format has been utilized. The future hope with this project is to be introduced to other instructors at San Jose University so that with some initial modification, they can use CodEval in their courses as well. CodEval is open source and therefore anybody can enhance and modify it as per their requirements.

Those interested in using CodEval, can go to [8] and find the relevant information on how to get started with CodEval. To conclude, CodEval has garnered positive feedback not only from instructors and graders for simplifying their work but also from students who witnessed improvement in their coding skills and experience.

## REFERENCES

- [1] Instructure, "About Canvas | Edtech Learning Platform | Instructure", instructure.com, <https://www.instructure.com/canvas>
- [2] Moodle. Retrieved from <https://moodle.org/>
- [3] CodeLab by Turingscraft. Retrieved from <https://www.turingscraft.com/>
- [4] D. Jackson and M. Usher. "Grading student programs using ASSYST.", in Proceedings of the 28th SIGCSE Technical Symposium on Computer Science Education. ACM, NY, NY, pp. 335–339. 1997.
- [5] D. Insa and J. Silva, "Automatic assessment of Java code." in Computer Languages, Systems & Structures, vol. 53, 59–72, Sept. 2018. <https://doi.org/10.1016/j.cl.2018.01.004>
- [6] Markus: Online Marking, Retrieved from <https://github.com/MarkUsProject/Markus>
- [7] Submittity: An Open Source, Highly-Configurable Platform for Grading of Programming Assignments. Retrieved from <https://github.com/Submittity/Submittity>
- [8] CodEval, "CodEval: Timely Evaluation of Code Submission for Canvas", Access 11 May, 2022, Retrieved from <https://github.com/SJSU-CS-systems-group/CodEval>
- [9] N. Rashid, Laurianne Lim, Ooi Sin Eng, Tan Huck Ping, Z. Zainol, O. Majid," A Framework of an automatic assessment system for learning programming", Advanced Computer and Communication Engineering Technology, vol. 362, 2016, doi: [https://doi.org/10.1007/978-3-319-24584-3\\_82](https://doi.org/10.1007/978-3-319-24584-3_82)
- [10] Hussam Aldriye, Asma Alkhalaf, and Muath Alkhalaf, "Automated grading systems for programming assignments: A literature review" Int J Adv Comput Sci Appl, 10(3), 2019. <http://dx.doi.org/10.14569/IJACSA.2019.0100328>
- [11] J.C. Caiza, J.M. Del Alamo," Programming assignments automatic grading: review of tools and implementations ", in INTED2013 Proc., pp. 5691-5700, 2012.

- [12] C. A.Higgins, G. Gray, P. Symeonidis, A. Tsintsifas," Automated assessment and experiences of teaching programming.", J. Educ. Resour. Comput. , vol. 5, pp. 5., Sept. 2005. <https://doi.org/10.1145/1163405.1163410>
- [13] A. Patil, "Automatic Grading of Programming Assignments", 2010. doi: <https://doi.org/10.31979/etd.vnt7-hgnd>
- [14] S. H. Edwards, and M. A. Perez-Quinones," Web-CAT: automatically grading programming assignments.", in Proceedings of the 13th annual conference on Innovation and technology in computer science education(ITICSE '08), ACM SIGCSE Bulletin, vol. 40, pp. 328-328. doi: <https://doi.org/10.1145/1384271.1384371>
- [15] J. C. Rodríguez-del-Pino, E. Rubio Royo, and Z. Hernández Figueroa, "A Virtual Programming Lab for Moodle with automatic assessment and anti-plagiarism features," in Proc. WorldComp12, 2012.
- [16] A.Gordillo, "Effect of an instructor-centered tool for automatic assessment of programming assignments on student's perceptions and performance," Sustainability 11, no. 20: 5568, 2019. <https://doi.org/10.3390/su11205568>
- [17] J. Spacco, D.Hovemeyer, W. Pugh, F. Emad, J. K. Hollingsworth, N. Padua-Perez," Experiences with marmoset: designing and using an advanced submission and testing system for programming courses." in Proceedings of the 11th annual SIGCSE conf. on Innovation and technology in computer science education (ITICSE '06), pp. 13-17, 2006. doi: <https://doi.org/10.1145/1140124.1140131>
- [18] M. Amelung, P. Forbrig, D. Rösner," Towards generic and flexible web services for e-assessment.", ACM SIGCSE Bulletin, pp. 219-224, 2008.
- [19] X. Liu, S. Wang, P. Wang, and D. Wu, "Automatic Grading of Programming Assignments: An Approach Based on Formal Semantics," 2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET), pp. 126-137, 2019. doi: 10.1109/ICSE-SEET.2019.00022.

- [20] K. M. Ala-Mutka, "A survey of automated assessment approaches for programming assignments," Computer Science Education, vol. 15,no. 2, pp. 83–102., June 2005.
- [21] P. Ihanola, T. Ahoniemi, V. Karavirta, and O. Seppala," Review of recent systems for automatic assessment of programming assignments.", in Proceedings of the 10th Koli Calling International Conference on Computing Education Research (Koli Calling'10). ACM, New York, NY, 86–93, 2010.
- [22] Y. Liang, Q. Liu, J. Xu, and D. Wang. "The recent development of automated programming assessment.", in Computational Intelligence and Software Engineering, pp. 1-5, Dec 2009.
- [23] M. T. Helmick," Interface-based programming assignments and automatic grading of java programs.", in ITiCSE '07: Proceedings of the 12th annual SIGCSE Conf. on Innovation and technology in computer science education, ACM, New York, NY, USA, pp. 63–67, 2007.
- [24] J.C. Paiva, J.P. Leal, A. Figueria," Automated Assessment in Computer Science Education: A State-of-the-Art Review", in ACM Trans. Comput. Educ., vol. 22, Article 34, NY, NY, USA, Sept 2022.
- [25] U. Buranasaksee, " LINSIM: A framework of an automatic assessment for Linux-based operating system exercises.", in Proceedings of the 5th International Conference on Information and Education Technology (ICIET'17). ACM, New York, NY, pp. 121–125, 2017.
- [26] E. Gramond and S. H. Rodger, "Using JFLAP to interact with theorems in automata theory.", in Proceedings of the 30th SIGCSE Technical Symposium on Computer Science Education (SIGCSE'99). ACM, New York, NY,pp. 336–340. <https://doi.org/10.1145/299649.299800>
- [27] C. Hundt, M. Schlarb, and B. Schmidt, "SAUCE: A web application for interactive teaching and learning of parallel programming.", in Journal of Parallel and Distributed Computing, vol. 105,pp. 163–173, July 2017. <https://doi.org/10.1016/j.jpdc.2016.12.028>
- [28] H. Aldriye, A. Alkhalaf and M. Alkhalaf, "Automated Grading Systems for Programming Assignments: A Literature Review", in International Journal of

Advanced Computer Science and Application(IJAACSA), vol. 10, no. 3,2019.

<http://dx.doi.org/10.14569/IJACSA.2019.0100328>

- [29] D.M. Souza, K.R. Felizardo, and E.F. Barbosa, "A Systematic Literature Review of Assessment Tools for Programming Assignments", in IEEE 29th International Conference on Software Engineering Education and Training, New Delhi, India, vol. 1, pp. 22-23, 2016.
- [30] M. Blumenstein, S. Green, A. Nguyen, and V. Muthukkumarasamy, "GAME: a generic automated marking environment for programming assessment," in Proceedings of the International Conference on Information Technology: Coding and Computing, ser. ITCC '04, pp. 212–216, 2004.
- [31] D. Jackson, "A semi-automated approach to online assessment," SIGCSE Bull., pp. 164–167, 2000.
- [32] S. H. Edwards and M. A. Perez-Quinones, "Web-CAT: automatically grading programming assignments," SIGCSE Bull., pp. 328–328, 2008.
- [33] J. P. Leal and F. Silva, "Mooshak: a web-based multi-site programming contest system," in Software: Practice and Experience, pp. 567–581, 2003. <https://doi.org/10.1002/spe.522>
- [34] C. Higgins, T. Hergazy, P. Symeonidis, and A. Tsinsifas, "The CourseMarker CBA system: Improvements over Ceilidh.", in Education and Information Technologies, vol. 8, pp. 287 – 304,2003
- [35] M. Luck and M. Joy, "Automatic submission in an evolutionary approach to computer science teaching," Computers & Education, vol. 25,no. 3, pp. 105–111, 1995.
- [36] M. S. Joy and M. Luck, "A user-friendly online submissions system," in Proceedings of the Fourth Annual Conference on the Teaching of Computing, pp. 92–95, 1996.
- [37] J.M. Almo," A study of Online Assessment Tools to Practice Programming and Their Effect on Students Grades" in ASEE Mid-Atlantic Section Spring Conference, Washington, Columbia, 2018.

- [38] Programmr: Your Online Code Lab. Accessed 19 February 2019. Retrieved from <http://www.programmr.com/>.
- [39] CodeStepByStep: A web-based practice problem tool for computer science students. Accessed 20th February 2018. Retrieved from <https://www.codestepbystep.com>
- [40] R. Queiros and L. José, " Programming Exercises Evaluation Systems - An Interoperability Survey." in CSEDU 2012 - Proceedings of the 4th International Conference on Computer Supported Education, vol. 1, pp. 83-90, 2012.
- [41] HTML. Accessed 8 December, 2022. Retrieved from <https://html.spec.whatwg.org/multipage/>
- [42] XML. Accessed on 7 February,2022. Retrieved from <https://www.w3.org/TR/REC-xml/>
- [43] S.Safris, "A Deep Look at JSON vs. XML, Part 1: The History of Each". Toptal Engineering Blog. <https://www.toptal.com/web/json-vs-xml-part-1> (accessed Dec.31, 2022)
- [44] YAML. Retrieved from <https://yaml.org/>
- [45] PEML. Retrieved from <https://cssplice.github.io/peml/>
- [46] ArchieML. Retrieved from <http://archieml.org/>
- [47] C. F. [Goldfarb](#), "[The Roots of SGML – A Personal Recollection](#)". Accesses on July 7, 2007. Retrieved from [sgmlsource.com/history/roots.htm](http://sgmlsource.com/history/roots.htm)
- [48] Github. Retrieved from <https://github.com/>
- [49] Markdown. Retrieved from <https://www.markdownguide.org/getting-started/>
- [50] RStudio. Retrieved from <https://posit.co/>
- [51] iaWriter. Retrieved from <https://ia.net/writer>
- [52] ghostwriter. Retrieved from <https://kde.github.io/ghostwriter/>
- [53] Markdown Monster. Retrieved from <https://markdownmonster.west-wind.com/>
- [54] ReText. Retrieved from <https://github.com/retext-project/retext>
- [55] StackEdit. Retrieved from <https://www.markdownguide.org/tools/stackedit/>

- [56] "A formal spec for GitHub Flavored Markdown". GitHub Engineering. Accessed from 16 Mar 2017. Retrieved from <https://github.blog/2017-03-14-a-formal-spec-for-github-markdown/>
- [57] Drupal. Retrieved from <https://www.drupal.org/project/drupal/releases/9.4.8> from
- [58] TYPO3. Retrieved from [https://extensions.typo3.org/extension/markdown\\_content/](https://extensions.typo3.org/extension/markdown_content/) from
- [59] N. Nethercote, J. Seward, "Valgrind: A framework for heavyweight dynamic binary instrumentation" in Proceedings of ACM SIGPLAN 2007 Conference on Programming Language Design and Implementation (PLDI 2007), pp. 89-100, June 2007. doi: <https://doi.org/10.1145/1250734.1250746>
- [60] Jupyter. Retrieved from <https://jupyter.org/>
- [61] Canvas API. Retrieved from <https://canvas.instructure.com/doc/api/>
- [62] Markdown. Retrieved from <https://python-markdown.github.io/>
- [63] Click. Retrieved from <https://click.palletsprojects.com/en/8.1.x/#documentation> from
- [64] GeeksforGeeks. Retrieved from <https://www.geeksforgeeks.org/>
- [65] Stackoverflow. Retrieved from <https://stackoverflow.com/>
- [66] ChatGPT. Retrieved from <https://openai.com/blog/chatgpt>
- [67] Bard. Retrieved from <https://bard.google.com/>
- [68] Moss. Retrieved from <https://theory.stanford.edu/~aiken/moss/>
- [69] Blackboard. Retrieved from <https://www.blackboard.com/en-apac/teaching-learning/learning-management>
- [70] Moodle. Retrieved from <https://moodle.org/>
- [71] A. Agrawal, A. Jain, and B. Reed, "Codeval: Improving student success in programming assignments", in 14<sup>th</sup> International Conference on Education and New Learning Technologies, pp. 7546-7554, 2022. doi: [10.21125/edulearn.2022.1767](https://doi.org/10.21125/edulearn.2022.1767)