Spring 2016

Java Auto Grader

Ashrita Kunchala

Maneesh Gunnala Ranga Rao

Governors State University

Follow this and additional works at: http://opus.govst.edu/capstones

Part of the Computer Sciences Commons

Recommended Citation
http://opus.govst.edu/capstones/196

For more information about the academic degree, extended learning, and certificate programs of Governors State University, go to http://www.govst.edu/Academics/Degree_Programs_and_Certifications/

Visit the Governors State Computer Science Department
This Project Summary is brought to you for free and open access by the Student Capstone Projects at OPUS Open Portal to University Scholarship. It has been accepted for inclusion in All Capstone Projects by an authorized administrator of OPUS Open Portal to University Scholarship. For more information, please contact opus@govst.edu.
ABSTRACT

The project is developing a “Java Automatic Grader” for java project assignment. It ensures that every assignment or work given to the students is done in the right way. Students should get automatic evaluation after submitting the assignment. We are implementing each test case in such a way that student should only get marks for satisfying all the requirements of the project assignment.
# Table of Content

1. Project description ......................................................................................................................... 4  
   1.1 Competitive information ........................................................................................................... 4  
   1.2 Relationship to other Application/projects. ............................................................................... 5  
   1.3 Assumptions and dependencies ............................................................................................... 5  
   1.4 Future enhancements ............................................................................................................... 5  
   1.5 Definitions and Acronyms ...................................................................................................... 6  

2 Project Technical Description ........................................................................................................... 7  
   2.1 Application Architecture ........................................................................................................ 7  
   2.2 Application information flows .................................................................................................. 11  
   2.3 Interactions with other application .......................................................................................... 13  
   2.4 Capabilities .............................................................................................................................. 13  
   2.5 Risk assessment and management ........................................................................................... 13  

3 Project requirements .......................................................................................................................... 15  
   3.1 Identification requirements ...................................................................................................... 15  
   3.2 Operations, Administration, Maintenance and Provisioning ..................................................... 17  
   3.3 Security and Fraud prevention ................................................................................................ 17  
   3.4 Release and Transition plan ................................................................................................... 18  

4 Project design description .................................................................................................................. 19  

5. Internal/External interface impacts and specification ........................................................................ 24  

6. Design unit’s impacts ......................................................................................................................... 25  
   6.1 Functional Area/Design Unit A .............................................................................................. 25  
   6.1.1 Functional Overview ........................................................................................................ 25  
   6.1.2 Impacts ............................................................................................................................... 25  
   6.1.3 Requirements ..................................................................................................................... 25  
   6.2 Functional Area B/Design unit B ............................................................................................ 27  
   6.2.1 Functional Overview ........................................................................................................ 27  
   6.2.2 Impacts ............................................................................................................................... 27  
   6.2.3 Requirements ..................................................................................................................... 27  

7. Open issues ........................................................................................................................................ 29  

8. Acknowledgements .......................................................................................................................... 30  

9. References. ......................................................................................................................................... 31
1 Project Description

The project is developing an Automatic Grader for java project assignment. It makes assured that every assignment or work given to the students is done in the precise way. Students should get automatic evaluation after submitting the assignment.

We have developed a plugin for learning management system, Moodle. This plugin allows the instructor to grade java programming assignments automatically. This plugin is built in actual Assignment module, that gives us an easy way to allow students to upload any digital content for grading.

The student implements the java unit test question on the question editing form and writes the response in the answer-field. After finishing the attempt, the student submits the response. In case the student response compiles the response gets graded automatically with a JUnit test implemented by the teacher. The JUnit test evaluates the response and returns a value between 0 and 100%, which represents the correctness of the student's Java code. After that, the student's grade is computed automatically by Moodle from this percentage value. With compilation error the student's answer is graded as wrong with 0% correctness. After executing the grading, the student can view his given answer with a feedback about the test. The student gets all the questions with question text, the student-response and the compiler- or Junit execution-output.

This application will give teachers a way to consolidate students grades and to students the opportunity to know their grade proximately, and also check their development in the course. For computer science courses this question type allows a teacher to create Java questions and test the awareness of students in Java programming course. The students type the source code for a specified interface in Java and the answer gets graded automatically. This way the teacher can save time for corrections, which is particularly important when enormous number of students visiting the course.

1.1 Competitive Information

There are numerous programming tools developed as web applications, where users can exercise their programming skills by coding a solution for the given problem. Some of these tools are CodingBat, betterprogrammer, Practice-It. One of the advantages of these structures is that the users get instant feedback about their answers. Even if a user fails to solve a problem at the very first attempt, an accurate and detailed feedback inspires him to re-attempt the problem until he reaches the correct solution. CodingBat permits the users to practice Java and Python coding online. It also allows students to create accounts and save their work. Students can share their CodingBat accounts with their teachers. It simplifies teachers to view their student's advancement. Only recently, CodingBat has released a problem authoring feature where users can add their own software design problems to the existing set. Practice-It is another online tool developed by Mark Stepp to practice Java programs.
1.2 Relationship to Other Applications/Projects.

Computer science educators make extensive use of programming assignments. One way to enhance the outcomes of the learning process is to oblige students to solve practical coding problems. But the evaluation of programming assignments is not straightforward. It involves time-consuming steps for compiling answers and testing them for the various inputs. This evaluation overhead confines the number of programming assignments an instructor can assign during a course period. One way of reducing this overhead is to automate the evaluation process. In this project I have developed a system that automates the grading of Java programming assignments. Many instructors and universities are adopting Learning Management Systems (LMS) such as Moodle or Blackboard. These tools effectively lessen the course management efforts. In this project, we are extending Moodle to carry out an automatic grade evaluation. This would facilitate teachers to use one integrated LMS that provides automatic grading of the programming assignments along with the other course management aspects.

1.3 Assumptions and Dependencies

In this section of Assumptions and Dependencies we have used VPL (Virtual Programming Language) activity. It features of editing, running and evaluation of programs makes learning process for students, and the evaluation task for teachers, easier than ever. We have downloaded VPL and we have plug-in in Moodle. We have given assignments in VPL activity as it only runs Java Programming Language.

Before the assignment is released, the instructor must spend time preparing the assignment, writing a solution program, generating scripts that will evaluate the student program, testing the complete setup, and incorporating all this as a new VPL activity in Moodle. After the assignment is closed, a brief review of the submissions will ensure that everybody received a grade, and a quick scan of submitted programs will detect fraudulent attempts. The VPL plug-in provides a similarity test to flag submitted programs that have a high level of similarity, a useful feature for spotting possible cheating. After this half-hour intervention, the instructor is basically free of the assignment.

1.4 Future Enhancements

The features provided by Moodle are sufficient for a standard course. But Moodle can be further improved to provide more advanced features. One of the feature that Moodle lacks is the ability to generate and grade programming questions. Moodle has ten standard built-in question types. But none of these standard question types provides a way to assess programming problems.

A MoodleMoot is a forum for Moodle community members, held to learn about Moodle, share knowledges of the learning platform, discusses research in related enlightening technologies and contribute ideas to future Moodle development. Held around the world, MoodleMoots are organized by universities or other large organizations using Moodle, Moodle Partners, Moodle associations or Moodle HQ.
With the current Moodle plugin, the instructors have to manually create each question. This job is tedious if the number of questions to be created are more, Moodle provides a question import feature that facilitates importing questions in bulk into the question bank. Moodle supports number of formats for importing questions. With the Moodle XML format, instructors can specify the problem information in an XML document and import multiple 34 questions into Moodle.

1.5 Definitions and Acronyms

Acronym items should be included here. For each special term supply a definition here.

- MOODLE- Modular object oriented dynamic learning environment.
- VPL- Virtual programming language.
- WAMP- Windows, Apache, MySQL, PHP
- PHP- Pre hypertext processor
- SQL- Sequential query language
- XML- Extensible markup language
- HTML- Hypertext markup language.
2 Project Technical Description

Moodle is primarily developed in WAMP Server which is a Windows web development environment. It allows to create web applications with Apache2, PHP and a MySQL database. Alongside, PhpMyAdmin allows to manage the databases easily.

The basic requirements for Moodle are as follows:

- **Hardware Requirements:**
  - Disk space: 160MB free (min) plus as much as you need to store your materials. 5GB is probably a realistic minimum.
  - Backups: at least the same again (at a remote location preferably) as above to keep backups of your site
  - Memory: 256MB (min), 1GB or more is strongly recommended. The general rule of thumb is that Moodle can support 10 to 20 concurrent users for every 1GB of RAM, but this will vary depending on your specific hardware and software combination and the type of use. 'Concurrent' really means web server processes in memory at the same time.

- **Software Requirements:**
  - Web server: Primarily Apache or IIS. Moodle will refuse to install on any other web server. Your web server needs to be correctly configured to serve PHP files. The version is not critical but try to use the newest web server build available to you.
  - PHP: The minimum version is currently 5.3.2. A number of extensions are required; see the PHP page for full details. Installation will halt at the environment check if any of the required extensions are missing.
  - Minimum browser for accessing Moodle: Firefox 4, Internet Explorer 8, Safari 5, Google Chrome 11, Opera 9 plus whatever plugins and applications you will need for the content you plan to use

2.1 Application Architecture

Java auto grader is designed to run on WAMP Server. Since we used Moodle web framework, the website is composed of many Moodle applications. Moodle provides a common interface for all databases. Programmers do not have to keep in mind a specific database while writing code. Moodle applications designed for Java Auto Grader are listed below:
2.1.1 Courses

Course application creates and manage courses. It may seem that course is tightly coupled with Assignment and an Assignment cannot exist without a course. We, still, decided to make a separate application for Course. Our purpose is to make a web application that supports creation and evaluation of Assignments. Managing Course with separate application allows us to design it independently.

In initial release the Course application implements nine views. These are list in table 2.1.1

<table>
<thead>
<tr>
<th>View</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create Course</td>
<td>Creates a new Course. A user who creates Course automatically becomes instructor of the Course.</td>
</tr>
<tr>
<td>Edit Course</td>
<td>Modifies Course Details. Only instructor of a Course can modify its details.</td>
</tr>
<tr>
<td>Delete Course</td>
<td>Deletes a Course. Only instructor of a Course can delete a Course. The operation also deletes all information that is associated with it.</td>
</tr>
<tr>
<td>Course Details</td>
<td>Display a course details. All users can see the details.</td>
</tr>
<tr>
<td>List all Courses</td>
<td>Displays all available courses. All users can see the list.</td>
</tr>
<tr>
<td>Search Course</td>
<td>Perform search operation using course code or course name as a search key. All users can search.</td>
</tr>
<tr>
<td>Join Course</td>
<td>Registers a user in a course. All users can register for any course except instructor of the course.</td>
</tr>
<tr>
<td>Leave Course</td>
<td>Deregister a student from a Course.</td>
</tr>
<tr>
<td>List Students</td>
<td>Displays all users that are registered for a Course. Only instructor of a Course can access this information.</td>
</tr>
</tbody>
</table>

2.1.1: Views of Course Application
2.1.2 Assignments

This is one of the main applications. This application provides all features necessary for three level hierarchy of an Assignment. Its dependencies are course, evaluate and upload applications. The application provides following operations in table 2.1.2

<table>
<thead>
<tr>
<th>View</th>
<th>Description</th>
<th>Who Can Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create Assignment</td>
<td>Creates new Assignment of a Course.</td>
<td>Instructor</td>
</tr>
<tr>
<td>Edit Assignment</td>
<td>Modifies details of an existing Assignment.</td>
<td>Instructor</td>
</tr>
<tr>
<td>Assignment details</td>
<td>Displays details of an Assignment.</td>
<td>All users</td>
</tr>
<tr>
<td>Delete Assignment</td>
<td>Delete an Assignment. This operation will delete all associated Sections, Testcases and submissions of the Assignment</td>
<td>Instructor</td>
</tr>
<tr>
<td>Create Section</td>
<td>Adds a new Section in an Assignment.</td>
<td>Instructor</td>
</tr>
<tr>
<td>Edit Section</td>
<td>Modifies details of a Section</td>
<td>Instructor</td>
</tr>
<tr>
<td>Section details</td>
<td>Displays details of a Section</td>
<td>All users</td>
</tr>
<tr>
<td>Delete Section</td>
<td>Delete a Section. It will also delete all associated Testcases.</td>
<td>Instructor</td>
</tr>
<tr>
<td>Create Testcase</td>
<td>Adds a new Testcase to a Section</td>
<td>Instructor</td>
</tr>
<tr>
<td>Edit Testcase</td>
<td>Modifies details of a Testcase.</td>
<td>Instructor</td>
</tr>
<tr>
<td>Testcase Details</td>
<td>Displays details of Testcases</td>
<td>All users</td>
</tr>
<tr>
<td>Delete Testcase</td>
<td>Delete a Testcase.</td>
<td>Instructor</td>
</tr>
</tbody>
</table>
### 2.1.3 Evaluate

This application evaluates a submission. It gets necessary data from assignment application. A separate module is designed to compile and run a program. Evaluate application has three dependencies “assignment”, “evaluate” and “upload”. Operations provided by this application are listed in table 2.1.3

<table>
<thead>
<tr>
<th>Evaluate Assignment</th>
<th>Evaluate submission of an Assignment and display its result.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Check Output</td>
<td>If solution code is available then student can upload some input and check output produced by solution</td>
</tr>
<tr>
<td>Show results</td>
<td>Show results stored in database without re-evaluating a submission.</td>
</tr>
<tr>
<td>Run practice testcases</td>
<td>Run practice Testcases on a submission and display results.</td>
</tr>
<tr>
<td>Evaluate all submissions</td>
<td>This view evaluates all submission of an Assignment. Instructor can evaluate all submissions with one click.</td>
</tr>
<tr>
<td>Evaluation details of a submission</td>
<td>This view displays assignment result in a different from than “Evaluate Assignment” view</td>
</tr>
<tr>
<td>Evaluation details of an assignment</td>
<td>Provides a summary of Assignment results to instructor. It displays total number of failed Testcases and compiler errors.</td>
</tr>
</tbody>
</table>

Table 2.1.3: Views of Evaluate Applications

### 2.1.4 Download

Generally, any file can be accessed by a direct link to file from server’s root directory. Sometimes we need to do some processing before sending a file to the browser e.g. checking if a user has permissions to access the file or compressing a file to save bandwidth. Download application is designed to serve these purposes. As per initial release this application has only two vies. First is “Download All Submissions” which provides access to all submissions of an Assignment with a single click. All files are compressed in a single file before sending to client. Only instructor of a Course can download all files. Second view is “Download Assignment files”. Assignment,
Sections and Test cases have many files associated with them. This view provides single click access to all files. This application does not have any database table.

### 2.1.5 Upload

Assignment submissions are handled by this application. In first release, it looks like an unnecessary application because it has only one view to upload a submission which could be integrated to Assignment application. In future versions we may need more upload views. We can modify “upload submission” view to make it a generic view thus improving re-usability. There are two more views “Show all submission of assignment” and “Show submission of a user”. First view displays all submissions of an Assignment. Only instructor can access this view. Second view displays all submission by a user of a course. This view is used by a student to see all his/her submission in a single view.

### 2.2 Application Information flows

Java Auto Grader has many entities. In this section we focus on workflow of two main entities i.e. workflow from instructor’s perspective and workflow from student’s perspective.

#### 2.2.1 Workflow from Instructor’s Perspective

- **Create Course:**
  
  Course entity helps in organization of Assignment. It provides a level of abstraction whereby an instructor can associate a number of assignments with a Course. This abstraction also reduces search space of an ‘Search Assignment’ operation. If we know that an Assignment is associated with a Course, then it will be searched only within that Course. Students will see all Assignments available in system irrespective of their interest. As per current design, any user can create a Course. A user who creates a Course becomes instructor of that Course. Only an instructor can edit details of a Course and add Assignments to it.

- **Add Assignment:**
  
  An Assignment is always associated with a Course and is created in VPL activity. If an Assignment is newly created then it will not have any Section associated with it and therefore no further processing will be done. We can edit details of any existing Assignment. This operation is same as add Assignment but in case it has a Section associated with, it behaves differently. In this case when an instructor edits Assignment details, newly provided details must be in accordance with the details of its Sections.

- **Add Section:**
  
  Section is second level component of Assignment. After successful creation of a new Assignment to a Course, instructor can add a Section to the Assignment. At the time of creating new Section, Java Auto
Grader automatically verifies Section details with Assignment Details. Any discrepancy is reported immediately to the user.

- **Add Testcase:**
  This is the last step of Add Assignment process. A Testcase is added to a Section. It tells how to run a program created by its Section. In testcases we store Input files that are read by program. Content of one file provided here will be used as standard input. This file will be marked at the time of creation of Testcase and Output files that are produced by program. Output written on standard output by the program is stored in a file. This file is marked at the time of creation of Testcase.

- **Evaluate Assignment:**
  Evaluate Assignment process evaluates a given submission according to the configuration provided in Assignment specification. Currently Java autograder can evaluate programming assignments written in only java programming language. When we run the test case, if output produced by the program matches with expected output then Testcase is passed and marks corresponding to that Testcase are allocated to student. If output does not match, then Testcase is failed and zero marks awarded. Any error generated at run time will be reported to students.

2.2.2 **Workflow from student’s perspective**

Any user who joins a Course becomes a student of that Course. Now the user can submit his/her code for an Assignment of that Course.

- **Join Course:**
  Java autograder maintains two separate list of courses for each user. One list has all courses created by a user and another list has all courses that a user registered for. A user can join only those courses that are not created by him/her. All users can see course details and all its Assignments but no one can submit solution for any Assignment unless the user joins the course. Join course operation registers the user for that course.

- **Upload Assignment:**
  A user can submit or upload his/her solution code of an Assignment before its deadline is reached. Java autograder accepts any file which is maximum of 3mb. The system checks for all files that must be submitted for the Assignment. This information is taken from the field “List of file name to submit” of Assignment. Submissions are accepted only before deadline.
Run Practice Testcases:
All Testcases that are associated with a Section of type “Practice” are executed by ‘run practice testcases’ operation. These are special testcases and are used by students only to check output format of their programs. Student can run practice testcases immediately after uploading the submission in the Java auto grader.

2.3 Interactions with other Applications
Java auto grader does not interact with any other applications.

2.4 Capabilities
VPL (Virtual Programming Language) activity has the capability that needs to support the java auto grader project. VPL is a Moodle plug-in, and requires a dedicated separate execution server, or jail server for short. This jail server runs the test scripts on the programs submitted by the students.

VPL- Virtual Programming Lab is an activity module that manage programming assignments and whose salient features are:

- Enables to edit the programs source code in the browser
- Students can run interactively programs in the browser
- You can run tests to review the programs.
- Allows searching for similarity between files.
- Allows setting editing restrictions and avoiding external text pasting.

2.5 Risk Assessment and Management
The risk assessment and management of java auto grader project are listed below

- Make sure you upgrade your site often. Recent versions of Moodle have new fixes and warnings that will help you avoid security issues.
- Consider the spam risks involved in allowing certain capabilities for visitor accounts, such as replying forum posts or posting to blogs.
- Limit self-registration to particular email domains with the allowed email domains setting or deny email addresses from particular domains, such as mailinator.com and temporaryinbox.com, with the denied email domains setting. Both settings are in Administration > Site administration > Users > Authentication > Manage authentication common settings.

- Consider only enabling self-registration for a short period of time to allow users to create accounts, and then later disable it by setting 'Self registration' to Disable in the common settings in Administration > Site administration > Plugins > Authentication > Manage authentication.

- Keep "Email change confirmation" enabled in Administration > Site administration > Security > Site policies.
3 Project Requirements

3.1 Identification of Requirements

The identification of requirements of Java Auto Grader are WAMP Server and Moodle. The installation method for Moodle are explained in step wise process

*Step 1*: Installing WAMP Server 64 bit. WAMP includes Windows, Apache, MySQL, PHP software’s.

![WAMP Server Installation](image)

*Step 2*: We have downloaded Moodle 3.0.3 from the official site and we have saved Moodle as our project. When we open localhost in the web page we can see Moodle under Your Projects.
**Step 3:** After installing Moodle we have to plugin for OAuth 2.0 for google sign in.

**Step 4:** we have even installed VPL plug in for java programming assignments, where you can view in plugins overview.
3.2 Operations, Administration, Maintenance and Provisioning (OAM&P)

In Java auto grading project, the features in the administration allow teachers to manage Course settings, student and instructor enrollments and their groups, it even allows to view the course gradebook, create custom grading scales and access the instructor forum. The instructor forum is a private forum only available to instructor of that course. It can be used to discuss the course content the direction the course could take or even to attach files to that can be shared among the course instructors.

Instructor can upload any digital content for inclusion in an activity or as a resource for viewing or downloading. Files can be moved, renamed, or deleted. Some files can be edited directly. Instructor can also create a directory and display the whole contents of that directory to course students using the "add resource" drop down in any course section. Of course if your content resides out on the web then you don't need to upload the files at all by using the "add resource" menu or creating an HTML link on a content page.

Maintenance mode is for preventing any users other than administrators from using the site while maintenance is taking place, though it's not designed to prevent user access during version upgrades. When users attempt to access a course when your site is in maintenance mode, they obtain a message informing them that the site is in maintenance mode. If you wish, you can create a customized maintenance mode message, perhaps stating when the site will be available again or giving the reason for doing maintenance.

An administrator can put the site into maintenance mode in

\[ Administration \rightarrow Site administration \rightarrow Server \rightarrow Maintenance mode. \]
When a site is in maintenance mode, the link "In Maintenance Mode" is displayed near the top right of each page except the front page for admins.

### 3.3 Security and Fraud Prevention

During dynamic tests unknown and potentially faulty source code will be executed. Thus, certain security aspects have to be considered. Without precautions, a submission can execute with the privileges of the backend user any functions and programs, run denial of service attacks, or spy on information about other users, the system, or assignments especially the model solution. This results in security requirements that have to be taken into consideration in later deployment of the backends, depending on programming language and platform. For instance, a restricted interpreter or a sandbox environment can be used for program execution. Other possibilities include the deployment of additional software to limit the access to system calls. Furthermore, it is reasonable to set a time limit for testing the submitted programming code. If this time limit is exceeded, the execution of the current submission is aborted because the code is suspected of containing infinite recursions or infinite loops.

A password policy may be set up in Settings > Site administration > Security > Site polices

There is a check box to determine if password complexity should be enforced or not, the option to set the minimum length of the password, the minimum number of digits, the minimum number of lowercase characters, the minimum number of uppercase characters and the minimum number of non-alphanumeric characters. If a user enters a password that does not meet those requirements, they are given an error message indicating the nature of the problem with the entered password. Enforcing password complexity along with requiring users to change their initial password go a long way in helping ensure that users choose and are in fact using "good passwords". However, making the check too onerous just results in them writing it down so be realistic.

### 3.4 Release and Transition Plan

Moodle major releases with big new features are now on a regular 6-month cycle, in May and November. Each major release increments the version number by 0.1 (e.g. 2.4 -> 2.5 -> 2.6) and starts a new branch of minor releases.

Minor releases with bug fixes only are now on a 2-month cycle, unless a security emergency occurs. They will increment the major release by 0.0.1 (e.g. 2.5 -> 2.5.1 -> 2.5.2).

In this java auto grader, we are using Moodle 3.0.3 This package is built every week with new fixes produced by our stable development process. It contains a number of fixes made since the 3.0.3 release and is usually a better choice for production.
4 Project Design Description

Java auto grader project is mainly for java programming assignments so we have divided an assignment into three parts. We call these parts Assignment, Section and Testcase. This is a three level hierarchical structure. The Assignment is a high level entity it can have one or more Sections associated and each Section can have one or more Testcases.

Figure 4.1: Roles and actions done by an instructor
Assignment

Here in this section we have to add an Assignment in VPL activity, Assignment defines problem statement and all other details that are not specific to a particular compilation of code and execution of it. These details include deadlines of Assignment, programming language used to write code, some code template and name of files that must be submitted for this assignment etc.

Figure 4.2: Roles and actions done by students

VPL Setup and Operation VPL is a Moodle plug-in, and requires a dedicated separate execution server, or jail server for short. This jail server runs the test scripts on the programs submitted by the students. Should a student program crash the jail server, the Moodle server is unaffected. The instructor defines how the student program is...
evaluated and graded. This allows for testing properties of a program other than its. For example, in assembly language, it may be important for an assignment to generate a program with as small a memory footprint as possible. The instructor can create a script that will measure the static footprint of the student program and assign a grade inversely proportional to the program’s byte size. The instructor can define the rubric under which a VPL grade is assigned. This is controlled per VPL-assignment. The instructor can make the grade visible to the student, or not. In the latter case, the grade is revealed after the due date.

The instructor can control the resources needed by the jail server. For a given VPL activity, the programs submitted can be those of an individual student, or from a group of students. Access to the plug-in, including submission, can be restricted by IP address. The instructor can enforce for programs to be typed by hand in the submit window, and disable copy/paste of program code.

We have to upload an assignment in VPL activity as test 1 which can be viewed in the below screenshot as follows

![Screenshot of VPL activity](image)

**Testcase**

At this level, we store all information that is needed to execute the binary that was produced in a Section. A binary can accept command line arguments, some input from standard input. It might also read a file, write output on standard output and write some files. We provide input and output information required by a program on this level. We evaluate student’s program on the basis of output produced by it. This output is compared with correct output
which may be different for each Testcase. Therefore, we store the correct output or expected output on Testcase level. The following testcase below is given for an assignment that is to write a program that adds two numbers

Evaluate

When a student submits the assignment in java auto grader he/she gets automatic grade depending on his/her performance accordingly. The grade varies from 0 to 100 if the student does not make any error he/she scores 100 just on submitting the assignment and will pass the test, if else the student scores less grade i.e. 0 he fails in the test and can see the errors and comments automatically on the right corner of the test page. The following screenshots are shown below
Java programming

Automatic evaluation

WebPing.java
5 Internal/external Interface Impacts and Specification

5.1 Frontend Specification:

The basic idea is that instructors create online assignments, into which students submit their answers or solutions. These submissions are stored as assignments. Each submission which in turn, passes it on to the selected backend. The assessment of student submissions starts with the submission of a student’s answer and ends with the grading of an assignment by the instructor. This process can be modeled as a workflow, i.e., from the initial submission to the final grading, submissions are put through a number of workflow states. When an instructor creates a new assignment, it has to be associated with a certain backend.

Therefore, the assignment has to communicate directly which returns a list of available backends and their input fields that are needed for testing. Those input fields may vary according to the chosen backend and the assignment has to dynamically create the user interface, i.e., for a testcase based backend, the instructor has then to type in test data and a model solution, whereas we have to specify unit tests for respective unit test backends. Students can read the assignment text and submit their answers. If the submission period is restricted, submissions are only allowed until the submission period has ended. Multiple attempts to answer assignments are allowed up to the maximum number of attempts specified by the instructor.

5.2 Backend Specification:

For the development of new backends, submissions should be tested dynamically with unit tests, this requires the availability of a unit test framework for the chosen language. Furthermore, the conditions that lead to an early termination of the test run have to be determined. For instance, dynamic tests need not be run if a syntactical test has
already failed. It could also be defined that a test run comprising a number of tests should be terminated as soon as a
single test has failed. The chosen test method determines requirements for the input data, i.e., information that has to
be given by the instructor and information that is enclosed in the actual submission. In most cases, a student’s
submission contains the answer to an assignment in one or more text or source code files.

The instructor provides information about the constraints of the assignment. For enabling automatic testing of the
submission in addition to the description of the problem to solve the learner has to have the following information:
name of the functions we have to implement, number and types of function arguments, type of the return value. The
instructor provides the information necessary for testing. For a comparison with a model solution, it is necessary to
have such a model solution and also test data (input data). In this case, the instructor also has to set whether the
results of the submission should be identical to those of the model solution or if in case of a list valued result,
permutations are allowed as well. In addition to input data, the output of the backend must also be defined exactly
since this value will be forwarded as feedback to the learner. If all tests have been passed successfully, this should
result in a positive feedback. In the case of errors, there has to be a detailed feedback about the type of error such
that the student gets hints about the problems with his solution.

6 Design Units Impacts

The Functional areas of Java auto grader are the WAMP (Windows, Apache, MySQL, PHP) Server and Moodle. In
our project we have multiple design units i.e. one for web interface and one for our application, here WAMP Server
is the web interface and Moodle is the application for our Java auto grader project.

6.1 Functional Area A/Design Unit A

6.1.1 Functional Overview

Moodle is the application of java auto grader. Moodle is an online learning and course management tool. Moodle
stands for “Modular Object Oriented Dynamic Learning Environment”. Moodle is one of the most popular and
comprehensively used LMS today. At the time of this writing, Moodle has around 35 million users and 50,000
registered sites. In this project, we have developed a plugin for Moodle that facilitates the automatic grading of Java
programming assignments.

Here are the reasons why we chose Moodle for this project:

• Moodle is open source, stable and it has a good community support.

• The Moodle installation is very straightforward and it is easy to use.

• Moodle is developed on modular architecture. Hence it is easy to extend or customize for your own needs.

As discussed, the features provided by Moodle are sufficient for a standard course. But Moodle can be further
improved to provide more advanced features. One of the feature that Moodle lacks is the ability to generate and
grade programming questions. Moodle has ten standard built-in question types. But none of these standard question
types provides a way to assess programming problems.
6.1.2 Impacts

Java auto grader project does not have any impacts in the Moodle application.

6.1.3 Requirements

The requirements can be split into two perspectives i.e. instructors and students. At very high level, we describe the system that supports creation of an assignment, submission of an assignment and evaluation of submitted assignment. Detailed functional requirements of the system are as follows.

➢ Instructor’s Perspective

- Assignment creation and submission operations are to be performed by instructor and students respectively. We need a feature that can identify a user. In other words, we need a user login authentication and registration feature.
- Any user should be able to create a Course. In a Course instructor should be able to create an Assignment.
- For an Assignment instructor should be able to create Section. An assignment can have multiple Sections. A Section is an entity which stores necessary information to generate an executable program. There can be multiple Sections per Assignment.
- A section can be of two type ‘Practice’ and ‘Evaluate’. Practice tests will be used to verify output format by students and evaluation type will be used to grade the assignment.
- A program of a Section may need some input and may produce some output. This information is separately stored in an entity which is known as Testcase. An instructor should be able to create Testcase for a Section. There can be multiple Testcases per Section.
- Instructor should be able to see all submissions of his/her Course.
- Instructor should be able to run any Testcase that she/he created.
- Instructor should be able to modify Course/Assignment/Section/Testcase.
- If instructor modifies any of the entity that makes a submission inconsistent with the new detail, it should notify student to re-submit.
- Instructor should be able to enforce deadline based submission.
- Instructor should be able to run “evaluate all” operation which will run ‘Evaluate’ type Testcases of all submissions.

➢ Student’s Perspective

A student needs features to submit his/her solution code and to see result of this code online. Detailed requirements are as follow

- Any user other than instructor should be able to register(join) for a Course.
• User who joins a course will be considered as a student of that Course.
• Student should be able to see Assignment details.
• Student should be able to upload Assignment solution.
• Check on submission will be run to verify directory structure, file names and to ensure that all files are present in submission.
• Student should be able to run practice testcases before and after deadline and after the deadline student should be able to run ‘Evaluate’ testcases.
• After each run of testcases student should be able to see detailed results of the run. This include any error during compilation of program and run time error of program. Error messages must be detailed enough to help student understand the problem.
• Student should be able to check output of a program on any input if solution code is uploaded by instructor. This input will be given by student.

6.2 Functional Area B/Design Unit B

6.2.1 Functional Overview

WAMP Server is the web interface in java auto grader project. WAMP Server refers to a software stack for the Microsoft windows operating system, created by Romain Bourdon and consisting of the Apache Web Server, MySQL database and PHP programming language. The modularity of a WAMP stack may vary, but this particular software combination has become popular because it is entirely free and open-source software. This means that each component can be interchanged and adapted without overt vendor lock-in, and that the complete software stack is available free of cost. The components of the WAMP stack are present in the software repositories of most WAMP stack with some automation.

Apache is developed and maintained by an open community of developers under the auspices of the Apache software foundation. Released under the Apache license, Apache is open source software A wide variety of features are supported, and many of them are implemented as compiled modules which extend the core functionality of Apache. These can range from server-side programming language support to authentication schemes.

MySQL is a multithreaded, multi-user, SQL database management system (DBMS), acquired by Sun microsystems in 2008, which was then acquired by Oracle Corporation in 2010. Since its early years, the MySQL team has made its source code available under the terms of the GNU General Public License, as well as under a variety of proprietary agreements.

PHP is a server side-scripting language designed for web development but also used as a general-purpose programming language. PHP code is interpreted by a web server via a PHP processor module, which generates the
resulting web page. PHP commands can optionally be embedded directly into an HTML source document rather than calling an external file to process data. It has also evolved to include a command-line interface capability and can be used in standalone graphical applications.

6.2.2 Impacts

Java auto grader does not have any impacts in the web interface of WAMP Server.

6.2.3 Requirements

Specific solutions are required for websites that serve large numbers of requests, or provide services that demand high uptime. High-availability approaches for the WAMP stack may involve multiple web and database servers, combined with additional components that perform logical aggregation of resources provided by each of the servers, as well as distribution of the workload across multiple servers. The aggregation of web servers may be provided by placing a load balancer in front of them. For the aggregation of database servers, MySQL provides internal replication mechanisms that implement a master/slave relationship between the original database and its copies.

Such high-availability setups may also improve the availability of WAMP instances by providing various forms of redundancy, making it possible for a certain number of components separate servers to experience downtime without interrupting the users of services provided by a WAMP instance as a whole. Such redundant setups may also handle hardware failures resulting in data loss on individual servers in a way that prevents collectively stored data from actually becoming lost. Beside higher availability, such WAMP setups are capable of providing almost linear improvements in performance for services having the number of internal database read operations much higher than the number of write/update operations.
7 Open Issues

Issue tracking is an important part of a continuous quality control process. It involves reporting of problems that is bugs, ideas for improvement and new features. Unlike most proprietary software programs, Moodle issue reporting and tracking information is open to everyone. Moodle's issue tracking system is called the Tracker.

All Moodle users are encouraged to be active participants when it comes to testing. Anyone with a Tracker user account can create, view, comment on, vote, and watch bugs.

There are no issues in java auto grader project.
8 Acknowledgements

I owe a great many thanks to Dr. Clare Tang for trusting us with her idea. This project would not have been possible without her guidance and support. One simply could not wish for a more knowledgeable and caring adviser.
9 References

https://moodle.org/


http://scholarworks.sjsu.edu/cgi/viewcontent.cgi?article=1050&context=etd_projects

https://en.wikipedia.org/wiki/Moodle

https://en.wikipedia.org/wiki/LAMP_(software_bundle)#WAMP

https://docs.moodle.org/30/en/Main_page

https://moodle.org/plugins/mod_vpl