<table>
<thead>
<tr>
<th>TOPIC</th>
<th>PAGE NUMBER</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Introduction</td>
<td></td>
</tr>
<tr>
<td>1.1 Purpose</td>
<td>3</td>
</tr>
<tr>
<td>1.2 Scope</td>
<td>3</td>
</tr>
<tr>
<td>1.3 Definitions, acronyms, and abbreviations</td>
<td>4</td>
</tr>
<tr>
<td>1.4 References</td>
<td>5</td>
</tr>
<tr>
<td>1.5 Overview</td>
<td>5</td>
</tr>
<tr>
<td>2. Overall description</td>
<td></td>
</tr>
<tr>
<td>2.1 Product perspective</td>
<td>6</td>
</tr>
<tr>
<td>2.2 Product functions</td>
<td>6</td>
</tr>
<tr>
<td>2.3 User characteristics</td>
<td>6-14</td>
</tr>
<tr>
<td>2.4 Constraints</td>
<td>14</td>
</tr>
<tr>
<td>2.5 Assumptions and dependencies</td>
<td>14</td>
</tr>
<tr>
<td>3. Specific requirements</td>
<td></td>
</tr>
<tr>
<td>3.1 External interface requirements</td>
<td>14</td>
</tr>
<tr>
<td>3.1.1 User interfaces</td>
<td>14</td>
</tr>
<tr>
<td>3.1.2 Hardware interfaces</td>
<td>14</td>
</tr>
<tr>
<td>3.1.3 Software interfaces</td>
<td>14</td>
</tr>
<tr>
<td>3.1.4 Communications interfaces</td>
<td>14</td>
</tr>
<tr>
<td>3.2 Functional requirements</td>
<td></td>
</tr>
<tr>
<td>3.2.1 User class 1-Instructor</td>
<td>15-19</td>
</tr>
<tr>
<td>3.2.2 User class 2-Student</td>
<td>19-21</td>
</tr>
<tr>
<td>3.2.3 Diagrams</td>
<td></td>
</tr>
<tr>
<td>3.2.3.1 Use case diagram</td>
<td>23-39</td>
</tr>
<tr>
<td>3.2.3.2 Activity diagrams</td>
<td>40-43</td>
</tr>
<tr>
<td>3.2.3.3 Sequence diagrams</td>
<td>44-48</td>
</tr>
<tr>
<td>3.2.3.4 Data flow diagrams</td>
<td>49-50</td>
</tr>
<tr>
<td>3.3 Performance requirements</td>
<td>50</td>
</tr>
<tr>
<td>3.4 Design constraints</td>
<td>50</td>
</tr>
<tr>
<td>3.5 Software system attribute</td>
<td>50</td>
</tr>
<tr>
<td>3.6 Other requirements</td>
<td>50</td>
</tr>
</tbody>
</table>
1. Introduction

1.1 Purpose
The purpose of Software Requirements Specification (SRS) document is to describe the external behaviour of the Peer Evaluation System. Requirement specification defines and describes the operations, interfaces, performance, and quality assurance requirements of the Peer Evaluation System. The document also describes the non-functional requirements such as the user interfaces. It also describes the design constraints that are to be considered when the system is to be designed, and other factors necessary to provide a complete and comprehensive description of the requirements of the software. The Software Requirements Specification (SRS) captures the complete software requirements for the system or a portion of the system.

1.2 Scope
The Peer Evaluation System that is to be developed provides an evaluation mechanism for various Massive Open Online Courses (MOOCs).

- The system can be used for uploading assignments by instructors, and will allow students to write and evaluate solutions.
- The assignments that will be submitted, will be reviewed by the peers of students taking the course, and scores will be awarded accordingly. Incentives will be given to each student for good evaluation which will be accounted for in the final grade.
- The instructor can design the course by setting deadlines and the number of reviewers he wants for each assignment. He can choose from the following types of evaluation systems:
  1. Peer Evaluation - Each assignment will be reviewed by a number of other students.
  2. Self Evaluation - Each student will be asked to evaluate his own assignment.
  3. Autograding - This will be done for program codes, by available softwares.
  4. Instructor Evaluation - In this, the instructor himself will grade all the papers.
  5. A combination of any of the above methods, with weighted percentage to each technique.

1.3 Definitions, acronyms, and abbreviations
• MOOC - A massive open online course (MOOC) is an online course aimed at large-scale interactive participation and open access via the web.
• Instructor - Instructor is the teacher who is teaching the course. They upload assignments, check the solutions, review results and deliver grades to students.
• Student - A student is someone who is enrolled in a course.
• Author - Author is a student who writes assignments and submits them for evaluation.
• Reviewer - Reviewer is a student who evaluates the solutions.

1.4 References


1.5 Overview

The remainder of this document is two chapters, the first provides a full description of the project for the users of Peer Evaluation system and final chapter concern the details and functions performed by the system.
2. Overall description

2.1 Product Perspective
The Peer Evaluation System that we are creating is a web based tool. The software is to be used in association with a MOOC. The MOOC is used for uploading tutorials and enrolling students for various courses. The assignments for the course are to be provided on the Peer Evaluation System. Students will take up an assignment, and submit solutions. These solutions will then be evaluated by other students/instructor or himself and scores are given. The reviewer will get an incentive for evaluation which will then be accounted for in his final grade.

2.1.1 System interfaces

2.1.2 User interfaces
There are two different users of this software:

- Instructor who can create courses and upload/create assignments.
- Students who can write assignments and evaluate others’ solutions.

The features that are available to the instructor are:-

- Can upload /create assignments.
- View the list of enrolled students, their assignments and grades.
- Design the course choosing different evaluation schemes.
- Evaluate reviews and change the grades if he deems fit.

The features that are available to the student are:-

- View the list of courses and enroll in any course.
- Write assignments.
- Review assignments once the deadline for submission is over.
- View previous assignments and their solutions.
- Report wrong grading.
- Unenroll from any course.
2.1.3 Hardware interfaces

2.1.4 Software interfaces
The PEAS evaluation system is to be built on edx-platform which uses django framework.
The following are the different software interfaces-

a. LMS
   It is the edx platform which a student can access. Here the student submits the assignment solutions.

b. Xqueue
   It is an interface between the LMS and the external grading services.
   The solutions submitted by the students are transferred from LMS to Xqueue which then sends these to the grading services.

c. External graders
   These are the different grading modules which can be active or passive. These Modules grade the assignment and send back the response to Xqueue.

   ORA - edX ORA (Open Response Assessor) allows for the peer, instructor, and AI assessment of problems on the edX platform.
   Ease - EASE (Enhanced AI Scoring Engine) is a library that allows for machine learning based classification of textual content. This is useful for tasks such as scoring student essays.
   Xserver- It accepts student code submissions from the LMS and runs the code using courseware graders
   Discern- Discern allows anyone to use machine learning based automated textual classification as an API service.

2.1.5 Communications interfaces

2.1.6 Memory

2.1.7 Operations

2.1.8 Site adaptation requirements
2.2 Product functions
The Assessment System that is being developed is to be used in integration with a MOOC. The functionality of our system is fourfold,
1. It will provide peer evaluation.
2. It will allow self assessment.
3. There will be an option for using existing auto-grading softwares to check program codes.
4. The instructor of the course has the option to evaluate the papers himself.
Also, a combination of the above mentioned assessment mechanisms can be used, providing weightage to scores from each method.

1. Peer Evaluation:
In peer assessment, we develop a model that reliably measures the performance of students without the need for expert’s intervention. This system also gives the opportunity to students to learn by playing the role of both “student” and “teacher”. By reading other students’ answers every student learns alternate ways to solve a problem.
The following process is followed:

(i) Calibration:
The reviewer in the peer evaluation scheme needs an incentive to evaluate the assignment correctly. If he evaluates the paper accurately, he will get a good incentive, and not a good one otherwise. The calibration method is employed in the beginning when no user profile is maintained. Before the first assignment is to be evaluated, we need to divide the students in groups so that each paper is evaluated by a number of reviewers of different reviewing abilities. The ability of the reviewer to evaluate correctly is corresponding to the incentive he receives. The incentive is a number from 0 to 1.

For that, we need a calibration mechanism in the beginning of the course. There are three kinds of mechanisms, and the instructor of the course will make the decision on which method to use.
a) Peer Calibration:
In this method, the instructor provides an assignment and an arbitrary solution to that assignment to all the students enrolled in the course. Every student is asked to review the solution according to a set of questions specified in the feedback form, and also give a holistic rating on a scale of 1 to 10. The average of all the ratings is calculated. Let it be x. Let the rating provided by a student be x’. Also, let the incentive he receives be denoted by c. Then,
c = 1 - |x-x’|/MM, where MM is the maximum marks viz 10.

b) Expert Calibration:
Here, the instructor will provide an assignment and solution and will evaluate the paper himself and give a score on a scale of 1 to 10. He then asks all the students to evaluate the same paper and give their score on 10. Let x be the rating provided by instructor and x’ be the score provided by a student. Then, the incentive c received by each student is calculated in the same way,
c = 1 - |x-x’|/MM, where MM is the maximum marks viz 10.

c) No Calibration:
This mechanism is used if the instructor does not want to give anyone an incentive at the beginning of the course. Here, every student is considered to have 0 incentive.

(ii) Division in groups:
The students are distributed into groups so that there is uniformity in the different reviews that each student gets. The list of all students, ordered by the incentive they received (from high to low), is then divided into as many groups as there should be number of reviewers for each assignment. If there is more than one student with the same incentive, they are ordered alphabetically. For example, if there are 100 students and we need 4 reviewers for each assignment, the students are ranked according to the incentive they received. Then 4 (number of reviewers) groups are created, and first 25 students are placed in Group 1, next 25 in Group 2, next 25 in Group 3 and the last 25 in Group 4.

Number of groups = Number of reviewers for each assignment

(iii) Review Process:
For every student’s assignment, a student reviewer is selected at random from all the groups. The selection is at random, and makes sure that the student himself is not selected from his own group to evaluate his paper. Each student is informed on his page of the assignments that he has to evaluate. A feedback form is provided which states the characteristics on the basis of which evaluation is done. A holistic rating is also provided on a scale of 1 to 10.

(iv) Normalisation:
To ensure fair assessment, we need a mechanism that takes into account that every student might not correctly grade his peer. We use a normalisation method that which is used to eliminate problems like student being generous and biased in giving scores. Here, mainly 4 cases are taken into consideration:

a) Normal Case:
In this case, the marks given by peers are considered to be fair.

b) Over-generous Case:
A student taking up peer assessment can award more marks than necessary to his peers, and a quid-pro-quo arrangement can be made.

c) Creative-accounting Case:
Here, under this case a student gives less marks to all his peers, in order to get a higher rank.

d) Penalised Case:
Another peer assessment scenario is where the majority of the group members decide to go against one person. In this case all the peers decide to give low marks to a specific student. This could be a valid case when the member did not contribute much to the project or when he actually deserves low marks.

A few terms are explained further which are used to calculate normalised marks of a student.

Individual Effort Rating(IER):
This is the summation of the marks received by a student from all his peers.
\[ \text{IER} = \sum M_i \]  where \( M_i \)= marks given by student i.
Average Effort Rating (AER):
This is the summation of the IER of each student divided by number of students in consideration.
\[ \text{AER} = \frac{\text{IER}}{i} \] where \( i \) is the number of students

Individual Weighting Factor (IWF):
\[ \text{IWF} = \frac{\text{IER}}{\text{AER}} \]

Bias Factor:
This factor tells us how much bias has been done by a particular student.
\[ \text{Bias Factor} = \frac{\text{Rating given to others}}{\text{AER}} \]

Normalisation Factor:
If bias factor (calculated for each student individually) doesn’t come out to be in the range \([0.98, 1.02]\), then the normalisation factor (for each student) is calculated and multiplied with marks awarded by that student.
\[ \text{Normalisation Factor} = \frac{1}{\text{Bias Factor}} \]

We now have a table with updated marks and this process is repeated until all the bias factors fall in the required range.

Let us take an example of 6 students: A, B, C, D, E and F. Suppose that the instructor wants each assignment to be evaluated by 3 students, which means the students are to be divided into 3 groups (2 students in one group) after ordering them by incentive. Assume that the groups are created as: A, C in Group1; B, F in Group2; D, F in Group3.

<table>
<thead>
<tr>
<th>Reviewer Author</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>IER</th>
<th>IWF</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>8</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td></td>
<td></td>
<td>22</td>
<td>1.27</td>
</tr>
<tr>
<td>B</td>
<td>6</td>
<td></td>
<td>7</td>
<td></td>
<td>3</td>
<td></td>
<td>16</td>
<td>0.92</td>
</tr>
<tr>
<td>C</td>
<td>9</td>
<td>8</td>
<td></td>
<td>3</td>
<td></td>
<td></td>
<td>20</td>
<td>1.15</td>
</tr>
<tr>
<td></td>
<td>A</td>
<td>B</td>
<td>C</td>
<td>D</td>
<td>E</td>
<td>F</td>
<td>IER</td>
<td>IWF</td>
</tr>
<tr>
<td>----------</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>------</td>
<td>------</td>
</tr>
<tr>
<td>Reviewer</td>
<td>Author</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>7.36</td>
<td>7.56</td>
<td>7.63</td>
<td></td>
<td></td>
<td></td>
<td>22.55</td>
<td>1.29</td>
</tr>
<tr>
<td>B</td>
<td>4.68</td>
<td></td>
<td>6.44</td>
<td>4.35</td>
<td></td>
<td></td>
<td>15.47</td>
<td>0.89</td>
</tr>
<tr>
<td>C</td>
<td>7.02</td>
<td>7.36</td>
<td>2.76</td>
<td></td>
<td></td>
<td></td>
<td>17.14</td>
<td>0.98</td>
</tr>
<tr>
<td>D</td>
<td>4.32</td>
<td>9.81</td>
<td>4.35</td>
<td></td>
<td></td>
<td></td>
<td>18.48</td>
<td>1.06</td>
</tr>
<tr>
<td></td>
<td>E</td>
<td>5.46</td>
<td>2.76</td>
<td>0</td>
<td>8.22</td>
<td>0.47</td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>-----</td>
<td>------</td>
<td>------</td>
<td>-----</td>
<td>------</td>
<td>------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F</td>
<td></td>
<td>5.4</td>
<td>8.28</td>
<td>8.7</td>
<td>22.38</td>
<td>1.29</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total rating given to others</td>
<td>17.16</td>
<td>17.48</td>
<td>17.28</td>
<td>17.48</td>
<td>17.44</td>
<td>17.4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AER(104.24/6) =</td>
<td>17.37</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bias Factor</td>
<td>0.98</td>
<td>1.006</td>
<td>0.99</td>
<td>1.006</td>
<td>1.004</td>
<td>1.001</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Normalisation Factor</td>
<td>1.02</td>
<td>0.99</td>
<td>1.01</td>
<td>0.99</td>
<td>0.99</td>
<td>0.99</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Since all the bias factors fall in the required range, we stop our calculation here. The marks given to these students are the new normalised marks.

**Exception case**

In cases where a student does not review the assignments allocated to him, those unchecked assignments would be sent for rechecking. But this time these assignments would be forwarded only to good reviewers i.e. students with high incentives.

**2. Self Assessment:**

Self assessment is employed by the instructor when the main purpose of the course is learning. Students are asked to evaluate their own assignments on a scale of 1 to 10.
Here, in this assessment technique a student firstly completes the assignment and then he himself evaluates his assessment as a reviewer based on the evaluation sheet given by the instructor. There is no provision for incentive in this mechanism. The main purpose of this assessment technique is to motivate students to learn. It focuses on the development of student’s judgment skills. It encourages students to reflect on their work.

3. Autograde:
This will be done by special softwares for checking program codes and multiple choice questions (MCQs). Here, the system will automatically evaluate with these existing online softwares and give the result.

4. Instructor assessment:
When there are less students enrolled in a course, the instructor has the option to review the papers himself. He will score each assignment on a scale of 1 to 10.

5. Combination of above methods:
The instructor can adopt a combination of any of the above mentioned methods for evaluation for every assignment that he uploads. He has to provide the weightage for scores from each type of evaluation to be accounted in the final assignment score.

2.3 User characteristics
The instructor of the course should be a lecturer or teacher in some authentic university or school, and should have the required expertise to teach the course.

The student(author/reviewer) is anyone who wants to take the course. Since it is a massive open course, there are no prerequisites for anyone to take the course(except for a few courses that require the completion of other courses).

Both the student and instructor should have basic knowledge of computers and the web to use the software.

2.4 Constraints

2.5 Assumptions and dependencies

3. Specific Requirements

3.1 External interface requirements

3.1.1 User interfaces

3.1.2 Hardware interfaces

3.1.3 Software interfaces

3.1.4 Communications interfaces

3.2 Functional requirements

3.2.1 User class 1- Instructor
3.2.1.1 Functional requirement 1.1(REGISTRATION)
**Introduction** - The instructor shall register on the website, by providing personal information and course details. The credentials of the instructor are verified by sending an email to the institute.

**Inputs** - The instructor clicks on the “Register” button and provides following detail:

1. Name
2. Institute (Dropdown)
3. Specialisation
4. Date of Birth
5. Email
6. Gender

After all the details are filled, he presses the “Submit” button.

**Processing** - After the instructor fills in the required information, an email shall be sent to his institute to validate his credentials.

**Outputs** - After the verification, the instructor is informed via email of successful registration. A system generated password is provided, and the userid is his email address.

3.2.1.2 Functional requirement 1.2(LOGIN)

**Introduction** - The instructor is provided with a system generated password once he is registered. He can login using that for the first time, and change it later.

**Inputs** - The instructor writes his email address in the userid field and the current password in the password field. He then clicks on “Login” to continue.

**Processing** - The email id and password are matched against the database, and the instructor is logged in the system upon successful match. If the userid and password combination do not match, an error message “Invalid userid or password” is displayed and the instructor is directed to the login page again.

**Outputs** - After logging in, the instructor is directed to his Dashboard.

3.2.1.3 Functional requirement 1.3(CHANGE PASSWORD)

**Introduction** - The instructor is provided with a system generated password once he is registered. He can login using that for the first time, and change it by clicking on “Change Password” on his Dashboard.

**Inputs** - The instructor writes his old password. He then writes the new password that he wants to set.
Processing - The email id and the old password are matched against the database, and the new password is set as current password on finding a successful match. If the email id and old password do not match, an error message “Password does not match!” is displayed and the user is asked to fill the old and new password again.

Outputs - Once the new password is set, a confirmation message “Password changed successfully!” is displayed.

3.2.1.4 Functional requirement 1.4(CONTACT US)

Introduction - When the instructor clicks on “Contact Us” on his Dashboard, he is directed to a new page where he can submit his query. The query is intended for the administrator of the software.

Inputs - The instructor writes his query in the space provided and clicks on “Post Query”.

Processing - The query is emailed to the system administrator and is also posted on the website.

Outputs - The administrator will respond the query by uploading his response on the website.

3.2.1.5 Functional requirement 1.5(ADD/DELETE COURSES)

Introduction - The instructor clicks on “Add/Delete Course” on his Dashboard to add a new course, or delete an existing course. He is directed to a new page where the list of existing courses is displayed with an “Edit” button in front of each course name. An “Add” button is displayed at the end of the list to add a new course.

Inputs - When the instructor clicks on “Add” a new page is displayed where the instructor fills the following course details:

1. Title of Course
2. Duration of Course
3. Description of Course
4. Number of Assignments
5. Prerequisites of Course
If the instructor clicks on “Edit” option in front of a course name, the above details about a course will be displayed which can be edited, and a “Delete” button is provided to remove the course altogether.

**Processing** - When the instructor adds or edit a course, the course details are updated in the system database. If he deletes a course, the respective record of course is removed from the database.

**Outputs** - A confirmation message is displayed upon successful addition or deletion of course.

3.2.1.6 Functional requirement 1.6(UPLOAD ASSIGNMENT)

**Introduction** - When the instructor clicks on “Upload Assignment” on his Dashboard, he is shown the list of courses that he is teaching. The instructor selects the course and is shown a list of existing assignments of the course. The instructor clicks on “Add Assignment” and a form is displayed.

**Inputs** - The instructor fills following details about the assignment:

1. Type of Assignment (MCQ, Program Code or Subjective Type)
2. Evaluation Strategy (Self Evaluation, Peer Evaluation, Autograde, Instructor Evaluation or Combination of Above)
3. If Combination is selected, fill in the percentage weightage to each evaluation type.
4. Deadline for submitting the assignment(Calendar)
5. Number of Reviewers(if Peer Evaluation)

After providing the necessary details, the instructor browses the assignment pdf file and clicks on “Upload”.

**Processing** - The assignment is added to the course and stored in the system database.

**Outputs** - The assignment is reflected in the course details wherever the course is listed.

3.2.1.7 Functional requirement 1.7(UPLOAD MODEL SOLUTION)

**Introduction** - The instructor can upload model solution for each assignment by clicking on “Upload Model Solution” on his Dashboard.

**Inputs** - The instructor selects the course from the list of courses, and then the assignment for which the solution is to be uploaded. He browses the pdf file that is to be uploaded and clicks on “Upload”. OR he can create the assignment.
**Processing** - The model solution is added to the assignment and stored in the system database.

**Outputs** - The model solution is reflected everywhere in the assignment list of corresponding course.

3.2.1.8 Functional requirement 1.8(VIEW ENROLLED STUDENTS)

**Introduction** - The instructor can view the list of students enrolled in each course, the grades that they received and the solutions that they submitted.

**Inputs** - The instructor selects a course from the list of courses, and click on “List Students”.

**Processing** - The list of enrolled students for that corresponding course is extracted from the database and displayed with their grades. The instructor can click on each name to see the list of solutions uploaded by each student and the score and incentive awarded to him. The instructor has the authority to change the scores of students by clicking on “Edit”.

**Outputs** - After clicking on “Save”, the final grade is recalculated and displayed.

3.2.1.9 Functional requirement 1.9(RESOLVE COMPLAINTS)

**Introduction** - The instructor can reassess student's assignment and make changes in the grades if required.

**Inputs** - The instructor clicks on “View Complaints” on his Dashboard” that displays the list of student complaints.

**Processing** - The instructor reassesses student's assignment, makes necessary changes in students grades and clicks “Submit”.

**Outputs** - The grades are updated in database and modified grades are displayed on student's page.

3.2.1.10 Functional requirement 1.10(LOGOUT)

**Introduction** - The instructor can log out of the system.

**Inputs** - The instructor clicks on “Logout” on his Dashboard.

**Processing** - The system ends the instructors session.

**Outputs** - The instructor is directed to the home page.

3.2.2 User class 2- Student
3.2.2.1 Functional requirement 2.1 (REGISTRATION)

Introduction - The student shall register on the website, by providing personal information.

Inputs - The student clicks on the “Register” button and provides following detail:

1. Name
2. Institute (Dropdown)
3. Gender
4. Date of Birth
5. Email
6. Educational Qualifications

After all the details are filled, he presses the “Submit” button.

Processing - After the student fills in the required information, an email is sent to him with the userid and password.

Outputs - A system generated password is provided, and the userid is his email address.

3.2.2.2 Functional requirement 2.2 (LOGIN)

Introduction - The student is provided with a system generated password once he is registered. He can login using that for the first time, and change it later.

Inputs - The student writes his email address in the userid field and the current password in the password field. He then clicks on “Login” to continue.

Processing - The email id and password are matched against the database, and the student is logged in the system upon successful match. If the userid and password combination do not match, an error message “Invalid userid or password” is displayed and the student is directed to the login page again.

Outputs - After logging in, the student is directed to his Dashboard.

3.2.2.3 Functional requirement 2.3 (CHANGE PASSWORD)

Introduction - The student is provided with a system generated password once he is registered. He can login using that for the first time, and change it by clicking on “Change Password” on his Dashboard.

Inputs - The instructor writes his old password. He then writes the new password that he wants to set.
**Processing** - The email id and the old password are matched against the database, and the new password is set as current password on finding a successful match. If the email id and old password do not match, an error message “Password does not match!” is displayed and the user is asked to fill the old and new password again.

**Outputs** - Once the new password is set, a confirmation message “Password changed successfully!” is displayed.

3.2.2.4 Functional requirement 2.4(VIEW/ENROLL IN COURSE)

**Introduction** - The student can see the list of courses and enroll in new courses.

**Inputs** - The student clicks on “View Courses” on his Dashboard.

**Processing** - He is directed to a new page where a list of all courses is displayed. He can click on the course name to see course details. To enroll in a course, the student clicks on “Enroll” button at the end of the page.

**Outputs** - The course name is added to the database in the student’s record and he is shown a list of the assignments and corresponding deadlines of that course.

3.2.2.5 Functional requirement 2.5(SUBMIT SOLUTION)

**Introduction** - The student can submit solutions to the assignments of his course.

**Inputs** - The student selects the course from his list of enrolled courses, and then selects the assignment for which he wants to submit a solution. He browses the pdf file which he wants to upload and clicks on “Upload”.

**Processing** - The solution is stored in the student database.

**Outputs** - The file is reflected in the submitted solutions of the course.

3.2.2.6 Functional requirement 2.6(REVIEW ASSIGNMENTS)

**Introduction** - The student can evaluate/review his peers’ or his own assignments, as is designated by the instructor of the course.

**Inputs** - The student clicks on “Review Assignments” on his Dashboard.

**Processing** - A list of assignments whose deadlines have expired are listed that are to be evaluated by him. The student clicks on the assignment that he has to review and evaluates using the feedback form attached alongwith. After finishing, he clicks on the “Submit” button.

**Outputs** - The feedback form is saved and submitted, and a message “Thank you! Marks saved successfully.” is displayed.
3.2.2.7 Functional requirement 2.7(VIEW PREVIOUS ASSIGNMENTS)

**Introduction** - The student can view all his submitted assignments for a course.
**Inputs** - The student clicks on “View Previous Submissions” and a list of all his assignment solutions along with the course they belong to is displayed.
**Processing** - The student clicks on any assignment that he wants to view.
**Outputs** - The pdf file of the solution opens for him to read.

3.2.2.8 Functional requirement 2.8(UNENROLL COURSE)

**Introduction** - The students can discontinue a course whenever they want.
**Inputs** - The student selects the course from his list of enrolled courses, and clicks on “Unenroll” at the bottom of the page.
**Processing** - A confirmation message is displayed “Do you really want to discontinue the course?”. If the student selects “No”, nothing happens and he remains on the same course’s page. If he selects “Yes”, the database is updated accordingly, and all his assignments are deleted.
**Outputs** - A message “Successfully Unenrolled” is displayed on the user’s screen.

3.2.2.9 Functional requirement 2.9(REPORT WRONG GRADES)

**Introduction** - The students can complain to the instructor if he feels he has been wrongly awarded marks.
**Inputs** - The student selects the course and assignment and see his score. If he is not satisfied with the marks, he can click on “Submit Complaint”.
**Processing** - The complaint page is displayed where the student can write why he thinks he has been wrongly awarded the scores.
**Outputs** - The complaint is posted on instructor’s page. He may then review the assignment and score at his discretion.

3.2.2.10 Functional requirement 2.10(LOGOUT)

**Introduction** - The student can log out of the system.
**Inputs** - The student clicks on “Logout” on his Dashboard.
**Processing** - The system ends the student’s session.
**Outputs** - The student is directed to the home page.
3.2.3.1 Use case diagram

[Diagram of edX system with use cases and interconnections]
<table>
<thead>
<tr>
<th>Use Case Name</th>
<th>Registration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actors</td>
<td>Anonymous users</td>
</tr>
<tr>
<td>Description</td>
<td>Creating an account on the website</td>
</tr>
<tr>
<td>Course of Event (Flow)</td>
<td></td>
</tr>
<tr>
<td>Actor's action</td>
<td>System's action</td>
</tr>
<tr>
<td>1. Clicks 'register'.</td>
<td>2. Registration form is displayed.</td>
</tr>
<tr>
<td>3. Fills in details.</td>
<td>5. Validates information. In case of instructor, sends a mail to the institute for validation.</td>
</tr>
<tr>
<td>4. Clicks submit.</td>
<td>6. Sends mail to the user after validation of information.</td>
</tr>
<tr>
<td>Precondition</td>
<td>Should not be an existing user.</td>
</tr>
<tr>
<td>Postcondition</td>
<td>User is registered.</td>
</tr>
<tr>
<td>Alternate flow</td>
<td></td>
</tr>
<tr>
<td>Diagram Representation</td>
<td></td>
</tr>
<tr>
<td>Use Case Name</td>
<td>Request new password</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------------------------------------------------------</td>
</tr>
<tr>
<td>Actors</td>
<td>Registered instructor, registered students</td>
</tr>
<tr>
<td>Description</td>
<td>In case instructor or student forgets the password, they can reset the password to access the account.</td>
</tr>
<tr>
<td>Course of Event</td>
<td>(Flow)</td>
</tr>
<tr>
<td></td>
<td>Actor's action</td>
</tr>
<tr>
<td>1. Clicks 'change password'.</td>
<td>2. displays page.</td>
</tr>
<tr>
<td>3. Clicks submit.</td>
<td>4. Sends mail to user with a link.</td>
</tr>
<tr>
<td>5. Clicks link in the mail received.</td>
<td>6. Redirected to new page.</td>
</tr>
<tr>
<td>8. Enters new password.</td>
<td>7. Displays a page with details.</td>
</tr>
</tbody>
</table>
### Precondition
Should be a registered student/instructor.

### Postcondition
Password is changed successfully.

### Alternate flow

### Diagram Representation

### Use Case Name
Contact us

### Actors
Anonymous users, registered student/instructor.

### Description
Used to contact the administrator of site in case of any query or difficulty.

### Course of Event (Flow)

<table>
<thead>
<tr>
<th>Actor's action</th>
<th>System's action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Clicks 'contact us'.</td>
<td>2. Displays contact us page.</td>
</tr>
<tr>
<td>3. Enters details/queries.</td>
<td>5. Query/message posted on site</td>
</tr>
<tr>
<td></td>
<td>7. admin will post the answer on the site.</td>
</tr>
</tbody>
</table>

### Precondition
None.

### Postcondition
Answer to the query posted on site.

### Alternate flow
## Use Case Name
Login/logout

## Actors
Registered instructor/student.

## Description
Login and logout to perform some activity on the site.

## Course of Event (Flow)

<table>
<thead>
<tr>
<th>Actor's action</th>
<th>System's action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Enters user name or password. 2. clicks “login”.</td>
<td>3. Logs into the site.</td>
</tr>
</tbody>
</table>

## Precondition
Should be a registered user on site.

## Postcondition
User logged in/logged out.

## Alternate flow
Incorrect username or password.

<table>
<thead>
<tr>
<th>Actor's action</th>
<th>System's action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Enters</td>
<td></td>
</tr>
<tr>
<td>Incorrect user name or password.</td>
<td>2. clicks login. 3. displays error message. 4. user redirected to login page.</td>
</tr>
</tbody>
</table>

**Diagram Representation**

<table>
<thead>
<tr>
<th>Use Case Name</th>
<th>Add/Delete courses</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actors</td>
<td>Registered instructor</td>
</tr>
<tr>
<td>Description</td>
<td>Instructor adds a new course or deletes an existing course.</td>
</tr>
<tr>
<td>Course of Event (Flow)</td>
<td>Actor's action</td>
</tr>
<tr>
<td>------------------------</td>
<td>----------------</td>
</tr>
<tr>
<td>1. Clicks 'add course'.</td>
<td>2. A form is displayed with all the course details.</td>
</tr>
<tr>
<td>3. Instructor fills the form.</td>
<td>5. Course successfully added message is displayed and newly added course is displayed in the instructor's course list.</td>
</tr>
<tr>
<td>4. Clicks submit.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Precondition</th>
<th>Course should not already exist in the instructor's course list.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Postcondition</th>
<th>New course is added.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Alternate flow</th>
<th>Diagram Representation</th>
</tr>
</thead>
</table>

![Diagram](image)
<table>
<thead>
<tr>
<th>Use Case Name</th>
<th>Upload assignment.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actors</td>
<td>Registered instructor.</td>
</tr>
<tr>
<td>Description</td>
<td>Instructor uploads assignment to any of his courses.</td>
</tr>
<tr>
<td>Course of Event (Flow)</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Actor's action</th>
<th>System's action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Clicks on upload assignment.</td>
<td>2. Instructor's course list is displayed.</td>
</tr>
<tr>
<td>3. Instructor selects a course.</td>
<td>4. Selected course details along with a form for details of assignment to be uploaded is displayed.</td>
</tr>
<tr>
<td>5. Fills the form.</td>
<td>6. Assignment is added to the course.</td>
</tr>
</tbody>
</table>
6. Browses for the file and clicks on upload.

Precondition
Instructor should be logged in.

Postcondition
Assignment is added to the course.

Alternate flow
Assignment is added to the course.

Diagram Representation

Use Case Name
Upload model solution.

Actors
Registered instructor.

Description
Instructor would be able to upload a model solution to the assignment if he wants.

Course of Event
(Flow)

<table>
<thead>
<tr>
<th>Actor's action</th>
<th>System's action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Selects course and assignment for which he wants to upload solution. 2. Browses for the file and clicks upload.</td>
<td>3. 'uploaded successfully' message is displayed.</td>
</tr>
<tr>
<td>Precondition</td>
<td>Should upload after deadline.</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------------------------</td>
</tr>
<tr>
<td>Postcondition</td>
<td>Model solution uploaded and available to students.</td>
</tr>
<tr>
<td>Alternate flow</td>
<td></td>
</tr>
<tr>
<td>Diagram Representation</td>
<td><img src="image" alt="Diagram" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Use Case Name</th>
<th>View enrolled students</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actors</td>
<td>Registered instructor.</td>
</tr>
<tr>
<td>Description</td>
<td>Instructor would be able to view list of students enrolled in a course along with their grades.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Course of Event (Flow)</th>
<th>Actor's action</th>
<th>System's action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Selects a course and clicks on view enrolled students.</td>
<td>2. A list of all students enrolled in the selected course is displayed along with their grades/marks.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Precondition</th>
<th>Instructor should be logged in.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Postcondition</td>
<td>None.</td>
</tr>
<tr>
<td>Alternate flow</td>
<td></td>
</tr>
<tr>
<td>Diagram Representation</td>
<td></td>
</tr>
<tr>
<td>Use Case Name</td>
<td>View+enroll in courses.</td>
</tr>
<tr>
<td>---------------</td>
<td>-------------------------</td>
</tr>
<tr>
<td>Actors</td>
<td>Registered student.</td>
</tr>
<tr>
<td>Description</td>
<td>Student would be able to see the list of courses and enroll in new courses.</td>
</tr>
<tr>
<td>Course of Event (Flow)</td>
<td>Actor's action</td>
</tr>
<tr>
<td>-----------------------</td>
<td>----------------</td>
</tr>
</tbody>
</table>
| 1. Clicks view courses.  
3. Selects a course and clicks enroll. | 2. A list of all courses is displayed.  
4. Student is enrolled and a page is displayed with all course details and assignments. | |

<table>
<thead>
<tr>
<th>Precondition</th>
<th>Student should be logged in.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Postcondition</td>
<td>Student enrolled in new course.</td>
</tr>
</tbody>
</table>

| Alternate flow | 

<table>
<thead>
<tr>
<th>Diagram Representation</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Diagram" /></td>
</tr>
<tr>
<td>Use Case Name</td>
</tr>
<tr>
<td>-----------------------</td>
</tr>
<tr>
<td>Actors</td>
</tr>
<tr>
<td>Description</td>
</tr>
</tbody>
</table>

**Course of Event (Flow)**

<table>
<thead>
<tr>
<th>Actor's action</th>
<th>System's action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Student selects the course and the assignment.</td>
<td>3. Uploads the assignment.</td>
</tr>
<tr>
<td>2. Browses for the file and then clicks upload.</td>
<td>4. 'Assignment uploaded successfully' message is displayed.</td>
</tr>
</tbody>
</table>

**Precondition**

Deadline should not have passed.

**Postcondition**

Assignment uploaded.

**Alternate flow**

**Diagram Representation**

![Diagram](image)

---

<table>
<thead>
<tr>
<th>Use Case Name</th>
<th>Review assignments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actors</td>
<td>Registered student.</td>
</tr>
<tr>
<td>Description</td>
<td>Student can evaluate/review peer's/self assignment.</td>
</tr>
</tbody>
</table>

**Course of Event**
<table>
<thead>
<tr>
<th>(Flow)</th>
<th>Actor's action</th>
<th>System's action</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1. Clicks on 'review assignment'.</td>
<td>2. A list of students is displayed whom he has to evaluate.</td>
</tr>
<tr>
<td></td>
<td>3. Selects a student.</td>
<td>4. Selected student's assignment is displayed along with evaluation sheet (feedback).</td>
</tr>
<tr>
<td></td>
<td>5. Evaluates assignment and fills the feedback.</td>
<td>7. 'Marks updated successfully' message displayed</td>
</tr>
<tr>
<td></td>
<td>6. Clicks submit.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Precondition</th>
<th>Should be logged in and should have submitted assignment. Deadline should have passed.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Postcondition</th>
<th>Marks updated in database.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Alternate flow</th>
<th>Use Case Name</th>
<th>View previous assignments+solutions.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Actors</td>
<td>Registered student.</td>
</tr>
<tr>
<td></td>
<td>Description</td>
<td>Student can view his previous submissions.</td>
</tr>
<tr>
<td>Course of Event (Flow)</td>
<td>Actor's action</td>
<td>System's action</td>
</tr>
<tr>
<td>------------------------</td>
<td>----------------</td>
<td>----------------</td>
</tr>
<tr>
<td></td>
<td>1. Clicks 'view previous submissions'.</td>
<td>2. A list of previous assignments is displayed.</td>
</tr>
<tr>
<td></td>
<td>3. Selects an assignment</td>
<td>4. The selected assignment's solution is displayed.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Precondition</th>
<th>Student should be logged in.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Postcondition</td>
<td>None.</td>
</tr>
</tbody>
</table>

Alternate flow

Diagram Representation

![Diagram](image)

<table>
<thead>
<tr>
<th>Use Case Name</th>
<th>Unenroll courses.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actors</td>
<td>Registered student.</td>
</tr>
<tr>
<td>Description</td>
<td>Student would be able to discontinue a course.</td>
</tr>
<tr>
<td>Course of Event (Flow)</td>
<td>Actor's action</td>
</tr>
<tr>
<td>------------------------</td>
<td>----------------</td>
</tr>
<tr>
<td></td>
<td>1. Selects a course from his/her course list and clicks “unenroll”.</td>
</tr>
<tr>
<td></td>
<td>3. Clicks yes.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Precondition</th>
<th>Student should be registered in the selected course.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Postcondition</td>
<td>Course removed from student's course list. Student's name removed from list of enrolled students in that course.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Alternate flow</th>
<th></th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Diagram Representation</th>
<th></th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Use Case Name</th>
<th>Report wrong grades.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actors</td>
<td>Registered students.</td>
</tr>
<tr>
<td>Description</td>
<td>Student can complain about his grades to the instructor if he feels there is a mistake.</td>
</tr>
<tr>
<td>-----------------------------------------------------------------------------</td>
<td>--------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>

**Course of Event (Flow)**

<table>
<thead>
<tr>
<th>Actor's action</th>
<th>System's action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Selects the course and assignment. 2. Clicks “submit complaint”. 4. Student fills the form.</td>
<td>3. complaint page displayed. 5. complaint posted on instructor's page.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Precondition</th>
<th>Student has been awarded grades for the selected assignment.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Postcondition</td>
<td>Complaint displayed on instructor's page.</td>
</tr>
<tr>
<td>Alternate flow</td>
<td></td>
</tr>
</tbody>
</table>

**Diagram Representation**

![Diagram](image)

<table>
<thead>
<tr>
<th>Use Case Name</th>
<th>Resolve complaints regarding grades.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actors</td>
<td>Registered instructor.</td>
</tr>
<tr>
<td>Description</td>
<td>Instructor can reassess student's assignment and make changes in the grades if required.</td>
</tr>
<tr>
<td>-------------</td>
<td>---------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Course of Event (Flow)</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Actor's action</strong></td>
<td>System's action</td>
</tr>
<tr>
<td>1. Clicks on view complaints. 2. Selects a complaint.</td>
<td>3. Displays the selected student's complaints.</td>
</tr>
<tr>
<td>4. Reassess student's assignment. 5. Makes changes in students grades if required and clicks submit.</td>
<td>6. Grades are updated in database and modified grades are displayed on student's page.</td>
</tr>
<tr>
<td><strong>Precondition</strong></td>
<td>Instructor should be logged in.</td>
</tr>
<tr>
<td><strong>Postcondition</strong></td>
<td>Grades modified.</td>
</tr>
<tr>
<td><strong>Alternate flow</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Diagram Representation</strong></td>
<td><img src="image" alt="Diagram" /></td>
</tr>
</tbody>
</table>
3.2.3.2 Activity Diagrams
EVALUATION SYSTEM
CALIBRATION

Random division of all students into groups → if no calibration

if expert calibration → all students rate an assignment which is expert rated
if expert rating = x
Student rating = y

if peer calibration → all students rate an assignment, then rating calculated for that assignment = x

Each student rating = x
Incentive = 1 (or CM) x maximum marks

Students divided into groups based on their incentive
No. of groups = no. of reviewers
3.2.3.3 Sequence Diagrams

USER LOGIN SEQUENCE DIAGRAM

User

1. Prompt for Login

Enter Username

Enter Password

Validated

Interface

Forward

Validated

Database

Success

Display Homepage

On Success

Success

Failure

Display Error Page

On Failure

Enter username and password
USER LOGIN SEQUENCE DIAGRAM

User

1. Prompt for Login

Enter Username

Enter Password

Validate

Success

Database

Forward

Validate

Success

Display HomePage

On Success

Failure

Display ErrorPage

On Failure

Enter username and password
REVIEW ASSIGNMENT

User

Prompt for Review Assignments
Click on the tab
Forward
Select a student at a time
Forward
Forward
Forward
Displays list of students whose papers are to be evaluated
Forward
Displays the student's assignment and feedback form
Evaluate assignment and return feedback form
Validate
Forward
On Success
Success
Marks updated successfully in student profile

Database
validate
3.2.3.4 Dataflow diagram
3.3 Performance requirements
3.4 Design constraints
3.5 Software system attributes
3.6 Other requirements