

Project Requirements and Analysis Artifacts

Gurnoor Singh Khurana	190050045
Prabhanshu Katiyar	190050088
Sahasra Ranjan	190050102
Sibasis Nayak	190050115

Problem Description

The application consists of a question bank that can allow teachers to add questions, tag them with topics, sub topics, add levels of difficulty, specify appropriateness for specific kinds of exams. The application will also allow the teachers to get a detailed analysis of student data, which we describe in detail soon.

On the student's side, they can rate the questions based on difficulty level according to them, search for questions involving specific topics for practice, view statistics which show how many students **attempted the question** and how many could **correctly solve it**, average time taken to solve.

On the admin panel (teacher's side), there will be two options: a list of students with a separate page showing data per student and a summary of statistics. Details of what these pages will contain is given below.

A database is more suited for this task rather than a set of files because of the ease and efficiency of querying, inserting and updating that it provides. Authorization can also be handled easily in databases by assigning appropriate permissions to the users. Also, the atomicity guarantees that the database provides will be helpful for large-scale applications, so using a database is a more scalable option.

We plan to use relational database for the most part, and GraphDB for representing the hierarchy between various topics and sub topics. Relational database is a simple yet powerful tool for most of our application. The topic hierarchy, though can be represented using relational database, will be much more cleaner and efficient if done via graph database because of the inherent graph structure in it.

Per student page

1. Pie chart showing number of questions attempted and number of questions solved correctly
2. Average time per question
3. List of top 5 popular topics
4. Average difficulty level of questions

5. Bar graph showing number of questions attempted in the past week, month or year

Summary of statistics on teacher's side

1. Leaderboard containing students ranked by the difficulty level of the questions they solve
2. Total number of questions solved by students
3. Top 5 most popular questions
4. Top 5 most popular topics
5. Question distribution provided by the teacher: a sunburst diagram showing topics and subtopics by the teacher



A sunburst diagram [[Image source](#)]

In addition to this, we will provide a page containing a list of all questions. Each question has a hyperlink to a per question page which will contain:

Per Question Page

1. Number of students that attempted the question, and the number who could solve it
2. Average time taken to solve
3. Average difficulty assigned by students and difficulty assigned by teacher

The application has two interfaces.

Teacher: Can view/add/edit/remove questions from the database. Teacher can view past questions papers and statistics of individual questions. Teacher can filter questions based on tags and difficulties. Teacher can create exams either manually or by using a generator by passing specific tags and difficulties. For any question, teacher can view the exams in which this question appeared and what fraction of students were able to solve it.

Teacher can also view the report card of his student (advisee) which consists of his/her performance in all the exams he/she appeared for. Teacher can enroll students for exam

Student:

Student can access the coding environment, attempt the exam, rate the questions after the exam. Student can also view his report card generated by the application.

For dataset we plan to use the [cses](#) problemset, which consist of 300 questions with test cases and answers to the question

Use Cases

TEACHER

- Adding or removing questions from database
 - Description - Teacher can formulate questions and then add or remove questions that he/she had earlier added from the database
 - Trigger - This event will be performed when the teacher triggers this
 - Primary Actor - Teacher
 - Inputs - Question text, question tags (should be chosen from the specified set), question difficulty as rated by teacher (scale of 1 to 5), correct answer and test cases
 - Preconditions - This use case is relevant when the registered teacher accesses the running site
 - The database is modified to add a new entry and cascading changes on the other pages like list of questions/ other stats
 - Exceptions - Can check for possible mentions of tags that should not get overlapped
 - Postcondition - A new question will get added to the database
- View past questions papers and statistics of individual questions
 - Description - Teacher can view the list of questions that are present in the database and check the detailed statistics of each question like(average time taken, %percentage of correct answers, no of exams and list of exams it had appeared in)
 - Trigger - When teacher navigates to this page
 - Primary Actor - Teacher
 - Inputs - NA
 - Preconditions - This use case is relevant when the registered teacher accesses the running site and the database has questions
 - Database is accessed by querying the database and then the results are displayed on the site
 - Exceptions - If there are no questions yet, then display a “No question found” exception
 - Postcondition - Questions with details will be fetched to the page
- Filter questions based on tags and difficulties
 - Description - Teacher while choosing questions for exams can filter the questions from the list of total questions according to tags and difficulties
 - Trigger - We will have a page for this with relevant filters to choose, like tags or difficulties, and teacher will have to trigger by going to this page

- Primary Actor - Teacher
 - Inputs - Question tags, Primary difficulty level, Secondary difficulty level
 - Preconditions - This use case is relevant when there some questions in database
 - Database is accessed by querying the database with the relevant filters and then the results are displayed on the site
 - Exceptions - If there are no questions matching the filter then we can return an exception
 - Postcondition - Filtered out questions with details will be fetched to the page
- Create exams either manually or by using a generator by passing specific tags and difficulties
 - Description - Teacher can create an exam from the list of questions in two ways. Either by choosing each question manually for his exam or by passing specific tags and difficulties
 - Trigger - This event will be performed when the teacher triggers this
 - Primary Actor - Teacher
 - Inputs - For manual creation there are no inputs. For filtered paper inputs are Question tags, Primary difficulty level, Secondary difficulty level. For exam details we will have start time and end time of the exam.
 - Preconditions - This use case is relevant when there some questions in database
 - For manual we just have to provide the UI. For filtered Database is accessed by querying the database with the relevant filters and then from the results questions are randomly selected. The exam formulated is added to the database
 - Exceptions - If there are not enough questions we can return an exception
 - Postcondition - An exam will get added to the database and that will get displayed on the list of exams page
- Teacher can enroll students for exam
 - Description - Teacher can enroll students who wanted to give the exams
 - Trigger - This event will be performed when the teacher triggers this, to enroll students from the list of students
 - Primary Actor - Teacher
 - Inputs - List of student ids, and the corresponding exam id
 - Preconditions - This use case is relevant when the exam has already been created and students already exist
 - The database has to be modified to add the new students
 - Exceptions - Exams and students should exist otherwise return trigger
 - Postcondition - New exams and students have been created
- View the report card of his student (advisee) which consists of his/her performance in all the exams he/she appeared for
 - Description - Teacher can view the performance of students in the exams he/she had organized

- Trigger - This event will be performed when the teacher triggers this, to view the performance
- Primary Actor - Teacher
- Inputs - Exam_id
- Preconditions - This use case is relevant when the exam has already been created and it has been completed
- The database has to be modified to add the new students
- Exceptions - Exams and students should exist otherwise return exception
- Postcondition - New exam and students list have been created

STUDENT

- View past exam papers and his performance
 - Description - Student can view the list of questions that he appeared for and his performance in the paper.
 - Trigger - When student navigates to this page
 - Primary Actor - Student
 - Inputs - NA
 - Preconditions - This use case is relevant when the registered student accesses the running site and the database has questions and exam details
 - Database is accessed by querying the database and then the results are displayed on the site
 - Exceptions - If there are no such exam, then display a "Invalid exam request" exception
 - Postcondition - Exam and questions with details will be fetched to the page
- View his report card which consists of his/her performance in all the exams he/she appeared for
 - Description - Student can view his/her performance in the exams he/she had organized
 - Trigger - This event will be performed when the student triggers this, to view his/her performance
 - Primary Actor - Student
 - Inputs - Exam_id
 - Preconditions - This use case is relevant when the exam has already been created and it has been completed
 - The database is not modified
 - Exceptions - Exams and students should exist otherwise return exception
 - Postcondition - No postcond.
- Appear for an exam
 - Description - Student can appear for an exam in the exam hour slot and attempt the problems
 - Trigger - This event will be performed when the student triggers this in the given time duration

- Primary Actor - Student
- Inputs - Exam_id
- Preconditions - This use case is relevant when the exam has already been created and student want to appear for the exam
- The database tracks that the student has appeared and adds the relevant entries
- Exceptions - Exams and students should exist otherwise return exception
- Postcondition - No postcond.
- Submitting an exam
 - Description - Student can submit the exam if appeared and finished solving the problems or if the time limit is over
 - Trigger - This event will be performed when the student triggers this in the given time duration or time limit of test is over
 - Primary Actor - Student
 - Inputs - Exam_id
 - Preconditions - This use case is relevant when the exam has already been created and student appeared for the exam
 - The database tracks that the student has appeared and adds the relevant entries related to student's performance
 - Exceptions - Exams and students should exist otherwise return exception
 - Postcondition - Return student's performance in the exam along with aggregates performance of all the students who appeared for the exam

Entities

Question

- question_id
- text
- tags {}
- Primary_difficulty (as provided by creator)
- answer
- testcase
- visibility

Student

- student_id
- First Name
- Last name
- Date of Birth
- user_name
- password

Teacher

- teacher_id
- First name
- Last name
- Date of Birth
- user_name
- password

Institute

- institute_id
- name
- location

Exam

- exam_id
- exam_name
- year
- creator

Tags

- tag_id
- Tagname
- Child_nodes {}

examType

- id
- type (long/short/graded)
- average difficulty

Template

- template_id
- font

Language

- id
- name

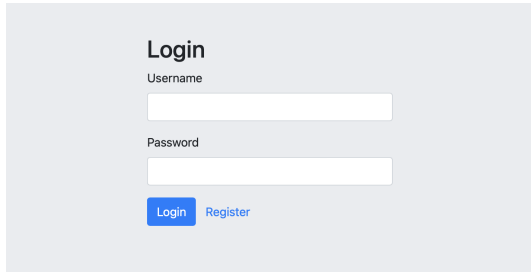
student_exam_question_stats (relation)

- rating
- time
- marks
- relevance

Forms

For each form the output will be a message indicating that the data has been submitted successfully.

Login



A login form with a light gray background. It features a title "Login" in bold. Below the title are two input fields: "Username" and "Password". At the bottom, there are two buttons: "Login" (blue) and "Register" (blue).

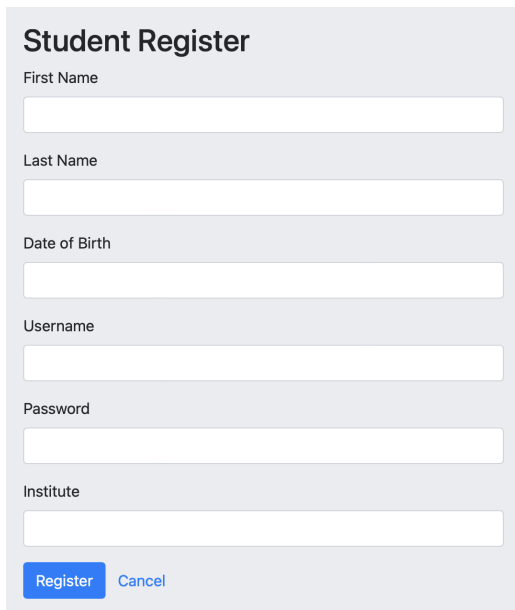
Login

Username

Password

Login Register

Student Register



A student registration form with a light gray background. It features a title "Student Register" in bold. Below the title are six input fields: "First Name", "Last Name", "Date of Birth", "Username", "Password", and "Institute". At the bottom, there are two buttons: "Register" (blue) and "Cancel" (blue).

Student Register

First Name

Last Name

Date of Birth

Username

Password

Institute

Register Cancel

Teacher Register

Teacher Register

First Name

Last Name

Date of Birth

Username

Password

Add Question

Add question

Question

Tags

Difficulty

Answer

Testcase

Add Template

Template details:

Template Name *

text *

Complete the form to enable button.

Add Template

Templates:

basic

test

akjnsfjka

Show Template

Add Institute

Add Institute

Name

Location

Add

Combine questions to generate exam



Show per page: ▼



<input type="checkbox"/> 1	#Tags	Detailed View ▼
<input type="checkbox"/> 2	#Tags	Detailed View ▼
<input type="checkbox"/> 3	#Tags	Detailed View ▼
<input type="checkbox"/> 4	#Tags	Detailed View ▼

Template
basic ▼

Generate exam

