ISSN No:-2456-2165

ExpertQGen : an Expert Automated Question Paper Generation System

Karthikayani K
Department of Computer Science and Engineering
SRM Institute of Science & Technology
Chennai, India

Mohit Tulsani
Department of Computer Science and Engineering
SRM Institute of Science & Technology
Chennai, India

Vaishnavi Raghavan
Department of Computer Science and Engineering
SRM Institute of Science & Technology
Chennai, India

Abstract:- In any educational course curriculum, the courses are defined with learning objectives. Faculty assesses the students through various tests to determine whether certain learning objectives have been achieved by them. Hence, generation of appropriate question papers plays a major role in this process. The need of the hour is to have a system that will automatically generate the question paper from teacher entered specifications within a few seconds. The question paper should satisfy various design constraints like coverage of syllabus and difficulty levels while adhering to the specific question paper template of a particular university. The traditional method where teachers manually prepare the question paper is tedious, challenging and time-consuming. The proposed system allows the teachers to create a question bank with varying difficulty levels for every chapter in a subject and generate a randomized question paper of the desired difficulty level on the click of a button. The system makes use of the Greedy algorithm and Randomized shuffling algorithm to generate the question paper in Microsoft Word Document format (.docx).

Keywords:- *Question Paper Generator, SQL, examination, question aggregator, education, question repository.*

I. INTRODUCTION

In the present competitive world, an examination plays a very crucial role in checking the intellectual growth of students. Thus, an examination is considered as the most important measure of competence in the current education system of our country and plays a decisive role in career building of the students. For any academic course, therefore, multiple examinations are conducted throughout the year in any university. This requires the teachers to create various question papers in order to meet the university's assessment norms. The university guidelines tend to focus more on the format of the question paper rather than its quality. It is a very challenging and time-consuming task for the teachers to cover all the aspects of the course objectives, avoid duplication of questions in subsequent exams while maintaining an adequate level of difficulty of the question paper. These hardships could lead to the deterioration of the quality of the question papers at times.

Question papers are often under debate for their levels of difficulty and lengthy content. Question paper setting process and evaluation has become very important especially for students hailing from different educational backgrounds and boards. Strict evaluation techniques are used by the paper setters who prefer difficult questions and as a result, make the process more stringent. The purpose of this process is to identify knowledgeable candidates and restrict copying, but they don't realise that implementing such difficult aspects may have a negative effect on students. There have been certain instances when due to public outcry, the governing bodies of these examinations have created committees to suggest means and modes for providing relief to affected students. Therefore, it is better to take precautions in order to avoid discord after examinations.

With the profound dispersal of technology in the area of education, acquiring technology to smooth the technique of examination paper creation and the creation of extensively vast question bank aiding the automatic question paper generation furnishes a key provision to the issue encountered during the manual composition of examination papers. This yields a stage to create a well-organized question paper and the automation would incorporate all the elements determining the quality of a question paper like the required paper format, desired difficulty level, and chapter weightage.

The structure of the system is general and is not for any specific branch of learning. This generic system can be utilized by all the departments of the university and can be easily redesigned if the need arises. Before the generation of any question paper, the teacher must compose the question bank according to the modules covered for an individual subject. During this process, the teacher can assign a difficulty label for every question that is uploaded. The system algorithmically creates a question paper from this semantically labeled question bank by choosing the questions of the most optimal difficulty level and presents the question paper in a Microsoft Word Document format for the ease of use. Hence, the proposed system fulfils the need for an unbiased selection of questions in a question paper as well as reduces the manpower and time required for the same.

Section II discusses the related literature survey. The design approach is elaborated in section III. The implementation aspects and the UI is described in section IV. Conclusion and Future scope of our work is available in section V which is followed by the references that have been used for our work.

II. LITERATURE SURVEY

A literature survey was initiated to understand the limitations of the existing paper-based system and the need for

ISSN No:-2456-2165

an automatic generation of question paper. Some of the common concerns about the existing manual paper-based system as expressed by the participants of the survey are listed as follows:

- Lack of storage space
- Prone to damage
- Inefficient document transportation
- Editing problems
- Limited collaboration
- Time-consuming
- Bias

In 2006, Ittizar Aldabe et al [1] attempted to generate automatic questions called Arik Iturri. This procedure was based on Corpora and NLP methods, and the source of information for the proposed system was linguistically inspected real corpora and was depicted in XML markup language. The advantage of deploying this kind of system is that it excludes the possibility of ill-developed questions.

In 2006, Li-Chun Sung et al [2] presented an outline for Automatic Quiz Generation for worldwide English E-learning System from a given English text to evaluate learner. This was built on quiz creation system for the understanding of text content. The question generator takes the pure semantic network and the information base of Word Net and Google to create questions for the quiz.

In 2010, Ming Liu et al [3] presented automatic question generation for literature review writing support. He took the literature review as input word and then the selected syntactic and semantic characteristic of words were used for question generation, which was based on a specification question bank consisting of equal match pattern and question templates. However, the questions were not simple word-based questions; they were based on groups like opinion, result, system, which evaluated the intellectual learning experience of the learner. Another close work done by them was "G-ASKS", where the similar method was used for academic writing to create questions built on Graesser & Person taxonomy, here the Citation Classification was finished with the aid of Naive Bayes Classifier.

III. DESIGN

We have implemented a system that facilitates automatic generation of question paper from a question repository created by the faculty member. universities and test paper setters who have a huge repository of questions for the various subjects and need to frequently generate question paper for different types of tests with ease. In this paper, the working of the system is demonstrated for written examination of Web Mining domain of an engineering curriculum.

A. System Architecture

We are providing three modules in our proposed system. The first module enables the user to input questions into our system and creates a question repository. The user will be prompted to enter the difficulty level for each question for a particular chapter in a specific course. The second module enables the user to view, update and delete the questions that are already existing in the repository. The third module generates a question paper based on the user's specification by executing the greedy and randomization algorithm. It applies the greedy based algorithm to fetch the questions from the question repository in order to give the best optimum solution and then the randomized algorithm is applied to shuffle these questions before creating the question paper. The final output is produced as a word document format as per the university's paper pattern.

The system architecture diagram is shown in figure 1.

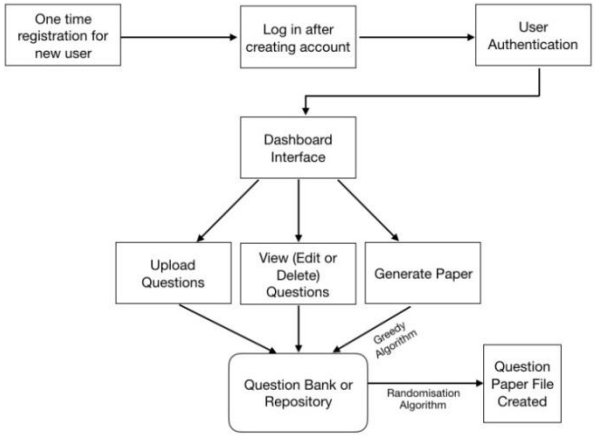

Fig 1:- System Architecture Diagram

B. Input Database

The system makes use of SQL Database which has separate tables for storing the users and the question repository. The question repository is split according to the type of questions (Multiple Choice Questions, Subjective Questions, and Long Essay Type Questions). The multiple choice questions have fields like question, option 1, option 2, option 3, option 4 and the difficulty level tag. The subjective and essay type question tables have a provision to store images as a blob (Binary Large Object) in case of any diagram based questions. The other fields include question id, username, subject code, subject name, unit, question and the difficulty level tag (low, moderate or high). Apart from these the common fields in all the database tables for the question repository are namely, username, subject code, subject name, and unit (chapter) number.

The structure for question repository of Part B is depicted below:

#	Field	Schema	Table	Type	Character Set
1	uname	qpg	questionbankb	VARCHAR	utf8
2	subject_code	qpg	questionbankb	VARCHAR	utf8
3	subject_name	qpg	questionbankb	VARCHAR	utf8
4	unit	qpg	questionbankb	VARCHAR	utf8
5	question_id	qpg	questionbankb	INT	binary
6	question	qpg	questionbankb	VARCHAR	utf8
7	image	qpg	questionbankb	BLOB	binary
8	difficulty	qpg	questionbankb	VARCHAR	utf8

Fig 2:- Database Structure

C. Paper Specifications

The subject handler will be asked to input two different specifications in order to generate the question paper namely (i) Test details (ii) Paper pattern details.

ISSN No:-2456-2165

- *Test Details*

In the first stage of question paper generation process, the subject handler will be asked to input various test details that constitute the header of the question paper. These test details include the subject code, test-type, month, year, semester, department and the desired difficulty level of the paper. Apart from these details, the header also includes the university name and an input field to enter the registration number of the candidate. The subject name for the header is fetched from the database depending upon the subject code that is specified.

- *Paper pattern details*

The next interface will prompt the user to enter the question paper pattern details. It allows the user to specify the number of questions for each unit based on the question type (Multiple choice, subjective and essay). Once these details have been submitted, they aid the algorithm in the generation of a complete question paper of the desired difficulty level.

D. Generation Process

Once the specification details are submitted the following two-part process is carried out in order to select the suitable questions for the paper.

- *Part One - Greedy Algorithm for fetching questions*

The greedy algorithm paradigm is used to make choices that seem to be the most optimum at that moment hoping that this choice will lead to a globally optimum solution. This algorithm makes greedy choices at each step in fetching the questions of the desired difficulty level. It tries to ensure that it doesn't have to go back and reverse the decision that has been made.

The algorithm can be explained by the following example. Assume that there is exactly "T" time to accomplish a set of tasks and the aim is to complete a maximum number of such tasks. The input is given as an array "A" of integers, where each array element indicates the time a particular task takes to complete and we have to calculate the maximum number of things that can be done in the limited time that is available.

The execution can be explained as follows. In each iteration, you have to greedily select the tasks that will require the minimum amount of time to complete while maintaining two variables "current Time" and "number of Tasks". In order to complete the calculation process, we must perform the following steps:

1 Sort the array A in an increasing order.
2 Select each and every task one at a time.
3 Add the time that will be taken to complete that particular task in "current Time" array.
4 Increment the "number of Tasks" by one.

This process should be repeated as long as the value of "current Time" is less than or equal to "T".

Thus, we modified the algorithm in a way that it handles the overall difficulty level of the paper depending upon the user's desired difficulty. For instance, if the user wishes to create a question paper of "hard" difficulty level, the algorithm checks if there are sufficient or more than sufficient questions for that particular difficulty and fetches them. If not, it changes its preference to the next difficulty level (in this case, moderate). If the minimum requirement is still not met, the algorithm fetches the questions of the least preferred difficulty level (in this case, low). Similarly, if the user's desired difficulty level is "moderate", the order of precedence that is followed by the algorithm is moderate —> low —> hard. If the desired difficulty level is "low", then the order of precedence that is followed by the algorithm is low —> moderate —> hard. This process is repeated for every unit or chapter in the course syllabus. Thus, once all the questions have been fetched they are sent to the randomisation algorithm for final random selection.

- *Part two - Randomisation algorithm for final selection*

The randomisation algorithm is generally used to make random decisions rather than deterministic decisions. This algorithm runs differently each time and hence, cannot produce worst-case results for any given input. This algorithm tries to achieve a good performance in average-case situations over all possible choices of random numbers. It handles the extent of randomness as a part of its logic.

A way of implementing the randomised algorithm is to assign the index values at their respective positions and then shuffle the array using random indexes. Once it has been shuffled then the elements can be accessed. This results in the generation of new combinations every time.

The array list of the questions retrieved from the previous step is processed as per the desired difficulty level and randomly shuffled to select the final list of questions based on the requirement from each unit. For instance, if the desired difficulty is "hard" and the requirement from a particular unit is "four", then the algorithm checks if there are sufficient or more than sufficient questions of that level. If true, then it randomly shuffles the questions and selects any "four" from the list. However, if there are less number of questions, then it selects all the "hard" level questions first and then shuffles the questions of the next preferred difficulty level (in this case, moderate). If the requirement is still not met, the algorithm selects the "hard" level and the "moderate" level questions first and then shuffles and randomly selects questions from the next preferred difficulty level (in this case, low). A similar process is carried out for other difficulty levels. This process is applied to questions from each and every available unit in the array list and thus, the final set of questions are derived and sent to the file creation process.

- *Question Paper File Generation*

The information retrieved from paper specifications and generation process is arranged in an orderly fashion and integrated to create a Microsoft Word Document (.docx format) which serves as the final question paper.

IV. IMPLEMENTATION

The proposed system has been developed as a web-based application using Eclipse IDE. The front-end technologies used for the development process are HTML, CSS, JavaScript, JSP and Bootstrap framework. The back-end code is written in Java Servlets. The database used for this application is SQL. The server used for this application is

Apache Tomcat and it runs fine without any issues. The file creation has been implemented through Apache POI API.

The application consists of three modules namely, upload questions (create question repository), view question repository and generate question paper.

A. Upload Questions

This web interface allows the subject handler to upload questions or create a question repository which can be accessed at any point in time. The questions can be uploaded according to the subject code, subject name, chapter (unit) and part type. According to the type of question mentioned, the appropriate question template is rendered wherein the user can enter the question and difficulty level tag. There is also a provision to upload an image along with the question for subjective and essay type questions. The multiple choice question template consists of input fields like a question, option 1, option 2, option 3, option 4 and the difficulty tag. The subjective and essay-type questions share a similar template wherein the user has to enter the question, attach an image if required, and also mention the difficulty level of that particular question.

The upload questions interface is depicted in the following images:

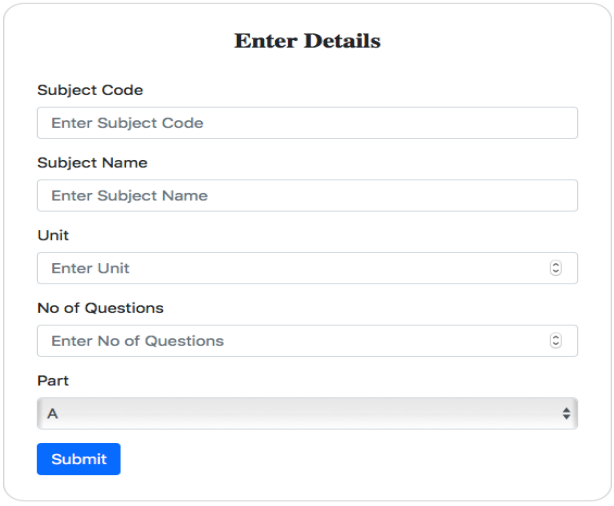

Fig 3:- Upload Questions Interface

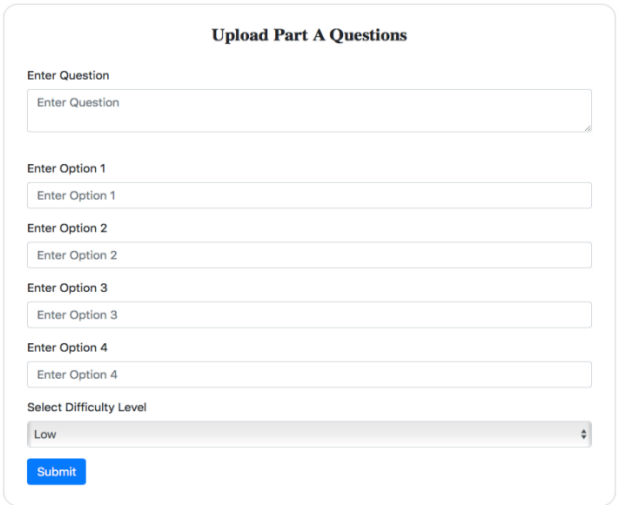

Fig 4:- Interface to upload questions for Part A

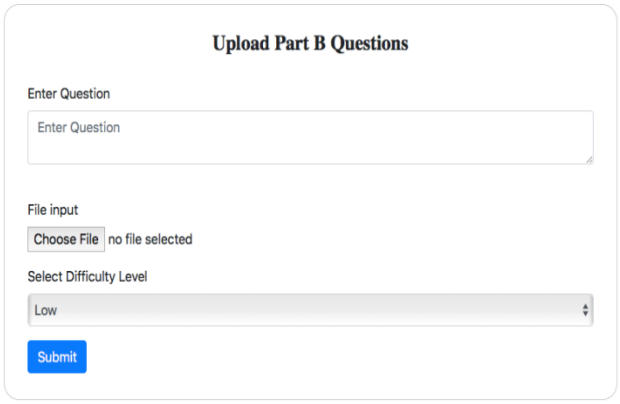

Fig 5:- Interface to upload questions for Part B

B. View Questions

This web interface allows the subject handler to view the questions from every unit in a particular subject or course based upon the question type. The view questions template is rendered based on the part-type that is specified as input. The unique question id is displayed along with every question. It also gives a provision to edit and delete a particular question when required. The edit option allows the user to modify any value pertaining to a particular question except the question id which is a read only field. In case of subjective and essay-type questions, the subject handler can also change the image that is associated with the question, if required. The delete option removes the complete record from the database.

The view questions interface is depicted in the following images:

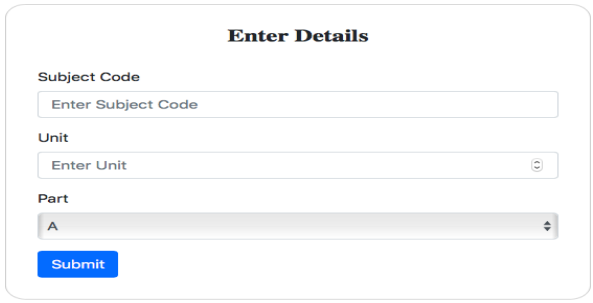

Fig 6:- View Questions Interface

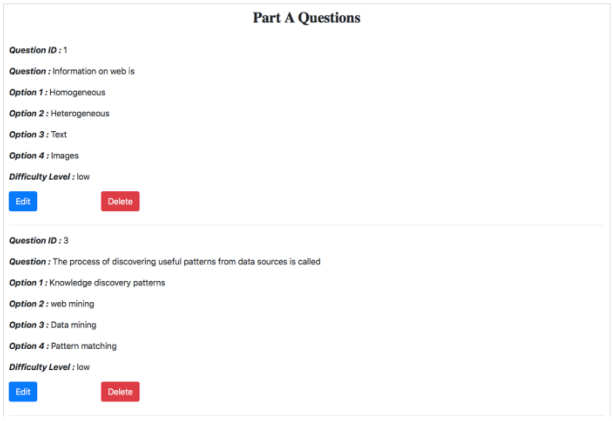

Fig 7:- Interface to view Part A questions

ISSN No:-2456-2165

Fig 8:- Interface to view Part B questions

C. Generate Paper

This web interface accepts the question paper specification details like the test details and paper pattern details. Based on these inputs, it generates a question paper which can be downloaded by the subject handler.

The generate paper interface is depicted in the following images:

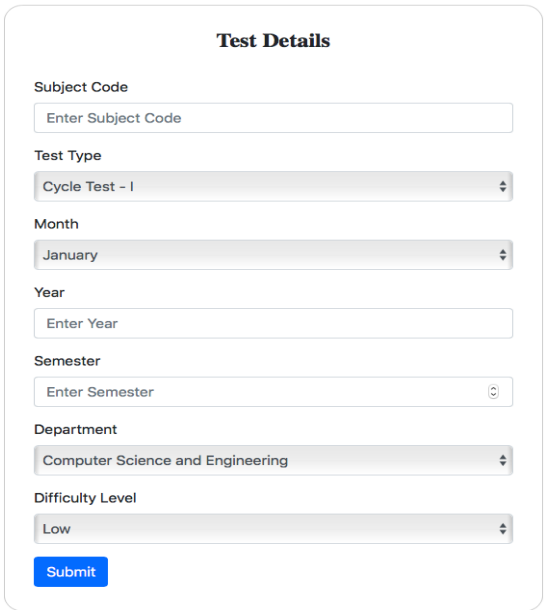

Fig 9:- Interface for test details

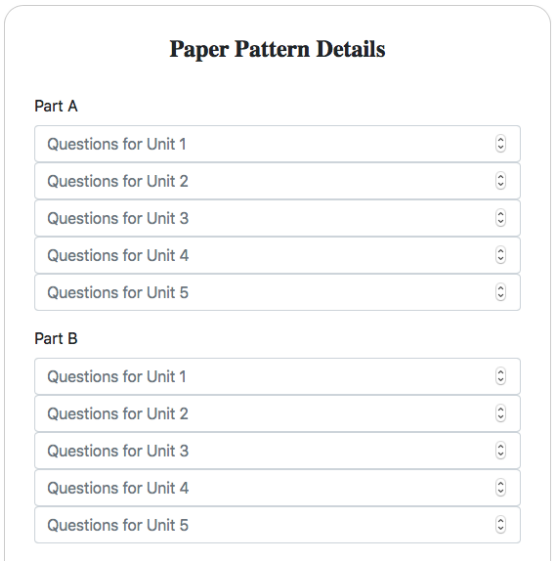

Fig 10:- Interface for paper pattern details

The paper pattern details interface takes the number of questions from each unit as input in order to select the set of questions. The questions are then randomly chosen based on the user's mentioned requirement.

V. CONCLUSION AND FUTURE SCOPE

Frequently generating quality question papers is a daunting task and our system overcomes the shortcomings of the existing paper-based system. It provides an easy-to-access user interface to the subject handlers wherein they can efficiently manage their respective question repositories for every subject. Our system also addresses the issue of duplicate entries in the question repository, that is, if two same questions have been selected, then it removes the duplicate value before placing it into the question paper. The generation of question paper is quick and automated and downloadable with the click of a button. Hence, it provides the flexibility which is lacking in the current system.

Even though this system addresses almost every shortcoming of the current system, there is always a scope for additional enhancements. Currently, this system has been designed to address the requirements of the test paper pattern followed by SRM Institute of Science and Technology. Hence, it can be modified before implementing in any other school, college or university. Another modification that can be made to this system is to give a provision to generate paper in multiple file formats other than word documents like XML or PDF.

REFERENCES

[1]. Itziar Aldabe, Maddalen Lopez de Lacalle, Montse Maritxalar, Edurne Martinez2, and Larraitz Uria, "Arik Iturri: An Automatic Question Generator Based on Corpora and NLP Techniques", ITS 2006, LNCS 4053, pp. 584 – 594, 2006.

[2]. Li-Chun Sung, Yi-Chien Lin, Meng Chang Chen, "The Design of Automatic Quiz Generation for Ubiquitous English eLearning System", 2006.

[3]. Ming Liu, Rafael A. Calvo, and Vasile Rus, "Automatic Question Generation for Literature Review Writing Support" (2010).

[4]. Ramesh, R., Mishra, S., Sasikumar, M., Iyer, S., "Semi-Automatic Generation of Metadata for Items in a Question Repository", Technology for Education (T4E), 2014 IEEE Sixth International Conference, Dec 2014.

[5]. Prita Patil and Kavita Shirsat, "An Integrated Automated Paperless Academic Module for Education Institutes," International Journal of Engineering Science Invention Research and Development issue IX, March 2015.

[6]. Rohan Bhirangi, Smita Bhoir "Automated Question Paper Generation System", ijermt, 2278-9359,April 2016.

[7]. Kapil Naik, Shreyas Sule, Shruti Jadhav, Surya Pandey, "Automatic Question Paper Generation System Using Randomization Algorithm", (IJETR) ISSN: 2321-0869,Volume-2, Issue-12, December 2014

[8]. Gauri Nalawade, Rekha Ramesh, "Automatic Generation of Question Paper from User Entered Specifications using a Semantically Tagged Question Repository", 2016 IEEE 8th International Conference on technology for Education.