GENESIS - A RANDOM QUIZ GENERATOR

A Project Report
Submitted to: **Dr. R. K. Ghosh**
for partial fulfilment of course
CS632: Topics in Distributed Systems

By:
Aakanksha Verma
(18111261 - Ph.D.)
Abhishek Gupta
(18111001 - M.Tech)
Kapil Kumar
(18111029 – M.Tech)

Department of Computer Science & Engineering
INDIAN INSTITUTE OF TECHNOLOGY KANPUR
Kanpur 208016, INDIA

TABLE OF CONTENT

Abstract

Examination process is an important activity of educational institutions to evaluate a student's performance. Current technologies help the instructors to store the questions in computer databases. The issue arises that how the current technologies would also help the instructors to automatically generate the different sets of questions from time to time. **Genesis** - A Random Quiz Generator performs all tasks related to setting of paper randomly for a given quiz number and for each student, starting from preparing question bank to displaying the paper when a student attempts the quiz. The main role is to provide randomization technique in generating question paper in which different set of questions could be generated for each student in a distributed system.

Introduction

Genesis (Random Quiz Generator) can be used in schools, institutions, colleges, and by other test paper setters who want to have a huge database of questions for frequent generation of question. This software can be implemented in various medical, engineering and coaching institutes for theory paper. You can create random question paper for each student with this software anytime within seconds. One can make many sets of paper from one database. This software assures no duplicity of questions in database.

Six types of questions are provided:

1. Multiple Choice Type
2. Numerical Type
3. Dropdown Type
4. Fill in the blanks Type
5. Short answer Type
6. Essay Type

Question paper is generated with great ease and accuracy in less than a minute. Record of generated question papers are kept in the database. Question paper is generated dynamically and randomly when a student clicks on the button to attempt a quiz. Full customization of quiz configuration is provided to the instructor. Number of questions of each type, total number of questions, marks for each type of question and maximum marks along with the duration of the quiz can all be set by the instructor.

It is a distributed application for distributed online quiz. The project also includes organization of question bank, and making system highly available. High Availability is provided using replicated servers and static load balancing technique improves the throughput of the whole system.

General Requirements

- Distributed database
- More than one server to handle request
- Load balancing
- Random quiz generation
- Question paper Requests using RPC
- Fault tolerance
- Session Maintenance
- Quiz configuration
- Quiz Response Store

Technical Aspect

FRONT END

- HTML - To make a webpage
- CSS - To design the webpage
- JAVASCRIPT - To validate the forms in web page
- MATERIAL TEMPLATE - Open source bootstrap template for designing

BACK END

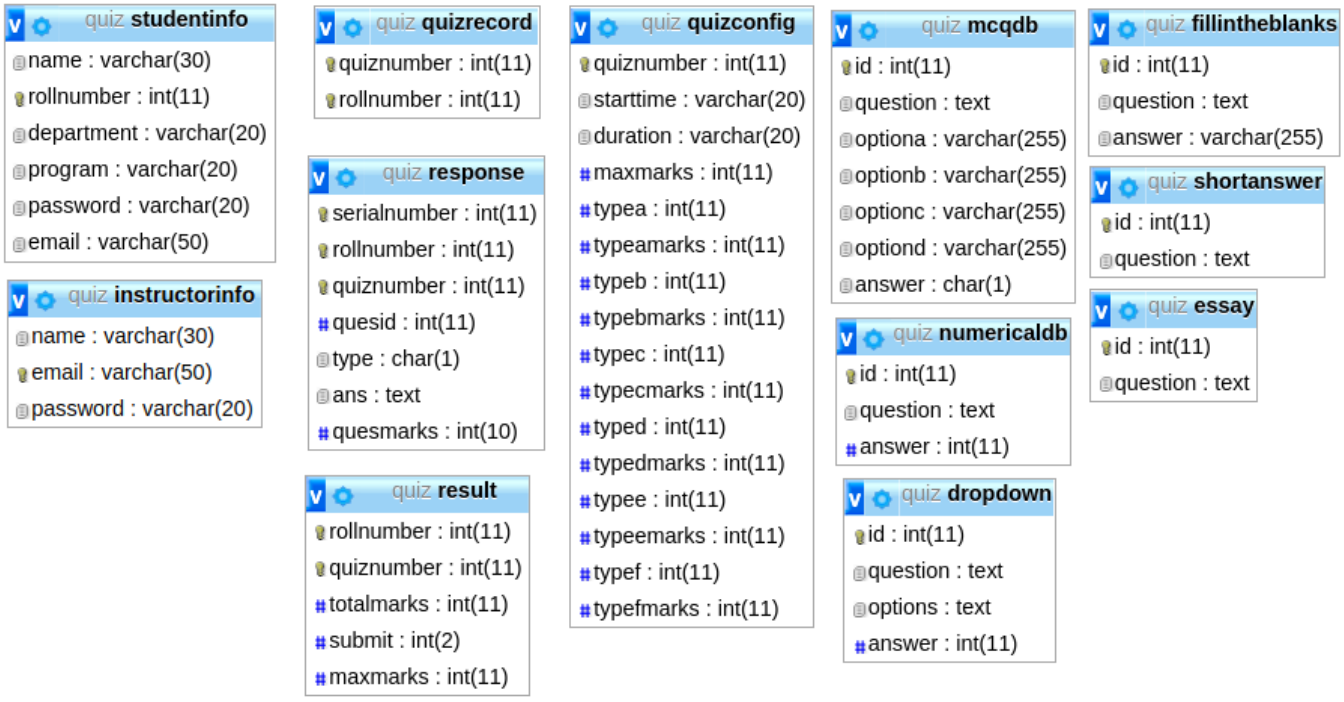
- PHP - To generate server-side scripting
- MYSQL - To store and manage database
- APACHE SERVER - To make system a web server
- JAVASCRIPT - To validate the forms in web page
- TCP CONNECTION - To establish connection

DEVELOPMENT TOOLS

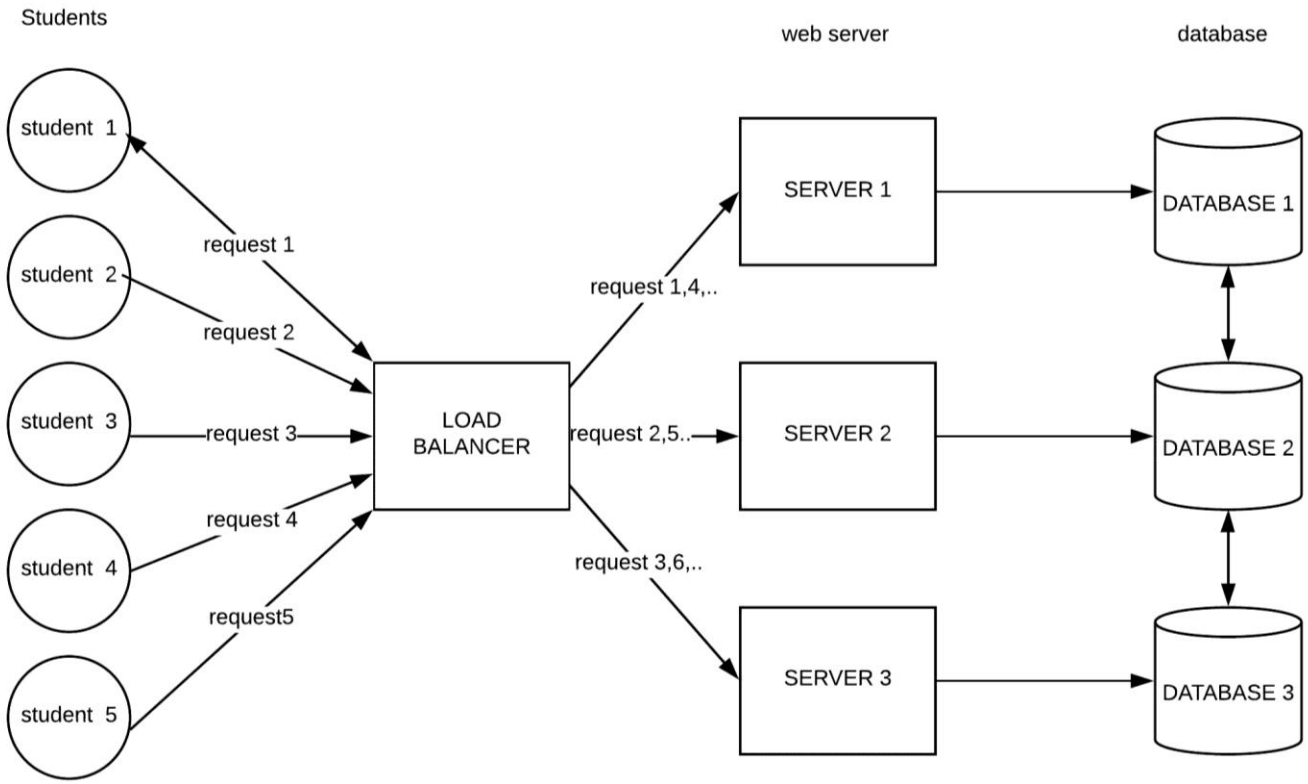
- SUBLIME TEXT - To edit php files
- LINUX - OS used for development
- PHPMYADMIN - Manage Database via GUI
- MYSQL SERVER - Manage Database users

Database Design

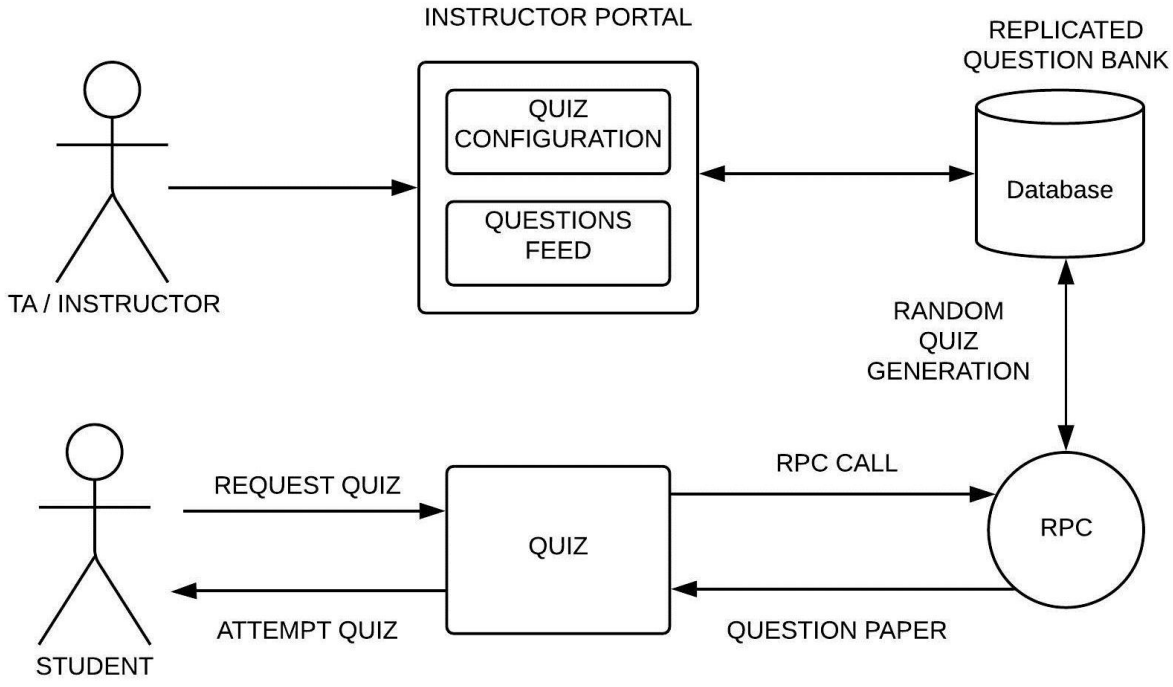
TABLE NAME	***PURPOSE***
studentinfo	Store information about the registered students
instrutorinfo	Store information about the Teacher and TAs
quizrecord	Store information about latest quiz given by students
response	Stores answers of the questions submitted by students
result	Stores information about marks of students
quizconfig	Stores configuration about quiz
mcqdb	Stores questions which have 4 choices
numericaldb	Stores numerical type questions
dropdown	Stores question which have multiple choices
fillintheblanks	Stores fill in the blanks type question
shortanswer	Stores short answer type question
essay	Stores essay type answer



Architecture Model



Use Case Model



Modules and Work Done

1. *Instructor Portal*
2. *Student Portal*
3. *Load Balancer and Scalability*
4. *Question Bank and Replication*
5. *Random Quiz Generation*
6. *Quiz Portal*

INSTRUCTOR PORTAL

Teacher and TAs will login through instructor portal using their registered email id and password. They can configure the quiz. Quiz configuration will be same for every student but questions will be randomly generated for every student. They can feed questions either on the local server only or can further select the replication option to feed the same question on every other server.

STUDENT PORTAL

Student will login through student portal using their roll number and password. He/she can attempt a live quiz after login or can check his/her last attempted quiz. Quiz generated for a student will contain each question in order according to the type of question. Question can be accessed in circular order or a student can jump to any question at any time mentioning the question number.

LOAD BALANCER AND SCALABILITY

Load balancer will equally distribute the request among all available server. We have used three techniques for load balancing - Round Robin, Weighted Round Robin and Randomized. Load balancer decides which request is to be forwarded to which server. Load balancer keeps a record of every request to tell which request is forwarded to which server.

ROUND ROBIN - Round-robin load balancing is one of the simplest methods for distributing client requests across a group of servers. Going down the list of servers in the group, the round-robin load balancer forwards a client request to each server in turn. When it reaches the end of the list, the load balancer loops back and goes down the list again

WEIGHT ROUND ROBIN – This build on the simple Round Robin load balancing method. In the weighted version each server in the pool is given a static numerical weighting. Servers with higher ratings get more requests sent to them.

RANDOMIZED - A pool of servers as an IP address is available at load balancer. Load balancer select an IP address randomly when a request arrived at load balancer and forward the request to the selected IP address.

QUESTION BANK AND REPLICATION

The whole question bank is classified into different types of questions. All except for the short answer type and essay type questions are fed along with their respective correct answers into the database.

QUESTION BANK CREATION- Instructor and TAs are provided with the interface where they can feed different types of questions into the question bank.

REPLICATION- Whenever a question is fed at one server the same is question is replicated on all the servers thus maintaining consistency. "Write through" policy is used for this implementation.

RANDOM QUIZ GENERATION

Quiz is generated randomly using a simple random function call. All questions are selected randomly within the same pool of the question bank. Number of questions are selected from the quiz configuration set by the instructor and these many questions are selected from the question bank available at local server.

QUIZ PORTAL

After the login, Student can attempt quiz whenever a quiz in live. Only one question can be seen by the student at a time. All the question opens in a circular sequential manner. He/she can jump to any question and submit at any time. Marks for every question is available with every question. All the feed answer can be seen by the student before submitting the quiz.

Future Scope for Enhancement

- Same latency at every place
- Load balancing based on area of request
- The system can be improved by implementing Dynamic Load Balancing instead of Static Load Balancing.
- We have implemented client-side fault tolerance. The system can be made non-blocking by implementing server-side fault tolerance also.
- Each type of question can be further classified based on three levels: easy, medium, difficult so as to increase the customizability of quiz configuration.

Conclusion

The project work is an automated system that progresses from the traditional method of paper generation to an automated process, by providing controlled access to the resources. We have also considered the importance of randomization in the task of paper generation. For a particular each student gets a random set of questions complying with the set configuration for that quiz. We distinguish between instructors and students by their unique user ids and passwords. Therefore, the resultant automated system for question paper generation provides improvement in terms of controlled access to the resources, random generation of question papers and a secure platform. Large amount of load is tackled using load balancing, client-side fault tolerance is implemented using sticky session and replication is achieved using write through policy during question feeding. Hence, Genesis is oriented to work in distributed environment.

References

- https://material.io/tools/icons/
- https://demos.creative-tim.com/material-kit/docs/2.0/getting-started/introduction.html
- https://www.w3schools.com/
- https://kemptechnologies.com/in/load-balancer/load-balancing-algorithms-techniques/