# Design and Implementation of an Online Exam Paper Generation System Using Java Servlets, JSP, and Oracle Database

Vanshika Mittal
Dept. of MCA
GLBITM
Greater Noida, India
24150cn059@glbitm.ac.in

Dr. Lalan Kumar
Dept. of MCA
GLBITM
Greater Noida, India
lalan.kumar@glbitm.ac.in

*Abstract*—We made a web-based Online Exam Paper Generation System using Java Servlets, JSP and Oracle XE. We used Apache Tomcat v9.0. Followed a strict MVC structure. There are two roles: administrators who manage the question bank and registered users who create exam papers. Administrators can add, view and delete questions. Organize them by department, semester, subject, module and difficulty. Registered users can log in browse the question bank and create exam papers by selecting questions from medium and difficult pools. The interface is built with Bootstrap 4, so it looks good on any device.
We will tell you about the systems architecture, modules, database design, testing and what we plan to do

Keywords— online Exam, question paper generation Java Servlet, JSP, Oracle, MVC, JDBC, e-assessment

## I. INTRODUCTION

Making fair and balanced exam papers is a big problem for universities. Faculty members spend a lot of time choosing questions trying to cover every module at the level and formatting everything. The results are not consistent. Sometimes papers repeat questions. Sometimes exams leak.

So, we decided to make an Online Exam Paper Generation System using Java Servlets, JSP and an Oracle XE database. We wanted to use a tech stack that most colleges can use. There are two kinds of users: administrators who fill and manage the question bank and registered faculty or staff who generate exam papers.

The administrator builds up the question bank tagging each question with branch, semester, subject, module and difficulty. Medium or hard. When someone creates an exam, they filter by what they need review the questions pick out what works and with one click the system gives them a formatted paper to print.

This paper explains how we built the Online Exam Paper Generation System what ideas from the literature shaped our solution and where we need to improve. Automating exam paper generation has been explored in previous studies [1], [2], but challenges like consistency, security, and efficiency remain.

## II. RELATED WORK

Automating exam paper creation is not new. Kumar et al. [1] demonstrated how web-based systems using

## IV. MODULES & FEATURES

### *A. Authentication and Session Handling*

MainController handles login, grabbing user credentials. Checking them via the UserLoginDAOImpl. If the password matches the session stores the user's email. Routes them to home.jsp. If not, they see the login page again with an error. Logout is simple. It clears the session. Sends users to the home page.

### *B. User Registration*

Registration is almost the same. Submissions go to UserRegistrationDAOImpl, which tries to add the record. If the email exists Oracle throws an error, which we catch and return to the user with a status message. On success users are redirected to continue.

### *C. Question Bank Management*

Administrators add questions by submitting the details. The DAO handles storage. To view or delete questions the administrator filters by branch, semester or subject then. Manages the results.

### *D. Exam Paper Generation*

Step one: set the papers metadata. The DAO fetches questions by those filters organizes them into medium and difficult lists and sends them to the selection page.

Step two: users select the questions they want grouped by difficulty. The servlet reads their selections wraps them in a SelectedQuestion object and forwards to the paper view, which outputs a printable exam paper formatted and divided into difficulty sections.

### *E. Supporting Pages*

There are some pages. Landing, About and Contact. To help users navigate. A quick email check page gives real-time feedback during registration. The home dashboard is the jump-off for all functions.

## V. TESTING AND RESULTS

Testing was done manually running Oracle XE locally. We checked every flow: new user registration, duplicate email handling, proper and failed logins adding various questions, filtering lists, deletion, creating papers and logging out. We also checked injection protection by submitting characters and SQL into input fields. PreparedStatements were used to prevent SQL injection attacks, as suggested in prior security studies [4].

Session handling works. Try to reach a protected page without logging in. You're bounced back to login. In terms of scale the codebase contains one controller, four DAOs, six entity classes, seventeen JSPs and a database connection class. Page loads for heavy question lists were

PHP and MySQL can efficiently manage question banks and generate exam papers. We followed the relationship-focused design but switched to Java EE for better type safety and easier deployment.

Sharma and Singh [2] emphasized that difficulty-based question selection improves test reliability. We borrowed that idea. Kept the final selection in human hands. It's the teacher who chooses, not the algorithm.

For architecture Patil et al. [3] highlighted that MVC architecture improves maintainability by separating concerns in Java EE applications. That's why all requests in our system go through one servlet and hand off to clean specialized DAO classes.

Security is another concern. Islam et al. [4] analysed security vulnerabilities in academic portals, identifying SQL injection and weak authentication as major risks. We locked down our database layer with PreparedStatements. We still have plain-text passwords, which is something we need to fix.

What makes our approach different? We kept the whole process manual and transparent. The faculty member curates the paper, supported by good filtering tools. The Online Exam Paper Generation System does not try to replace their judgment just make things smoother.

## III. SYSTEM ARCHITECTURE

### *A. Technology Stack*

The system has three layers. The front end uses JSP and Bootstrap 4 for a responsive interface. All the business logic runs in Java Servlets handling requests, sessions and calling the database layer. For storage we use Oracle XE 11g with JDBC relying on PreparedStatements to keep things secure. The implementation is based on standard Java Servlet and JSP principles [5], [6].

TABLE I.

| Component | Technology Used |
|---|---|
| Frontend | JSP + Bootstrap 4 |
| Backend | Java Servlets (Java EE) |
| Database | MySQL 8.0 |
| Server | Apache Tomcat v8.5 |
| DB Access | JDBC PreparedStatement |
| Build Tool | Maven |
| IDE | Eclipse IDE |

Fig. 1. Technology Stack of the Online Exam Paper Generation System

### *B. MVC Design Pattern*

We followed the MVC pattern as recommended in prior studies [3]. Entity classes handle the data. DAOs contain all our SQL, mapped carefully per entity. There's one MainController servlet that reads what users want spins up the entities calls DAO methods puts results in the session and forwards to the right JSP. Our JSPs use JSTL to render keeping Java code to a minimum in the views.

When we added difficulty-based filtering it only meant tweaks in one DAO and one JSP. The controller stayed the same. That's the beauty of separation. still fast clocking under 250ms on a laptop.

### *C. Database Schema*

We kept the database simple: two tables. Questions. Using the question text as a key makes some operations simple. We know it's not ideal. For the version we'll add a numeric key.

Biggest issue? Passwords are stored in text, which isn't secure. Another problem: using the text of the question as a key. So, no duplicate wording, even across different subjects. Both are on our to-do list.

## VI. CONCLUSION

Working on this Online Exam Paper Generation System gave us hands-on experience fitting Java EE parts together for academic needs. Sticking with MVC paid off. Adding features meant touching one DAO and one JSP leaving the main servlet untouched. Writing the JDBC layer was painless, and Bootstrap 4 made the UI responsive out of the box.

Where does this leave us? The Online Exam Paper Generation System does what it's supposed to: organizes a bank of questions lets you drill down with filters and pulls together well-formatted papers. Testers said they saved a ton of time compared to hunting through Word documents.

Still before calling this production-ready there's a punch list: hash passwords add a key for questions put some rate limits or CAPTCHAs on logins verify emails on registration and add export-to-PDF for direct printing. The way we built the DAOs means we can slot these in one by one without undoing what's already there. In hindsight that flexibility is the win of a solid MVC approach.

## ACKNOWLEDGMENT

Special thanks to the Department of MCA, GLBITM, Greater Noida for all the support and feedback, during this project.

## REFERENCES

[1] S. Kumar, A. Verma, and R. Singh, "Online Examination System Using PHP and MySQL," International Journal of Computer Applications, vol. 118, no. 6, pp. 1–5, 2015.

[2] R. Sharma and M. Singh, "Difficulty-Based Question Selection in Computer-Adaptive Testing for E-Recruitment Systems," in Proc. International Conference on Computational Intelligence, 2018, pp. 45–50.

[3] A. Patil, R. Desai, and S. Kulkarni, "Maintainability Analysis of MVC Architecture in Java EE Web Applications," International Journal of Software Engineering, vol. 9, no. 3, pp. 12–18, 2017.

[4] M. Islam, T. Ahmed, and H. Kabir, "Security Analysis of Web-Based Academic Portals," in Proc. IEEE International Conference on Cyber Security, 2020, pp. 210–215.

[5] J. Hunter and W. Crawford, Java Servlet Programming. Sebastopol, CA, USA: O'Reilly Media, 2001.

[6] B. Basham, K. Sierra, and B. Bates, Head First Servlets and JSP, 2nd ed. Sebastopol, CA, USA: O'Reilly Media, 2008.