International Journal Software Engineering and Computer Science (IJSECS)
4 (3), 2024, 1292-1304
Published Online December 2024 in IJSECS (http://www.journal.lembagakita.org/index.php/ijsecs)
P-ISSN: 2776-4869, E-ISSN: 2776-3242. DOI: https://doi.org/10.35870/ijsecs.v4i3.3285.

RESEARCH ARTICLE **Open Access**

Development of a Web-Based SQL Query Online Examination System with Automated Grading Using the MVC Design Pattern

Yoppy Yunhasnawa *
D4 Informatics Engineering, Politeknik Negeri Malang, Malang City, East Java Province, Indonesia.
Corresponding Email: yunhasnawa@polinema.ac.id.

Atif Windawati
D4 Islamic Banking, Politeknik Negeri Semarang, Semarang City, Central Java Province, Indonesia.
Email: atifwinda@polines.ac.id.

Toga Aldila Cinderatama
D4 Informatics Engineering, Politeknik Negeri Malang, Malang City, East Java Province, Indonesia.
Email: toga.aldila@polinema.ac.id.

Candra Bella Vista
D4 Informatics Engineering, Politeknik Negeri Malang, Malang City, East Java Province, Indonesia.
Email: bellavista@polinema.ac.id.

Moch. Zawaruddin Abdullah
D4 Informatics Engineering, Politeknik Negeri Malang, Malang City, East Java Province, Indonesia.
Email: zawaruddin@polinema.ac.id.

Received: October 28, 2024; Accepted: November 20, 2024; Published: December 1, 2024.

Abstract: In this study, the Researchers provide the design, development, and evaluation of a web-based SQL exam system that utilizes the Model-View-Controller (MVC) architectural pattern to enhance automated grading functionality and ease of maintenance. The main objectives of the system are to simplify the process of administering SQL exams, to make it user-friendly for students to enter their SQL statements, and as a means for teachers to automate the grading process. This allows a clear separation between three separate modules: Model to manage data, View to present the application to users, and Controller to manage the application logic. This separation allows for modular development, easier maintenance, and code reuse. The fundamental aspect of the system lies in its automated grading mechanism, which intelligently compares the SQL queries submitted by students with the corresponding validated answer keys stored in the database. Extensive black-box testing was conducted to ensure the reliability and accuracy of the system with various test cases to assess its ability to assess responses and provide real-time feedback to students, in addition to smooth and intuitive navigation within the system. All testing criteria yielded successful results with 100% agreement proving the robustness of the system with all possible locations that could potentially be used in higher education structures. The system provides a scalable and flexible approach to address the challenges associated with SQL assessment in academic institutions, thereby facilitating uniform, efficient, and objective evaluation standards. The system uses data up to October 2023 to prevent the model from becoming obsolete.

Keywords: Web-based SQL Examination System; Automated Grading; Model-View-Controller (MVC) Design Pattern; Online Assessment; SQL Query Evaluation.

1. Introduction

Database Maintenance is one of the important courses that must be taken by IT majors in the course because this course helps students know about the details related to how to store, retrieve, and organize data [1]. Structured Query Language (SQL) is one of the most important courses in database studies. SQL is a language used to manage and manipulate data in Database Management Systems. SQL has been suggested for study programs driven by professional data development [2], and therefore students learn SQL, which seems to be widely used to solve professional data problems for IT applications and development in many fields and potentially across industries. SELECT Statement – The SELECT statement is one of the main types of SQL statements which is a type of syntax used to 'request' data from a database. The SELECT statement has many syntactic variations and is an important concept for students studying databases to learn because this statement provides information about the data in the database, expenditure views, and reports [3].

Assessment is an integral part of the teaching process. For reflection, students can identify the extent of their understanding of the material taught [4]. In addition, evaluation provides feedback to instructors on the usefulness of their teaching methods and the extent to which learning objectives have been met [5]. Meaningful assessment also shows where students are struggling, meaning instructors can provide more help if needed. This is important to ensure that all students have a fair chance to improve.

Manually grading SQL SELECT exams for students is a challenging process. The instructor has to read each student's responses individually and check them against a set answer key. Furthermore, if the SQL SELECT query being tested is complex, the instructor may have to run the student's SQL answers, in a real DBMS, to check whether the resulting data set is correct. This can be a long and tedious process and can slow down the feedback each student receives, which in turn makes the learning process less efficient. Furthermore, human error associated with manual grading can also hamper the redundancy and fairness of evaluation results [6]. Therefore, there is an urgent need to find faster and more accurate ways to solve the problem.

Web technology has introduced automation to the process of conducting exams and evaluating SQL SELECT exams [7]. The purpose of this study is to design and develop an online exam system that evaluates student work directly in the system with feedback displayed immediately after the exam is completed. This system works by storing the database on the server, students access the system through a web application that implements the Model-View-Controller (MVC) design pattern as a layer between the database and students. By automating the evaluation process, this system is expected to increase the efficiency of the overall assessment procedure, reduce the workload of instructors, and provide fast and accurate feedback to students. Evaluation and teaching also become very flexible and available from anywhere with this system.

An online exam system is a digital system used to conduct exams on the internet. This system combines various technology modules to handle registration operations, question sharing, answer submission, assessment, and result announcements. Online exam systems offer a scalable solution for authentication and integrity in higher education [8]. One of the best solutions to this problem is the implementation of an online exam system in educational institutions, which is based on intelligence, which requires less infrastructure. Such online exam systems are platforms such as Google Forms, Moodle, Blackboard, etc. Google Forms offers a simple solution for creating quizzes and surveys with a variety of question formats. Moodle is a powerful Learning Management System (LMS) that allows educators to create online exams with a variety of question formats and instant grading features. Another LMS, Blackboard, offers similar functionality, combined with integration tools for educational resources.

SQL online exam systems are intended to test the SQL proficiency and knowledge of the participants. While a typical online exam system may only include multiple choice or essay questions, SQL exam systems require participants to write and execute SQL queries directly on a database. The system for automatically executing and grading these SQL queries needs to provide appropriate feedback based on the results of their execution as quickly as possible, thus lightening the workload of teachers as it is very high [9]. SQL exam systems available online include HackerRank, Codility, SQLFiddle, etc. HackerRank and Codility offer coding platforms including SQL skill assessments using coding challenges and exams. SQLFiddle is another simpler option, where users can write and test SQL queries in a web browser, providing a real-world method for evaluating SQL skills. Design patterns are general, reusable solutions to common problems in a particular context in software design. There are dozens and dozens of design patterns that can help with common software engineering problems, frameworks for best practices, and optimizing development. The use of design patterns can yield many benefits, such as increased productivity and efficiency, by implementing well-known solutions, improved quality, because design patterns are well-accepted best practices, better communication, because some parts of design patterns can serve as common terminology, and easier maintenance and further development with increased reuse and matching of design and implementation [10][11]. There are several design patterns that are widely used in designing web application frameworks, such as:

1) MVC (Model-View-Controller)
   Separates the application into three main elements: model, controller, and view, maintaineing layers with defined functions in order to enhance modularity and separation of concerns [12].
2) MVVM (Model-View-ViewModel)
   Similar to MVC but focuses more on separating the view logic from the business logic, with closer data binding. This pattern consists of three components: Model, View, and ViewModel, which separates domain logic and display logic for easier tests and maintenance [13].
3) Singleton
   The Singleton Design Pattern guarantees that a class has only a single instance and offers global access to that instance. This approach ensures that only one object of a specific type is present in the application, addressing issues such as managing unique resources or maintaining consistent operations [14].
4) Factory
   Provides a way to create objects without specifying the exact class of the object to be created, this pattern simplifies the creation of class instances [15].
5) Observer
   Enables an object to inform other objects when its state changes. The Observer pattern manages how one object registers interest in and receives updates about changes in another object. This approach facilitates coordination among large and complex groups of objects with minimal interdependence [16][17].

MVC (Model-View-Controller) is a design pattern that divides an application into three main components: Model, View, and Controller. The Model is responsible for managing data and business logic. The View is responsible for presenting data to the user. The Controller acts as an intermediary, managing user input and updating both the Model and the View. The main advantages of MVC include:

1) Separation of Concerns
   MVC divides business logic and data handling from the user interface, enhancing the system's performance, scalability, and maintainability [18].
2) Accelerated Development and Improved Quality
   In enterprise development, MVC simplifies the process, speeds up development, and enhances the overall quality and maintainability of the software [19].
3) Enhanced User Interface Creation
   Patterns stemming from MVC, like the Composite View pattern, support the development of complex interfaces by combining multiple views [20].
4) Greater Scalability and Maintainability
   The Service Layer pattern, an MVC extension, introduces an abstraction between the data access and controller layers, increasing both scalability and maintainability [20].
5) Reduced System Complexity
   MVC structural approach helps manage large, complex systems, such as e-commerce platforms, by minimizing system complexity [18].

In this study, the MVC design pattern was chosen due to its suitability for the SQL examination system being developed, particularly for its separation of concerns and code reusability. Separation of concerns is essential, as the online SQL exam system includes a user interface that collects SQL answer input from examinees, a main component that controls the checking and display of exam results, and a component directly interacting with the database to execute students' SQL answers, which are then compared against the answer key. These three components, from an MVC perspective, can clearly be categorized into View, Controller, and Model. Additionally, the characteristic of code reusability is also needed, given the potential future development of this system to add important features, such as DML (INSERT, UPDATE, DELETE) capabilities, or other types of SQL statements. Thus, the development of an MVC-based online SQL exam system is expected to provide an efficient and effective solution in assessing students' SQL skills, reducing the workload of teachers, and providing fast and accurate feedback, thereby improving the overall quality of learning.

2. Research Method

In this study, the online SQL SELECT examination system is implemented as a web-based application accessible to examinees through their browsers via a local network or the internet. This web application is developed using PHP and follows the Model-View-Controller (MVC) architecture, consisting of Model, View, and Controller components. The web application is connected to a database server that contains several tables. Two of the most important tables are the questions table, which stores the list of questions for each SQL SELECT exam session, and the answer_keys table, which stores the correct SQL answers for each question in the questions table.

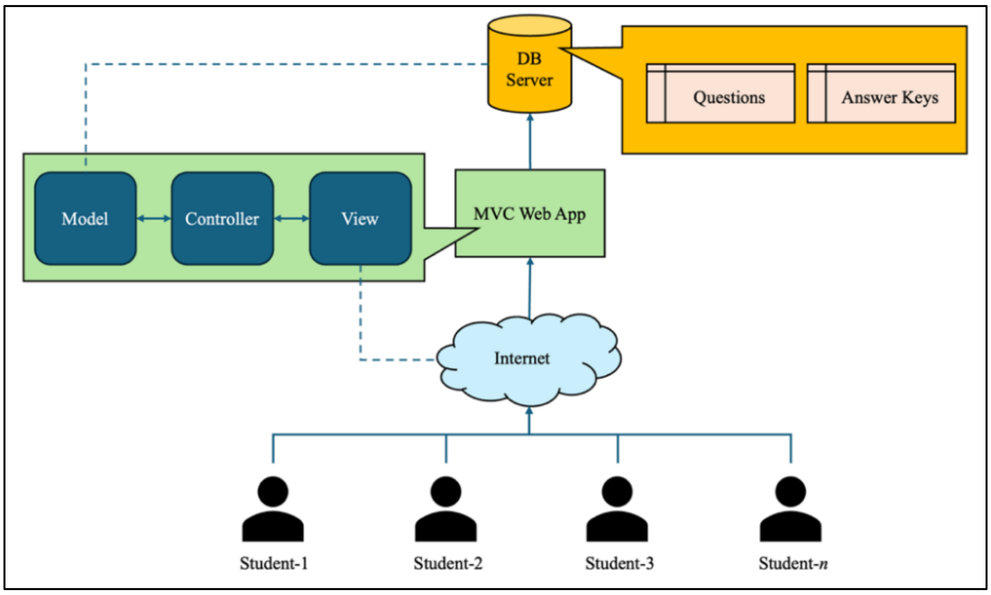


Figure 1. Overall System Architecture

As depicted in the system architecture diagram, the View component of the web application is the front-facing component that directly handles requests from examinees over a local network or the internet. This component then forwards the requests to the Controller, which interprets them and manages the answers that need to be evaluated. These answers are processed by the Model, which communicates with the database to retrieve question data and answer keys. While the system is database-agnostic, the default database configured for use is MySQL.

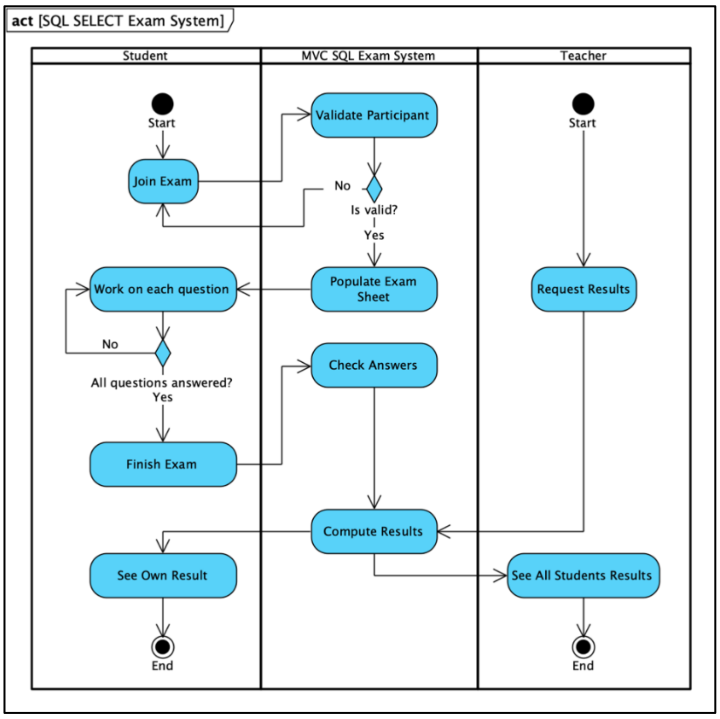


Figure 2. Exam Session Flow

The app contains two user types: instructors and examinees. The instructor prepares the exam sessions with the SQL questions and the desired answers. The questions and answer keys are subsequently uploaded to the database server, which will be used to evaluate the responses of the examinees. "You can have multiple examinees sitting in a session per exam date. Questions are accessed via web page opened in the browser by the examinees. Each examinee can independently conduct the exam session, or the instructor can gather everyone in one room. The instructor can see all the scores of all examinees stored in the database server on the end of the session. Above is the activity diagram showing the detailed flow of an exam session. The exam starts once the examinee opens the online exam application and completes the form for an exam session of its choice, as seen in activity diagram. In this version, the test taker gets to fill in personal details and upload their exam key file Online examination system validates the personal data and exam key file after the examinee submits the form. If the data is not valid, the examinee can re-run the process, but if the data is valid, the online examination system will generate the exam sheet with questions, and the text area to insert the SQL script responses to each question. Once the exam sheet is on the screen, the examinee has to start working on it. Once the examinee has answered all the questions, they may end the exam. The exam answers are then submitted online for grading, once the exam is completed. Scores are aggregated on the number of correct and incorrect answers, and the results are stored in the database. The examiner can view the scores of each examinee and each examinee's score can be viewed on their browser screen immediately after evaluation.

2.1 System Structure

The following class diagram illustrates the architecture of the online SQL examination system, designed based on the Model-View-Controller (MVC) design pattern. This pattern divides the system's structure into three main components—Model, View, and Controller—to enhance modularity, facilitate maintenance, and streamline the overall system functionality.

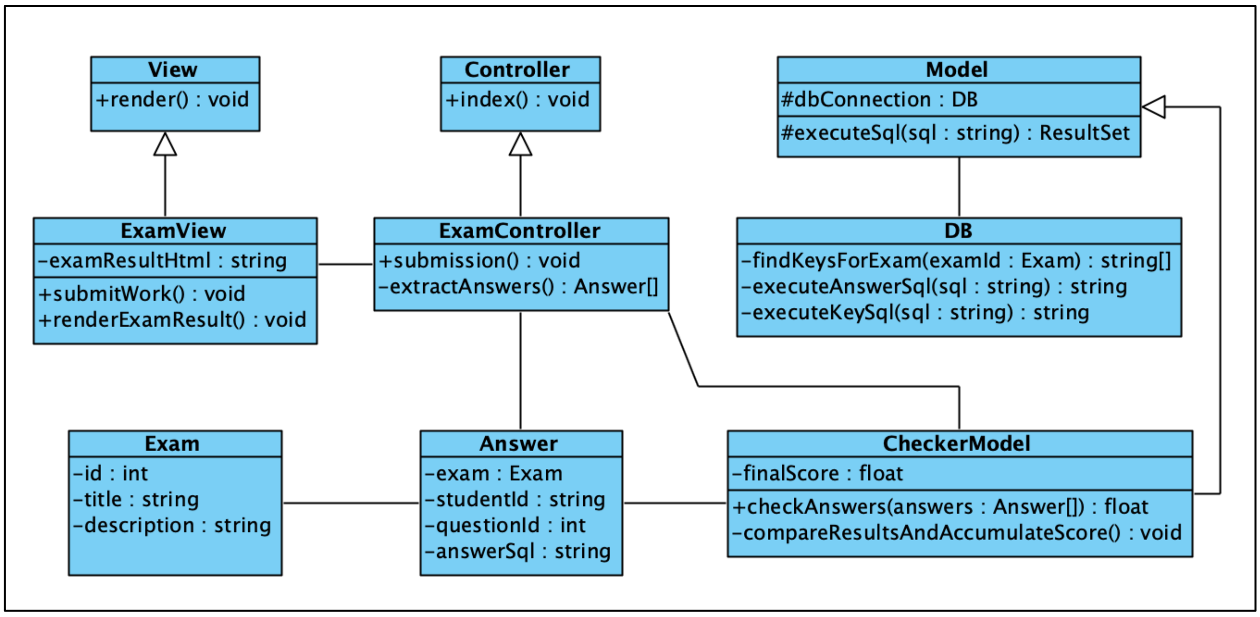


Figure 3. Class diagram depicting system structure.

The Model layer encompasses the core data and business logic of the system. This layer includes several key classes, such as the DB class, which extends the base Model class and is responsible for interacting with the database. The DB class provides methods like findKeysForExam() to retrieve answer keys and executeAnswerSql() and executeKeySql() to execute SQL queries related to students' answers and answer keys. This class plays an important role in data retrieval and storage operations required during the examination process. Additionally, there is the CheckerModel class, which is also part of the model layer and is responsible for verifying and scoring students' submitted answers. CheckerModel has attributes like finalScore and methods such as checkAnswers() and compareResultsAndAccumulateScore(), which are essential for evaluating students' responses and calculating their final score. This class directly interacts with the Answer class to ensure accurate assessment.

The View layer is represented by the base View class and a specific ExamView class. The View class serves as a general interface for displaying the user interface within the system, providing a render() method that can be extended by more specific views. ExamView is a subclass of View designed specifically to present exam-related content. This class includes an examResultHtml attribute to store the HTML content of the exam results and provides methods like submitWork() and renderExamResult() to facilitate the submission and display of exam results. ExamView ensures a smooth user experience for students as they interact with exam questions and view their results.

The Controller layer includes the base Controller class and a specific ExamController class, which manages the flow of operations between the Model and View layers. Controller acts as a general interface for handling user requests, with an index() method to initialize actions. The ExamController class extends Controller to handle exam-related requests specifically, including submission() for processing exam submissions and extractAnswers() for retrieving students' answers. ExamController acts as a bridge between the View and Model layers, ensuring that the correct data is displayed to users and processed by the Model layer.

Two core entities within this system are the Exam and Answer classes. The Exam class has attributes like id, title, and description, which describe each exam. This class encapsulates information about the exams being conducted. Meanwhile, the Answer class stores details about each student's response, with attributes such as exam, studentId, questionId, and answerSql. Each Answer instance is associated with a specific Exam instance and contains the SQL query provided by the student in response to a particular question. This class works closely with CheckerModel to enable accurate evaluation of each response.

Directed associations among these classes clarify the dependencies. ExamView relies on ExamController to handle user actions related to the exam, while ExamController depends on DB for database operations and interacts with CheckerModel for result processing. CheckerModel interacts with Answer to evaluate and score student responses. Additionally, Answer and Exam classes are linked, as each answer corresponds to a specific exam instance.

Overall, this class diagram illustrates a well-structured and modular system design, adhering to the principles of the MVC pattern. The Model layer provides data logic and operations, the View layer manages the user interface and interactions, and the Controller layer coordinates activities between the two. Each component has a distinct role, ensuring that the online SQL examination system can efficiently manage exams, validate student responses, and deliver results while maintaining a clear separation of responsibilities.

2.2 Automated Grading Mechanism

One of the main advantages offered by this online SQL examination system is its capability to automatically grade examinee responses. Instructors no longer need to review each answer individually, as the system automatically matches the answers submitted by each examinee with the answer keys stored in the database. This automated grading process can be observed in the following Sequence Diagram.

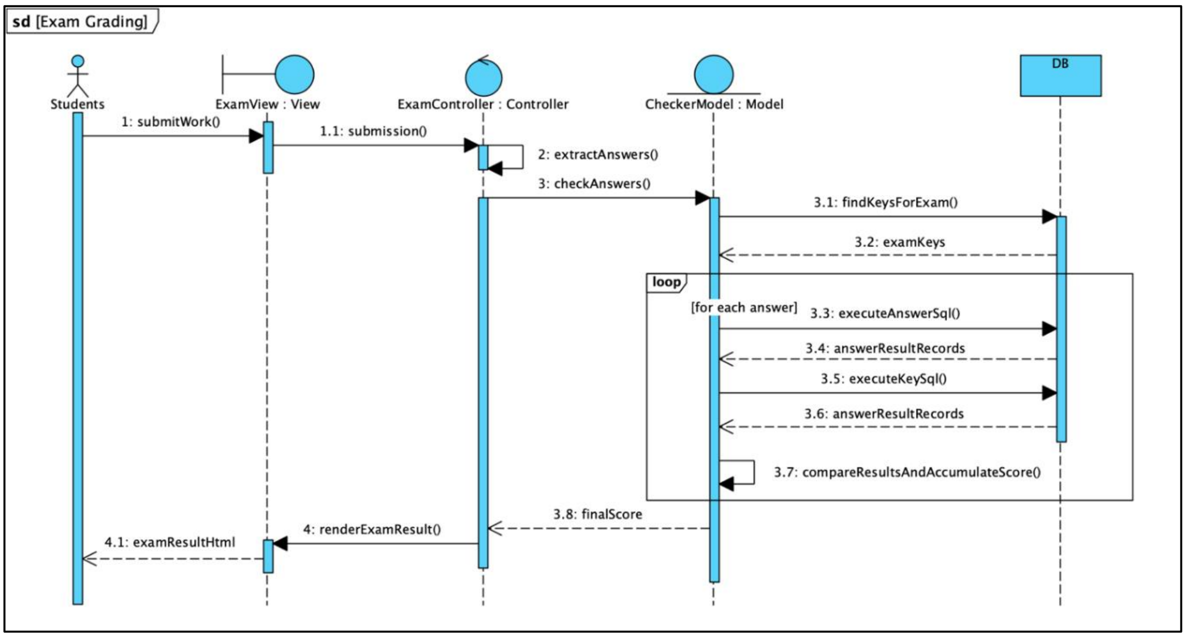


Figure 4. Sequence diagram depicting automated grading mechanism

The sequence diagram illustrates the flow of the automated grading process within the online SQL examination system, involving four main classes: ExamView, ExamController, CheckerModel, and DB. The process begins when an examinee (Student) submits their answer through the submitWork() method in the ExamView class. ExamView then calls the submission() method in ExamController to initiate the grading process. ExamController subsequently extracts the examinee's answers by calling the extractAnswers() method and passes these answers to CheckerModel via the checkAnswers() method. Within CheckerModel, the system begins verifying the examinee's answers. The first step is to retrieve the relevant answer keys from the database via the findKeysForExam() method, which interacts directly with the DB class. Once the answer keys are retrieved and stored as examKeys, the system enters a loop to evaluate each answer submitted by the examinee. In each iteration, the system runs the executeAnswerSql() method to execute the examinee's SQL query and stores the result as answerResultRecords. The answer key is also executed through executeKeySql(), and its result is stored in the same manner.

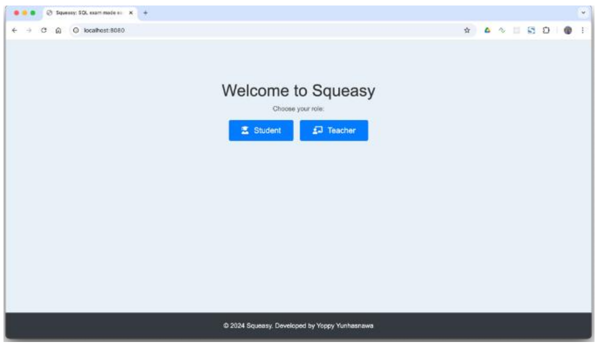
After all answers have been evaluated, the compareResultsAndAccumulateScore() method is used to compare the examinee's query results with the answer key results and then accumulate the final score (finalScore) based on the accuracy of the responses. This final score is returned to ExamController, which subsequently calls the renderExamResult() method in ExamView to display the grading results to the examinee in the form of examResultHtml. This explanation highlights the efficiency of the automated grading process, enabling examinees' results to be processed and displayed immediately without manual intervention from the instructor. By leveraging the classes within the system, this process ensures that each answer is checked accurately and quickly according to the answer keys stored in the database.

3. Result and Discussion

3.1 Results

3.1.1 System Implementation

After completing the system design and implementation process, a simple, concise, and user-friendly web-based online SQL examination system was produced. Below are some of the main interfaces of this system. The first display is the home page, which welcomes users with the message "Welcome to Squeasy" and provides two role options, namely Student and Teacher. Users can select one of these roles to proceed with the system interaction according to their access rights.

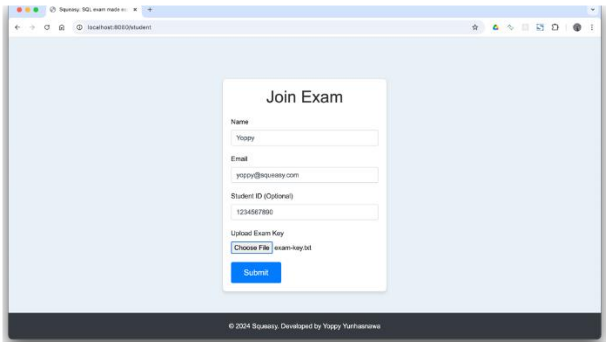

localhost:8080

Welcome to Squeasy

Choose your role:

© 2024 Squeasy. Developed by Yoppy Yunhasnawa

Figure 5. Home Page


localhost:8080/student

Join Exam

Name: Yoppy

Email: yoppy@squeasy.com

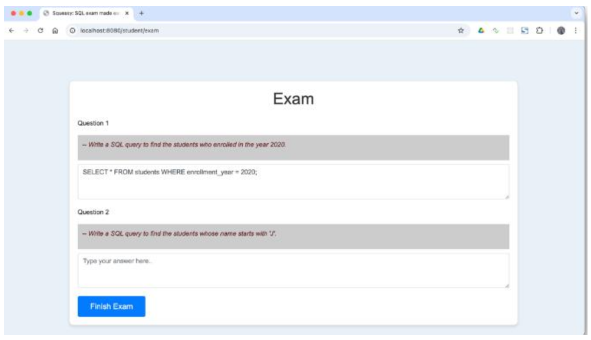
Student ID (Optional): 1234567890

Upload Exam Key: exam-key.txt

© 2024 Squeasy. Developed by Yoppy Yunhasnawa

Figure 6. Join Exam Page

When a student selects the student role, they are directed to the Join Exam page. On this page, students are required to fill in several important details, such as name, email, and optionally, Student ID. Additionally, students have the option to upload an Exam Key file, which is necessary to join the exam. After completing all the required information, students can click the Submit button to start the exam (Figure 6). Upon joining the exam, students are directed to the Exam page, which displays the SQL exam questions that need to be answered. Each question includes clear instructions, and students can type their SQL query as an answer in the provided input field under each question. After answering all the questions, students can finish the exam by clicking the Finish Exam button, which submits their answers to the system (Figure 7).


localhost:8080/student/exam

Exam

Question 1

-- Write a SQL query to find the students who enrolled in the year 2020.

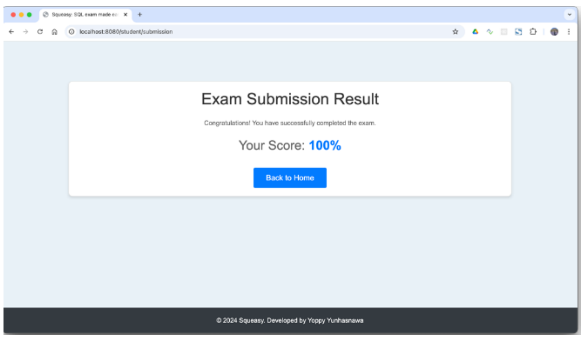
SELECT * FROM students WHERE enrollment_year = 2020;

Question 2

-- Write a SQL query to find the students whose name starts with 'J'.

Type your answer here...

Figure 7. Exam Sheet Page


localhost:8080/student/submission

Exam Submission Result

Congratulations! You have successfully completed the exam.

Your Score: 100%

© 2024 Squeasy. Developed by Yoppy Yunhasnawa

Figure 8. Exam Submission Result Page

After submitting their answers, students are directed to the Exam Submission Result page. On this page, students can view their final score, which is automatically calculated by the system based on the answers they submitted. This page displays a message confirming that the exam has been successfully completed, with the final score prominently displayed to provide immediate feedback to the students. At the bottom of the page, there is a Back to Home button, allowing students to return to the home page (Figure 8). This system is designed to provide an efficient examination experience, enabling students to take SQL exams independently with the support of automated grading, which simplifies the evaluation process for instructors. The system is run using environment as follows.

Table 1. System Environment

No.	Aspects	Values
1	CPU Core	1 vCPU
2	RAM	1 GB
3	OS	Linux 64 bit
4	Web Server	Apache 2
5	PHP Version	7
6	DBMS	MySQL 8
7	Browser	Chrome, Safari, Firefox (Latest Version)

3.1.2 System Testing & Validation

To ensure the reliability and functionality of the SQL auto-grading system, black box testing was employed as the primary method of validation. This approach was selected to focus on verifying the system's behavior against specified requirements without examining the internal code structure [26][27]. The testing process involved a comprehensive set of test cases designed to validate both the core grading functionality and the user interface (UI) elements of the system. The system was tested using various SQL queries, including correct, incorrect, and edge-case queries, to verify that the grading outputs were consistent and accurate. Additionally, UI functionality was tested to ensure that users could seamlessly navigate the system. The test cases covered scenarios such as opening the home page, logging in as a student, joining an exam, submitting responses, and viewing exam results. Each test case outlined expected behaviors and outcomes, which were then compared to the actual results observed during testing. The test cases included aspects like:

1) Valid SQL Query Returns Correct Result
   Verified that correct SQL queries submitted by students return the expected result and score.
2) Invalid SQL Syntax
   Ensured that the system provides error feedback when students submit syntactically incorrect queries.
3) SQL Injection Attempt
   Confirmed that security mechanisms prevent and reject potentially harmful queries.
4) UI Navigation Tests
   Included scenarios such as opening the home page, logging in as a student, and submitting exams to ensure smooth user experience.

The system underwent rigorous testing with all test cases, achieving a 100% pass rate. This result demonstrated that the SQL autograding system met its functional requirements, providing accurate grading, effective feedback, and a user-friendly interface. The system performed as expected under various test scenarios, reinforcing its reliability for real-world educational use. This comprehensive testing validated that the system not only handles SQL queries and grades them correctly but also offers robust and seamless navigation for both instructors and students. The table below shows the list of the test cases employed in this black box testing scenario:

Table 2. Black Box Test Cases

Test Case ID	Test Scenario	Input (SQL Query)	Expected Result	Result
TCG-01	Valid SQL query returns correct result	SELECT * FROM students;	Correct result set returned, score = 100%	Pass ✅
TCG-02	Invalid SQL syntax	SELEC * FROM students;	Error feedback provided, score = 0%	Pass ✅
TCG-03	SQL injection attempt	SELECT * FROM students; DROP TABLE students;	Rejection of query with security warning	Pass ✅
TCG-04	Query with a syntax variation (e.g., upper vs. lowercase keywords)	select * from students;	Correct result set returned, score = 100%	Pass ✅
TUI-01	Able to open home page	N/A	Home page opens without error	Pass ✅

TUI-02	Able to login as student	N/A	Student login successful, redirected to join exam page	Pass ✅
TUI-03	Able to join exam	N/A	Exam join process completes, exam page displayed	Pass ✅
TUI-04	Able to submit exam	N/A	Exam submission successful	Pass ✅
TUI-05	Able to see the result/score of the exam	N/A	Result page displays score feedback accurately	Pass ✅

3.2 Discussion

The research has successfully designed, implemented, and validated a web-based SQL exam system based on Model-View-Controller (MVC) architecture. The system is designed to simplify the entire SQL exam process, starting from exam creation, questions are created by the exam editor with SQL statements to be tested in the exam and this is evaluated by scoring according to the requirements of each exam. The results of the system implementation create an easy-to-use interface with a user-friendly home page, displaying user role options (students and teachers), a student exam registration page, an exam sheet page with SQL questions, and an exam result page that displays scores automatically. The system functions in a controlled setting where a 64-bit Linux operating system, Apache 2 web server, PHP version 7, and MySQL 8 DBMS are involved with any contemporary web browser, including Chrome, Safari, or Firefox. System validation is done using black-box testing, which addresses the specifications of the system's functionality without requiring internal code analysis. In this section, a complete set of test cases will be executed in a way that checks the accuracy of the assessment based on SQL queries, responses to invalid syntax, and also takes care to secure against SQL injection attacks with smooth navigation on the user interface. Test scenarios for positive and negative SQL query scenarios, syntax variations, and potentially malicious injection attempts were among the test cases used. Part of the testing also involved navigating the user interface, from opening the homepage to viewing the exam results. The system successfully passed the test cases with a 100% success rate. This means that the functional coverage requirements were met by the automated SQL scoring engine, thus accurately assessing users, providing helpful feedback and navigation as they use it. In addition, the system was also able to correctly process and assess SQL queries in test scenarios with syntax errors and security attack attempts, which were likely to succeed. The successful completion of these tests validates the system for use in a school environment, where accuracy and speed of assessment are of utmost importance. This resulted in an efficient testing process and objective and consistent assessment of exams across the board. The results of this study can be used as input for developing similar systems in the future that can take into account changing requirements and over time across educational institutions and can be integrated with other online learning platforms, thereby expanding their reach and benefits in education.

4. Related Work

While several efforts have been made to address SQL testing systems, and to suggest automated grading techniques, little attention has been paid to the maintenance and extensibility properties of the underlying software used in these systems. Studies by Wanjiru et al. Buchhave et. al (2022) and Kovacic & Green (2012) showed that automated grading systems are effective in decreasing instructor workload and in increasing grading uniformity [21][22]. Al-Salmi (2019) and Kleiner & Heine (2024), Others called for feedback mechanisms to be improved and greater support for awarding partial credits [23][24]. Also, a real-time feedback tool was presented by Kleerekoper & Schofield (2018), focused on enhancing student awareness and learning practice [25]. Although these contributions greatly enhance the efficiency of student assessment and learning, relatively few studies have addressed the ease of system development and maintenance, important factors that determine the long-term adaptability and sustainability of educational technology. The main aspect to be considered to allow educational systems to adapt to continuous changes is related to the maintainability and ease of software development, considering that it would make new updates, bugs fixes, and new features addition feasible. The Model-View-Controller (MVC) design pattern is an efficient way to prevent many of these problems, as it divides the system architecture into layers that are kept modular. This method provides a clear separation between business logic (Model), user interface (View) and control process (Controller) so you can easily update or expand one layer without affecting the other layers. The MVC design pattern has been confirmed as a successful design pattern used in the development of structured and maintainable software [10][11][12][13][20].

Although not the focus of this study, the MVC pattern can ease the adaptation of the system to handle various SQL dialects, such as ANSI SQL, T-SQL, and PL/SQL, by isolating the SQL processing logic in the Model layer. It can also improve compatibility with various database management systems (DBMS) by allowing the Controller to facilitate communication between the Model and the DBMS, thus keeping the codebase clean and modular. In addition, MVC supports integration with external systems such as Learning Management Systems (LMS) by allowing the Controller to manage data exchange, while the View layer adjusts user interactions. These examples illustrate the importance of ease of development and maintenance of SQL exam systems. A modular approach such as MVC ensures the functionality and adaptability of the system in the long term, which ultimately supports the ever-evolving educational landscape that requires reliable, scalable, and easy-to-maintain tools. The implementation of this design pattern is also in line with the modern software development trend that emphasizes modularity and separation of concerns [19][20]. Although various approaches have been proposed for automated SQL exam systems, this study focuses on the application of the MVC design pattern as a solution to improve software maintainability and extensibility. Thus, this research aims to make a significant contribution to the development of a more adaptive and sustainable system for SQL learning.

5. Conclusion

This research has developed a web-based online SQL exam system that implements the Model-View-Controller (MVC) design pattern. SQLExaminer is a simple SQL exam system that can help students as examinees and instructors who need a simple but efficient assessment system. This is the most effective way to handle this problem automatically. Technically, this feature makes it easier for instructors to check answers manually and provide fast and accurate feedback to students. Although the current system performs the above query checking function (SELECT) quite well, there is still room for optimization. In the next stage of evolution, it would be useful to add an analysis of the existing exam results in the system to visualize students' understanding of SQL. In addition, connectivity to other platforms like LMS will expand the application and ultimately the data obtained from students. By adding automatic grading capabilities to other types of SQL statements such as DDL and DML, it will also increase the usability of the system. Time series visualization, ports to Windows, and enhanced integration serve to make the web-based SQL checking system an increasingly powerful tool for learning and evaluation in information technology.

Acknowledgment
The authors would like to express their appreciation to the directorate and management of Politeknik Negeri Malang for their continuous support. The Center for Research and Community Service (P3M) at Politeknik Negeri Malang is also acknowledged for their crucial role in facilitating this research project. This research was funded by an internal research grant, No. SP DIPA–023.18.2.677606/2023. Furthermore, the authors extend their gratitude to the colleagues and staff of the Information Technology Department for their collective contributions, including insightful feedback and technical guidance, which were essential to the success of this project.

References

[1] Revathi, D. A. R., Bhuvaneswaran, R., Arshath, K. M., Prakash, U. H., & Aravinth, M. L. (2020). OVERVIEW OF DATABASE EVOLUTION AND ELEVATION. *International Journal of Advance Research and Innovative Ideas in Education*, *6*, 422–428.

[2] Ambasana, J., Sahasrabudhe, S., & Iyer, S. (2023, December). SQL-Wordle: Gamification of SQL Programming Exercises. In *Proceedings of the ACM Conference on Global Computing Education Vol 2* (pp. 190-190). https://doi.org/10.1145/3617650.3624949

[3] Teate, R. M. P. (2021). The SELECT statement. In *SQL for data scientists: A beginner's guide for building datasets for analysis* (pp. 15-32). John Wiley & Sons. https://doi.org/10.1002/9781119669388.ch2

[4] Kidman, G., & Chang, C. H. (2022). Assessment and evaluation in geographical and environmental education. *International Research in Geographical and Environmental Education*, *31*(3), 169-171. https://doi.org/10.1080/10382046.2022.2105499

[5] Cladera, M. (2021). An application of importance-performance analysis to students' evaluation of teaching. *Educational Assessment, Evaluation and Accountability*, *33*(4), 701-715. https://doi.org/10.1007/s11092-020-09338-4

[6] Fabijanić, M., Đambić, G., & Fulanović, B. (2020, September). A novel system for automatic, configurable and partial assessment of student SQL queries. In *2020 43rd International Convention on Information, Communication and Electronic Technology (MIPRO)* (pp. 832-837). IEEE. https://doi.org/10.23919/MIPRO48935.2020.9245264

[7] Nayak, S., Agarwal, R., & Khatri, S. K. (2022, January). Review of Automated Assessment Tools for grading student SQL queries. In *2022 International Conference on Computer Communication and Informatics (ICCCI)* (pp. 1-4). IEEE. https://doi.org/10.1109/ICCCI54379.2022.9740799

[8] Butler-Henderson, K., & Crawford, J. (2020). A systematic review of online examinations: A pedagogical innovation for scalable authentication and integrity. *Computers & Education*, *159*, 104024. https://doi.org/10.1016/j.compedu.2020.104024

[9] Du-xiang, W. (2009). Using ASP and SQL Server 2000 the Development of Online Examination System. *Computer Knowledge and Technology*.

[10] Asghar, M. Z., Alam, K. A., & Javed, S. (2019, December). Software design patterns recommendation: a systematic literature review. In *2019 international conference on Frontiers of Information Technology (FIT)* (pp. 167-1675). IEEE. https://doi.org/10.1109/FIT47737.2019.00040

[11] Mayvan, B. B., Rasoolzadegan, A., & Yazdi, Z. G. (2017). The state of the art on design patterns: A systematic mapping of the literature. *Journal of Systems and Software*, *125*, 93-118. https://doi.org/10.1016/j.jss.2016.11.030

[12] Ramírez-Noriega, A., Martínez-Ramírez, Y., Jiménez, S., Soto-Vega, J., & Figueroa-Pérez, J. F. (2022). inDev: A software to generate an MVC architecture based on the ER model. *Computer Applications in Engineering Education*, *30*(1), 259-274. https://doi.org/10.1002/cae.22455

[13] Wongtanuwat, W., & Senivongse, T. (2020, July). Detection of Violation of MVVM Design Pattern in Objective-C Programs. In *Proceedings of the 8th International Conference on Computer and Communications Management* (pp. 54-58). https://doi.org/10.1145/3411174.3411193

[14] Freeman, A., & Freeman, A. (2015). The singleton pattern. *Pro Design Patterns in Swift*, 113-136. https://doi.org/10.1007/978-1-4842-0394-1_6

[15] McDonough, J. E., & McDonough, J. E. (2017). Factory Design Patterns. *Object-Oriented Design with ABAP: A Practical Approach*, 173-190. https://doi.org/10.1007/978-1-4842-2838-8_14

[16] Freeman, A., & Freeman, A. (2015). The Observer Pattern. *Pro Design Patterns in Swift*, 447-472. https://doi.org/10.1007/978-1-4842-0394-1_22

[17] McDonough, J. E., & McDonough, J. E. (2017). Observer Design Pattern. *Object-Oriented Design with ABAP: A Practical Approach*, 155-171. https://doi.org/10.1007/978-1-4842-2838-8_13

[18] Zhou, K. (2020). Application Method of Struts Framework Based on MVC Design Pattern. https://doi.org/10.25236/ictmic.2020.133

[19] Haitao, L. (2011). Application of MVC Design Pattern in JSP Development. *Computer Programming Skills & Maintenance*.

[20] Kumar, A., Pandey, S. K., Prakash, S., Singh, K. U., Singh, T., & Kumar, G. (2023, April). Enhancing Web Application Efficiency: Exploring Modern Design Patterns within the MVC Framework. In *2023 International Conference on Computational Intelligence and Sustainable Engineering Solutions (CISES)* (pp. 43-48). IEEE. https://doi.org/10.1109/CISES58720.2023.10183582

[21] Wanjiru, B., van Bommel, P., & Hiemstra, D. (2022). Dynamic and Partial Grading of SQL Queries. *Journal of Emerging Network and Research Systems*. [Online]. Available: https://www.jenrs.com/publications/JENRS_0308001.pdf

[22] Kovacic, Z. J., & Green, J. (2012). Automatic grading of spreadsheet and database skills. *Journal of information Technology Education. Innovations in practice*, *11*, 53.

[23] Al-Salmi, A. (2019). *Semi-automatic assessment of basic SQL statements* (Doctoral dissertation, Loughborough University). Available: https://core.ac.uk/download/pdf/288353187.pdf

[24] Kleiner, C., & Heine, F. (2024). Enhancing Feedback Generation for Autograded SQL Statements to Improve Student Learning. In *Proceedings of the 2024 on Innovation and Technology in Computer Science Education V. 1* (pp. 248-254). Available: https://dl.acm.org/doi/abs/10.1145/3649217.3653579

[25] Kleerekoper, A., & Schofield, A. (2018, July). SQL tester: an online SQL assessment tool and its impact. In *Proceedings of the 23rd annual ACM conference on innovation and technology in computer science education* (pp. 87-92). Available: https://e-space.mmu.ac.uk/620119/1/SQL

[26] Rushby, J. (1993). *Formal methods and the certification of critical systems* (Vol. 37). SRI International, Computer Science Laboratory. Available: http://www.csl.sri.com/~rushby/papers/csl-93-7.pdf

[27] Yang, Y., Xia, X., Lo, D., & Grundy, J. (2022). A survey on deep learning for software engineering. *ACM Computing Surveys (CSUR)*, *54*(10s), 1-73. https://doi.org/10.1145/3505243.