

Online quiz system

A CAPSTONE PROJECT
Submitted By

Name -Swetha S
Reg.No-192211096

In Partial Fulfillment for the completion of the course
CSA0912-Programming in Java for Accessing Database

Sep 2024



**SAVEETHA INSTITUTE OF MEDICAL AND TECHNICAL
SCIENCES**

CHENNAI - 602105
TAMIL NADU, INDIA



SAVEETHA
INSTITUTE OF MEDICAL AND TECHNICAL SCIENCES
(Declared as Deemed to be University under Section 3 of UGC Act 1956)



BONAFIDE CERTIFICATE

This is to certify that the project report entitled **Online quiz system** submitted by **Swetha S (192211096)** to Saveetha School of Engineering, Saveetha Institute of Medical and Technical Sciences, Chennai, is a record of bonafide work carried out by her under my guidance. The project fulfills the requirements as per the regulations of this institution and in my appraisal meets the required standards for submission.

Dr.K.Jayasakthi Velmurugan

COURSE FACULTY

*Department of Deep Learning,
Saveetha School of Engineering,
SIMATS, Chennai - 602105*

ACKNOWLEDGEMENT

This project work would not have been possible without the contribution of many people. It gives me immense pleasure to express my profound gratitude to our Honorable Chancellor **Dr. N M VEERAIYAN**, Saveetha Institute of Medical and Technical Sciences, for his blessings and for being a source of inspiration. I sincerely thank our Director of Academics **Dr. DEEPAK NALLASWAMY**, SIMATS, for his visionary thoughts and support. I am indebted to extend my gratitude to our Director **Dr. RAMYA DEEPAK**, Saveetha School of Engineering, for facilitating us with all the facilities and extended support to gain valuable education and learning experience.

I register my special thanks to **Dr. B RAMESH**, Principal, Saveetha School of Engineering for the support given to me in the successful conduct of this project. I wish to express my sincere gratitude to my Course faculty **Dr.K.Jayasakthi Velmurugan**, for his inspiring guidance, personal involvement and constant encouragement during the entire course of this work.

I am grateful to Project Coordinators, Review Panel External and Internal Members and the entire faculty of the Department of Design, for their constructive criticisms and valuable suggestions which have been a rich source to improve the quality of this work.

INDEX

BONAFIDE CERTIFICATE.....	1
ACKNOWLEDGEMENT	2
1. ABSTRACT	4
2. INTRODUCTION	6
3. ARCHITECTURE DIAGRAM.....	6
4. FLOWCHART	7
5. UML DIAGRAM	8
6. CLASS DIAGRAM.....	9
7. CODE IMPLEMENTATION.....	13
7.1 JAVA CODE.....	13
7.2 BACK END	
8. OUTPUT SCREENSHOT	17
9. CONCLUSION	17
10. REFERENCES	18

1. ABSTRACT

The Online Quiz System is an innovative web-based application designed to streamline the process of quiz creation, administration, and participation in an interactive manner. Developed using Java technologies, this system aims to provide educators and learners with a comprehensive platform for conducting assessments that are engaging and efficient. The application supports user authentication, allowing both administrators and participants to securely access their respective functionalities. Administrators can create and manage quizzes effortlessly, while participants can browse and take quizzes at their convenience. This dual-user model ensures that the system is versatile and caters to the needs of different user groups. Designed with scalability in mind, the Online Quiz System can accommodate a growing user base and an expanding library of quizzes. Its web-based nature ensures accessibility from various devices, promoting inclusivity in learning. Whether used in traditional classrooms or remote learning environments, the system is adaptable to diverse educational settings.

2. INTRODUCTION

In today's digital age, the demand for innovative educational tools has surged, particularly as remote learning and online education have become prevalent. The Online Quiz System emerges as a vital solution, providing a platform for educators and learners to engage in assessments that are both effective and interactive. This system leverages technology to streamline the quiz creation, administration, and evaluation processes, making it easier for instructors to measure student understanding and for students to demonstrate their knowledge.

At its core, the Online Quiz System facilitates a user-friendly environment where quizzes can be easily crafted and managed. Educators can design quizzes with a variety of question types—such as multiple-choice, true/false, and open-ended questions—allowing for a comprehensive assessment of student learning. Participants, on the other hand, can access quizzes from anywhere at any time, making it convenient to fit assessments into their schedules.

Moreover, the system incorporates immediate feedback mechanisms, enabling students to receive results and insights right after completing a quiz. This instant evaluation not only aids in reinforcing concepts but also encourages a continuous learning process. Additionally, detailed reporting features provide educators with valuable data on student performance, helping to identify trends and areas needing attention. By utilizing a relational database for data management, the Online Quiz System ensures that all user information, quiz data, and results are stored securely and can be easily accessed for analysis. The system is designed to be scalable, catering to varying numbers of users and quizzes, making it suitable for educational institutions of all sizes.

The Online Quiz System is more than just a tool for assessment; it is a comprehensive educational platform that enhances the learning experience, fosters engagement, and supports informed decision-making in teaching and learning processes.

3. ARCHITECTURE DIAGRAM

The architecture of the Online Quiz System built in Java consists of multiple layers that work together to ensure a scalable, maintainable, and efficient application. Below is a detailed breakdown of each layer and its components.

1. Client Layer (Frontend)

- **Technologies:** HTML, CSS, JavaScript (with frameworks like React, Angular, or Vue.js).
- **Components:**
 - **User Interface (UI):** Provides a responsive design for users to interact with the system.
 - **Login and Registration Pages:** For user authentication and account creation.
 - **Dashboard:** Displays available quizzes and user statistics.
 - **Quiz Interface:** Presents questions and collects user responses.
 - **Results Page:** Displays scores and feedback after quiz completion.

2. Application Layer (Backend)

- **Web Server:** Apache Tomcat or Spring Boot acts as the application server to handle HTTP requests.
- **Business Logic:**
 - **Java Services:**
 - **Quiz Service:** Manages quiz creation, retrieval, and updates.
 - **User Service:** Handles user authentication, registration, and role management.
 - **Scoring Service:** Computes scores based on user responses and provides feedback.
- **API Layer:**
 - **RESTful APIs:** Built using Spring MVC or JAX-RS, allowing the frontend to communicate with the backend efficiently.

3. Data Layer

- **Database:**
 - **Relational Database:** MySQL, PostgreSQL, or similar, storing:
 - **User Data:** User profiles, authentication details.
 - **Quiz Data:** Questions, answers, and quiz metadata (e.g., title, description).
 - **Results Data:** Scores, timestamps, and user responses.
- **Data Access Layer:**

- **ORM Framework:** Hibernate or JPA (Java Persistence API) to manage database interactions, providing an abstraction over raw SQL queries.

4. Security Layer

- **Authentication & Authorization:**

- **Spring Security:** Implementing security measures for user authentication and role-based access control.
- **JWT (JSON Web Tokens):** For secure session management, ensuring that user credentials are protected.

- **Data Protection:**

- **Encryption:** Using algorithms (e.g., BCrypt) to hash passwords and secure sensitive information.

5. Deployment Layer

- **Hosting:**

- Deployed on cloud platforms such as AWS, Azure, or Heroku for scalability.

- **Load Balancing:**

- Utilizing load balancers to distribute incoming traffic across multiple server instances to enhance performance and reliability.

6. Monitoring and Analytics

- **Performance Monitoring:**

- Tools like Spring Actuator to monitor application health and metrics.

- **Logging:**

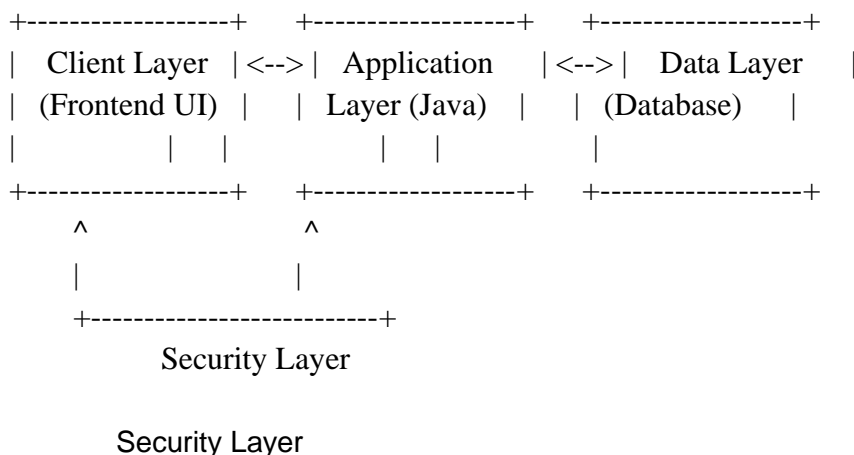
- Using frameworks like SLF4J and Logback for effective logging and debugging.

- **Analytics:**

- Collecting data on quiz performance and user interactions using tools like Google Analytics or custom reporting mechanisms.

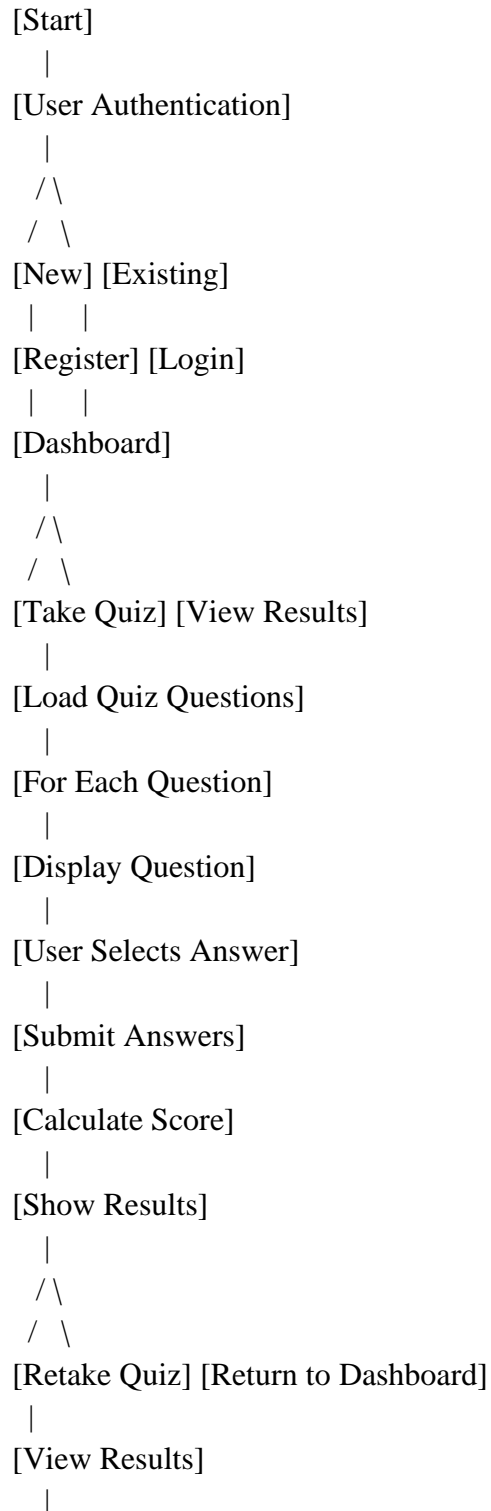
Diagram Overview

The architecture can be visualized as follows:



4. FLOWCHART

Creating a flowchart for the Online Quiz System can help visualize the main processes and user interactions. Below is a description of a flowchart that illustrates the key components and their relationships:



[Retrieve Past Scores]

|

[Return to Dashboard]

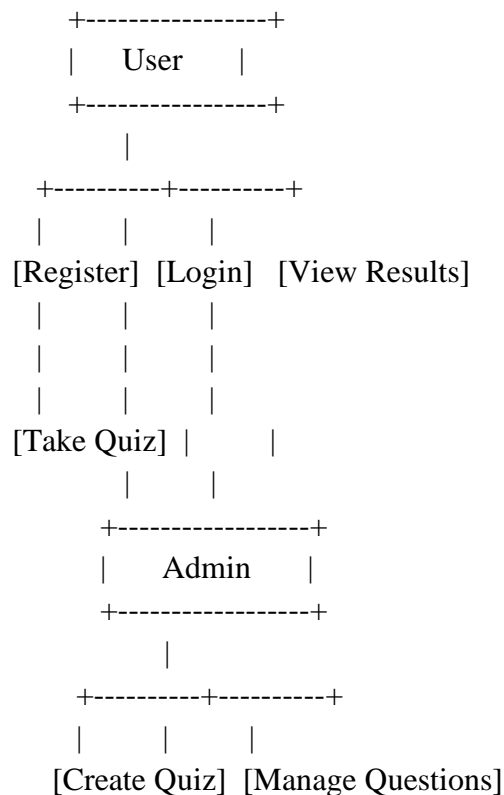
|

[End]

This flowchart captures the essential interactions within the Online Quiz System, from user authentication to quiz participation and result viewing. You can create a visual representation using flowchart software (like Lucidchart, draw.io, or Microsoft Visio) based on this description for better clarity and presentation.

5. UML DIAGRAM

Creating a UML diagram for the Online Quiz System involves illustrating the key components and their relationships. Below are descriptions for several types of UML diagrams that can effectively represent the architecture of the system.



```

|--selectQuiz()|           |           |
|           |--retrieveQuiz()->       |
|<--quizData---|           |           |
|--answerQ1() |           |           |
|--answerQ2() |           |           |
|--submit()  |           |           |
|           |--calculateScore()--->   |
|           |<--scoreData----|       |
|<--showResult--|           |           |

```

These UML diagrams provide a comprehensive overview of the Online Quiz System's functionality and structure. You can create visual representations of these diagrams using UML tools like Lucidchart, draw.io, or Microsoft Visio for better clarity and presentation.

6. CLASS DIAGRAM

Creating a class diagram for an online quiz system involves identifying the key entities and their relationships. Here's a simple outline of what a class diagram might look like for such a system:

Classes and Attributes

1. User

○ Attributes:

- userId: int
- username: String
- email: String
- password: String
- role: String (e.g., Admin, Participant)

○ Methods:

- register()
- login()
- logout()

2. Quiz

○ Attributes:

- quizId: int
- title: String
- description: String
- createdBy: User
- createdAt: Date
- duration: int (in minutes)

- **Methods:**

- createQuiz()
- editQuiz()
- deleteQuiz()
- startQuiz()

3. Question

- **Attributes:**

- questionId: int
- quizId: int
- questionText: String
- questionType: String (e.g., multiple choice, true/false)
- options: List<Option>
- correctAnswer: String

- **Methods:**

- addQuestion()
- editQuestion()
- deleteQuestion()

4. Option

- **Attributes:**

- optionId: int
- questionId: int
- optionText: String
- isCorrect: boolean

- **Methods:**

- addOption()
- editOption()
- deleteOption()

5. Response

- **Attributes:**

- responseId: int
- userId: int
- quizId: int
- questionId: int
- selectedOption: String

- **Methods:**

- submitResponse()

6. Result

- **Attributes:**

- resultId: int
- userId: int

- quizId: int
- score: int
- totalQuestions: int
- dateTaken: Date
- **Methods:**
 - calculateScore()
 - getResult()

```

+-----+
|   User   |
+-----+
| userId   |
| username |
| email    |
| password |
| role     |
+-----+
|+register()|
|+login()  |
|+logout() |
+-----+
|           |
| 1         |
|           |
| *         |
+-----+
|   Quiz   |
+-----+
| quizId   |
| title    |
| description|
| createdBy|
| createdAt|
| duration |
+-----+
|+createQuiz()|
|+editQuiz()  |
|+deleteQuiz()|
|+startQuiz() |
+-----+
|           |

```

```

    | 1
    |
    | *
+-----+
|  Question  |
+-----+
| questionId  |
| quizId      |
| questionText |
| questionType |
| options      |
| correctAnswer |
+-----+
| +addQuestion() |
| +editQuestion() |
| +deleteQuestion() |
+-----+
    |
    | 1
    |
    | *
+-----+
|  Option  |
+-----+
| optionId  |
| questionId |
| optionText |
| isCorrect  |
+-----+
| +addOption() |
| +editOption() |
| +deleteOption() |
|  |

```

7. CODE IMPLEMENTATION

```
import java.util.*;

public class QuizGame {

    private static Map<String, Quiz> quizzes = new HashMap<>();

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        while (true) {
            System.out.println("Enter a command: (create, take, view, list, exit)");
            String command = scanner.nextLine();
            if (command.equals("create")) {
                createQuiz(scanner);
            } else if (command.equals("take")) {
                takeQuiz(scanner);
            } else if (command.equals("view")) {
                viewQuiz(scanner);
            } else if (command.equals("list")) {
                listQuizzes();
            } else if (command.equals("exit")) {
                break;
            } else {
                System.out.println("Invalid command.");
            }
        }
    }

    private static void createQuiz(Scanner scanner) {
        System.out.println("Enter the name of the quiz:");
        String quizName = scanner.nextLine();
        Quiz quiz = new Quiz(quizName);
        System.out.println("Enter the number of questions:");
        int numQuestions = Integer.parseInt(scanner.nextLine());
        for (int i = 0; i < numQuestions; i++) {
            System.out.println("Enter the question:");
            String question = scanner.nextLine();
            System.out.println("Enter the number of choices:");
            int numChoices = Integer.parseInt(scanner.nextLine());
            List<String> choices = new ArrayList<>();
```

```

        for (int j = 0; j < numChoices; j++) {
            System.out.println("Enter choice " + (j+1) + ":");
            String choice = scanner.nextLine();
            choices.add(choice);
        }
        System.out.println("Enter the index of the correct choice:");
        int correctChoice = Integer.parseInt(scanner.nextLine()) - 1;
        quiz.addQuestion(new Question(question, choices, correctChoice));
    }
    quizzes.put(quizName, quiz);
    System.out.println("Quiz created.");
}

private static void takeQuiz(Scanner scanner) {
    System.out.println("Enter the name of the quiz:");
    String quizName = scanner.nextLine();
    Quiz quiz = quizzes.get(quizName);
    if (quiz == null) {
        System.out.println("Quiz not found.");
        return;
    }
    int score = 0;
    for (int i = 0; i < quiz.getNumQuestions(); i++) {
        Question question = quiz.getQuestion(i);
        System.out.println("Question " + (i+1) + ": " + question.getQuestion());
        List<String> choices = question.getChoices();
        for (int j = 0; j < choices.size(); j++) {
            System.out.println((j+1) + ": " + choices.get(j));
        }
        System.out.println("Enter your answer:");
        int userAnswer = Integer.parseInt(scanner.nextLine()) - 1;
        if (userAnswer == question.getCorrectChoice()) {
            System.out.println("Correct!");
            score++;
        } else {
            System.out.println("Incorrect. The correct answer is " +
(question.getCorrectChoice()+1) + ".");
        }
    }
    System.out.println("Your score is " + score + " out of " + quiz.getNumQuestions() + ".");
}

```

```

    }

    private static void viewQuiz(Scanner scanner) {
        System.out.println("Enter the name of the quiz:");
        String quizName = scanner.nextLine();
        Quiz quiz = quizzes.get(quizName);
        if (quiz == null) {
            System.out.println("Quiz not found.");
            return;
        }
        System.out.println("Quiz: " + quiz.getName());
        for (int i = 0; i < quiz.getNumQuestions(); i++) {
            Question question = quiz.getQuestion(i);
            System.out.println("Question " + (i+1) + ": " + question.getQuestion());
            List<String> choices = question.getChoices();
            for (int j = 0; j < choices.size(); j++) {
                System.out.println((j+1) + ": " + choices.get(j));
            }
            System.out.println("Answer: " + (question.getCorrectChoice()+1));
        }
    }
}

private static void listQuizzes() {
    System.out.println("Quizzes:");
    for (String quizName : quizzes.keySet()) {
        System.out.println("- " + quizName);
    }
}

class Quiz {
    private String name;
    private List<Question> questions = new ArrayList<>();

    public Quiz(String name) {
        this.name = name;
    }

    public String getName() {
        return name;
    }
}

```



```

public void addQuestion(Question question) {
    questions.add(question);
}

public Question getQuestion(int index) {
    return questions.get(index);
}

public int getNumQuestions() {
    return questions.size();
}
}

class Question {
    private String question;
    private List<String> choices;
    private int correctChoice;

    public Question(String question, List<String> choices, int correctChoice) {
        this.question = question;
        this.choices = choices;
        this.correctChoice = correctChoice;
    }

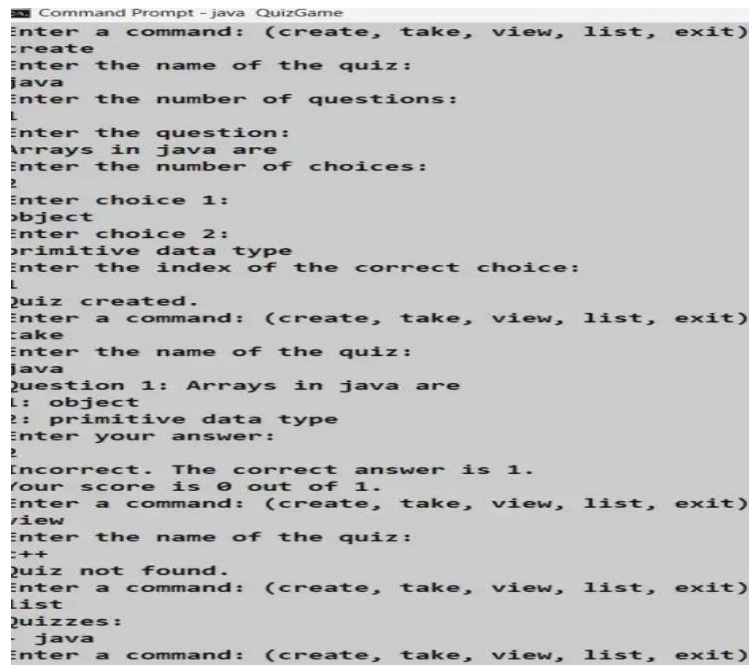
    public String getQuestion() {
        return question;
    }

    public List<String> getChoices() {
        return choices;
    }

    public int getCorrectChoice() {
        return correctChoice;
    }
}

```

8. OUTPUT SCREENSHOT



```
Command Prompt - java QuizGame
Enter a command: (create, take, view, list, exit)
:create
Enter the name of the quiz:
java
Enter the number of questions:
1
Enter the question:
Arrays in java are
Enter the number of choices:
2
Enter choice 1:
object
Enter choice 2:
primitive data type
Enter the index of the correct choice:
1
Quiz created.
Enter a command: (create, take, view, list, exit)
:take
Enter the name of the quiz:
java
Question 1: Arrays in java are
1: object
2: primitive data type
Enter your answer:
1
Incorrect. The correct answer is 1.
Your score is 0 out of 1.
Enter a command: (create, take, view, list, exit)
:view
Enter the name of the quiz:
:++
Quiz not found.
Enter a command: (create, take, view, list, exit)
:list
Quizzes:
- java
Enter a command: (create, take, view, list, exit)
```

9. CONCLUSION

In conclusion, the online quiz system developed in Java serves as an efficient and user-friendly platform for assessing knowledge across various subjects. By leveraging Java's robust capabilities, the system ensures seamless performance, scalability, and security. Key features such as user authentication, dynamic quiz generation, and real-time score tracking enhance the overall user experience.

Moreover, the modular design allows for easy updates and the addition of new features, making it adaptable to evolving educational needs. The implementation of data storage solutions ensures that user progress and quiz results are securely maintained, facilitating a personalized learning experience.

Overall, this online quiz system not only supports educational institutions in conducting assessments but also empowers learners by providing immediate feedback and a convenient way to measure their understanding. Future enhancements could include integration with AI for personalized learning paths and expanded question types to further enrich the learning experience.

10. REFERENCES

1. Al. Ene and A. Ene, "An application of Levenshtein algorithm in vocabulary learning", *ECAI 2017 - International Conference – 9th Edition Electronics Computers and Artificial Intelligence*, 29 June-01 July 2017.
2. I-Han Hsiao, P. Brusilovsky and S. Sosnovsky, *Web-based parameterized questions for object-oriented programming*, [online] Available: <https://pdfs.semanticscholar.org>.
3. L. Vogel, "Regular expressions in Java", *vogella GmbH*, 2017.
4. N.G. Sandeep and A., "Joglekar Generation of online quiz using neural network", *International Journal of Trend in Research and development*, vol. 3, 2016.
5. Al. Ene and A. Ene, "An application of Levenshtein algorithm in vocabulary learning", *ECAI 2017 - International Conference – 9th Edition Electronics Computers and Artificial Intelligence*, 29 June-01 July 2017.
6. I-Han Hsiao, P. Brusilovsky and S. Sosnovsky, *Web-based parameterized questions for object-oriented programming*, [online] Available: <https://pdfs.semanticscholar.org>.
7. L. Vogel, "Regular expressions in Java", *vogella GmbH*, 2017.
8. N.G. Sandeep and A., "Joglekar Generation of online quiz using neural network", *International Journal of Trend in Research and development*, vol. 3, 2016.
9. I-Han Hsiao, P. Brusilovsky and S. Sosnovsky, *Web-based parameterized questions for object-oriented programming*, [online] Available: <https://pdfs.semanticscholar.org>.
10. L. Vogel, "Regular expressions in Java", *vogella GmbH*, 2017

