Online Quiz System

Name	id
Zeyad Mohamed Hassan	2101919
Eman Essam Al-Dien Mahmoud	2101870
Marina essam Kamel	2102018
Eman Maeawad	2101873
Amr Esmaail Yousef	2101992

Introduction

In the digital age, online quizzes have become a popular tool for education, entertainment, and evaluation. To create an efficient and scalable online quiz system, we can employ various software design patterns and architectural principles. These patterns ensure modularity, maintainability, and adaptability of the system.

Key Design Patterns and Concepts in the System

1. Singleton Pattern
   The Singleton design pattern is used to manage resources that should only have a single instance throughout the application. For instance, the database connection or configuration manager in the online quiz system will leverage Singleton to ensure that only one instance is used, reducing redundancy and conserving resources.

2. Factory Pattern
   The Factory pattern is used to create quiz components dynamically based on user preferences or input. For example, different question types (multiple-choice, true/false, fill-in-the-blank) can be instantiated using a factory class, allowing for extensibility and reusability.

3. Model-View-Controller (MVC)
   The MVC architecture separates the system into three interconnected components:
   - Model: Handles the business logic, such as fetching quiz data, scoring, and user records.
   - View: Manages the presentation layer, displaying quiz questions, user progress, and results in an intuitive format.
   - Controller: Acts as the intermediary between the Model and View, processing user input and updating the system accordingly.

4. Session Management
   Effective session management ensures that user data, such as ongoing quiz progress and login status, is maintained securely across multiple requests. By implementing session handling, users can pause and resume quizzes, ensuring a seamless experience while safeguarding their data.

5. Template Method Pattern
   The Template Method design pattern provides a framework for defining the structure of quiz operations while allowing flexibility in certain steps. For example, the quiz flow (loading questions, validating answers, calculating scores, and displaying results) can be outlined in a base class, with specific behaviors customized in derived classes for different quiz types.

1. Singleton Pattern:

- QuizManager: Manages quiz operations such as creation, starting, and progress tracking.
- ScoreManager: Handles scoring and result management.

```java
public class QuizManager {
    private static QuizManager instance;

    // Private constructor to prevent instantiation
    private QuizManager() {}

    public static QuizManager getInstance() {
        if (instance == null) {
            synchronized (QuizManager.class) {
                if (instance == null) {
                    instance = new QuizManager();
                }
            }
        }
        return instance;
    }

    public void startQuiz() {
        System.out.println("Quiz started.");
    }

    public void trackProgress() {
        System.out.println("Tracking quiz progress.");
    }
}
```

```java
public class ScoreManager {
    private static ScoreManager instance;

    private ScoreManager() {}

    public static ScoreManager getInstance() {
        if (instance == null) {
            synchronized (ScoreManager.class) {
                if (instance == null) {
                    instance = new ScoreManager();
                }
            }
        }
        return instance;
    }

    public void addScore(String userId, int score) {
        System.out.println("Score added for user " + userId + ": " + score);
    }

    public void getScores(String userId) {
        System.out.println("Fetching scores for user " + userId);
    }
}
```

2.Factory Pattern:

- QuestionFactory: Creates different question types (Multiple Choice, True/False).
- UserFactory: Generates user roles (Admin, Student, Teacher) and assigns roles.

```java
abstract class Question {
    String questionText;
    public abstract void displayQuestion();
}
```

```java
class TrueFalseQuestion extends Question {
    public TrueFalseQuestion(String questionText) {
        this.questionText = questionText;
    }

    @Override
    public void displayQuestion() {
        System.out.println("True/False: " + questionText);
    }
}
```

```java
class MultipleChoiceQuestion extends Question {
    public MultipleChoiceQuestion(String questionText) {
        this.questionText = questionText;
    }

    @Override
    public void displayQuestion() {
        System.out.println("Multiple Choice: " + questionText);
    }
}
```

- UserFactory: Generates user roles (Admin, Student) and assigns roles.

```java
abstract class User {
    String name;
    String role;

    public abstract void displayRole();
}
```

```java
class Admin extends User {
    public Admin(String name) {
        this.name = name;
        this.role = "Admin";
    }

    @Override
    public void displayRole() {
        System.out.println(name + " is an Admin.");
    }
}
```

```java
class Student extends User {
    public Student(String name) {
        this.name = name;
        this.role = "Student";
    }

    @Override
    public void displayRole() {
        System.out.println(name + " is a Student.");
    }
}
```

```java
class UserFactory {
    public static User createUser(String role, String name) {
        switch (role.toLowerCase()) {
            case "admin":
                return new Admin(name);
            case "student":
                return new Student(name);
            default:
                throw new IllegalArgumentException("Unknown role: " + role);
        }
    }
}
```

3. MVC (Model-View-Controller)

- The Servlets (register.java, login.java, logout.java, etc.) act as controllers by handling HTTP requests, managing user sessions, and interfacing with the database.
- JSPs are used as the View layer (e.g., login.jsp, index.jsp, setmarks.jsp) to render the interface for users.
- Model is absent; database queries are embedded directly in the Servlets, mixing the Controller and Model layers. This violates the separation of concerns that MVC aims to achieve.

```java
import javax.servlet.http.HttpSession;

/**
 *
 * @author Ziad w al97bh
 */
@WebServlet(name = "register", urlPatterns = {"/register"})
public class register extends HttpServlet {

    /**
     * Processes requests for both HTTP <code>GET</code> and <code>POST</code>
     * methods.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException, ClassNotFoundException, SQLException {
        response.setContentType("text/html;charset=UTF-8");
        try (PrintWriter out = response.getWriter()) {...50 lines }
    }

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {...9 lines }

    /** Handles the HTTP <code>POST</code> method ...8 lines */
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {...9 lines }

    /** Returns a short description of the servlet ...5 lines */
    @Override
    public String getServletInfo() {...3 lines }// </editor-fold>

}
```

```java
 *
 * @author Ziad w al97bh
 */
@WebServlet(name = "login", urlPatterns = {"/login"})
public class login extends HttpServlet {

    /** Processes requests for both HTTP <code>GET</code> and <code>POST</code> ...9 lines */
    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException, ClassNotFoundException, SQLException {
        response.setContentType("text/html;charset=UTF-8");
        try (PrintWriter out = response.getWriter()) {...65 lines }
    }

    // <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the + sign on the left to edit the co
    /**
     * Handles the HTTP <code>GET</code> method.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        try {
            processRequest(request, response);
        } catch (ClassNotFoundException ex) {
            Logger.getLogger(login.class.getName()).log(Level.SEVERE, null, ex);
        } catch (SQLException ex) {
            Logger.getLogger(login.class.getName()).log(Level.SEVERE, null, ex);
        }
    }

    /** Handles the HTTP <code>POST</code> method ...8 lines */
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {...9 lines }

    /** Returns a short description of the servlet ...5 lines */
    @Override
    public String getServletInfo() {...3 lines }// </editor-fold>

}
```

```java
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

/**
 *
 * @author Ziad w al97bh
 */
@WebServlet(name = "logout", urlPatterns = {"/logout"})
public class logout extends HttpServlet {

    /**
     * Processes requests for both HTTP <code>GET</code> and <code>POST</code>
     * methods.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        try (PrintWriter out = response.getWriter()) {
            /* TODO output your page here. You may use following sample code. */
            out.println("<!DOCTYPE html>");
            out.println("<html>");
            out.println("<head>");
            out.println("<title>Servlet logout</title>");
            out.println("</head>");
            out.println("<body>");

            HttpSession hs = request.getSession(false);
            hs.removeAttribute("examsubject");
            hs.removeAttribute("email");
            hs.removeAttribute("username");
            hs.invalidate();
            response.sendRedirect("index.jsp");

            out.println("</body>");
            out.println("</html>");
        }
    }

    HttpServlet methods. Click on the + sign on the left to edit the code.
```

4. Session Management

- The project uses HttpSession to maintain user state across multiple requests, particularly for login (login.java) and logout (logout.java).

```java
                if(rs.next()){

                    name = rs.getString("username");
                    emailid = rs.getString("email");

                    HttpSession hs = request.getSession();
                    //hs.setAttribute("id", id);
                    hs.setAttribute("name", name);
                    hs.setAttribute("email", emailid);
                    RequestDispatcher rd = request.getRequestDispatcher("examscreen.jsp");
                    rd.forward(request, response);
                }
                else{
                    HttpSession hs = request.getSession();
                    RequestDispatcher d = request.getRequestDispatcher("login.jsp");
                    hs.setAttribute("err", "User Credentials Incorrect");
                    d.forward(request,response);
                    rs.close();
                    ps.close();
                }
            }
```

```java
        HttpSession hs = request.getSession(false);
        hs.removeAttribute("examsubject");
        hs.removeAttribute("email");
        hs.removeAttribute("username");
        hs.invalidate();
        response.sendRedirect("index.jsp");
```

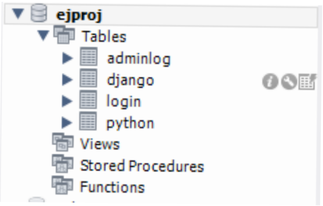
5. Template Method

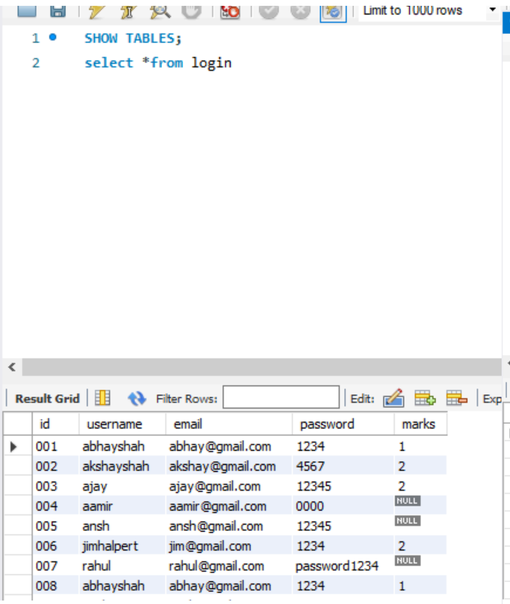
- The **HttpServlet** lifecycle methods (**doGet**, **doPost**) are overridden in Servlets, leveraging the Template Method pattern. This ensures that common functionality is handled by the framework while allowing specific behavior to be implemented.

```java
// <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the + sign on the left to edit the code.">
/**
 * Handles the HTTP <code>GET</code> method.
 *
 * @param request servlet request
 * @param response servlet response
 * @throws ServletException if a servlet-specific error occurs
 * @throws IOException if an I/O error occurs
 */
@Override
protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
    try {
        processRequest(request, response);
    } catch (ClassNotFoundException ex) {
        Logger.getLogger(login.class.getName()).log(Level.SEVERE, null, ex);
    } catch (SQLException ex) {
        Logger.getLogger(login.class.getName()).log(Level.SEVERE, null, ex);
    }
}
```

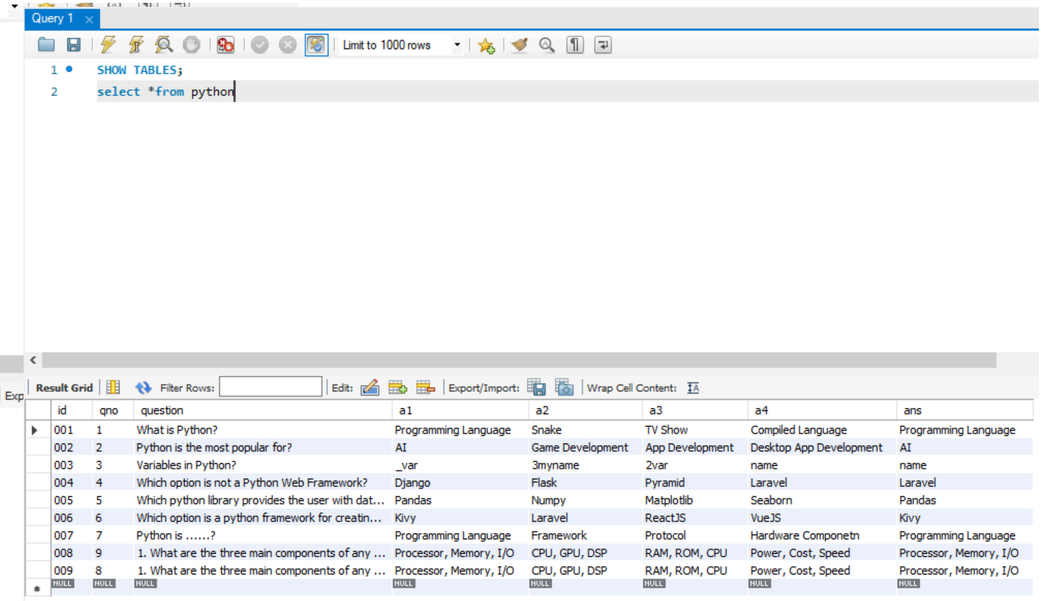
```java
protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
    try {
        processRequest(request, response);
    } catch (ClassNotFoundException ex) {
        Logger.getLogger(login.class.getName()).log(Level.SEVERE, null, ex);
    } catch (SQLException ex) {
        Logger.getLogger(login.class.getName()).log(Level.SEVERE, null, ex);
    }
}

/**
 * Returns a short description of the servlet.
 *
 * @return a String containing servlet description
 */
```

Connection mySql database

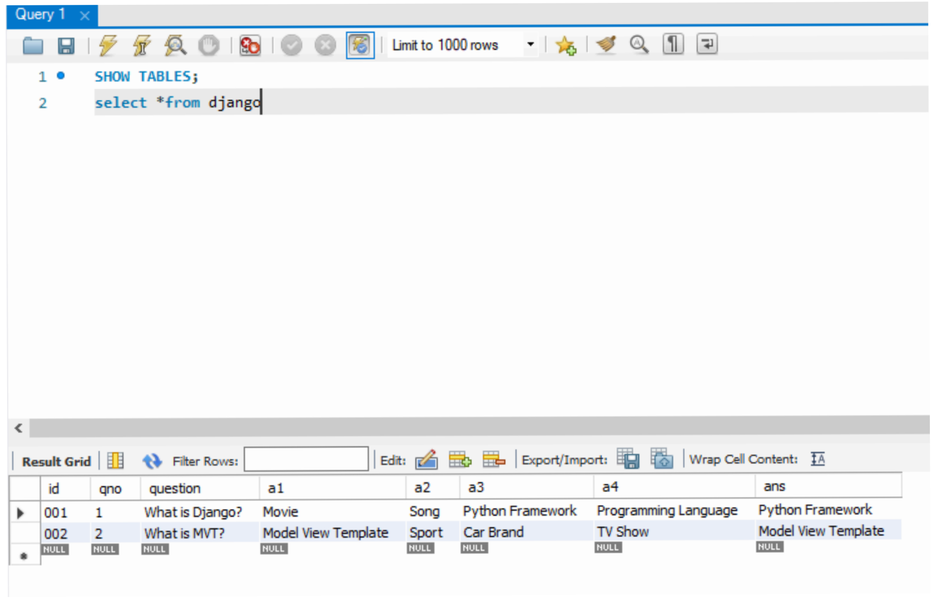




id	username	email	password	marks
001	abhayshah	abhay@gmail.com	1234	1
002	akshayshah	akshay@gmail.com	4567	2
003	ajay	ajay@gmail.com	12345	2
004	aamir	aamir@gmail.com	0000	NULL
005	ansh	ansh@gmail.com	12345	NULL
006	jimhalpert	jim@gmail.com	1234	2
007	rahul	rahul@gmail.com	password1234	NULL
008	abhayshah	abhay@gmail.com	1234	1



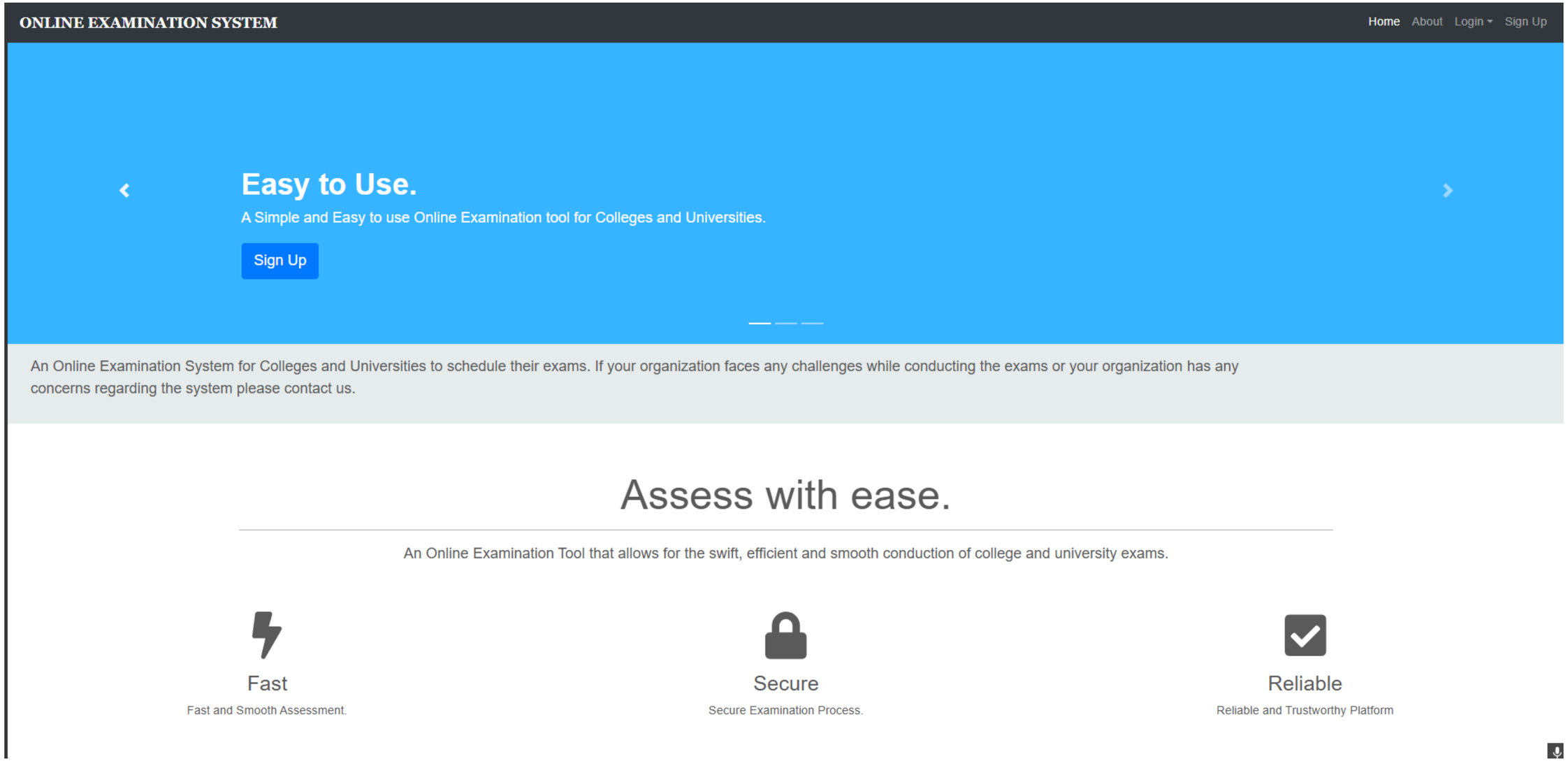
id	qno	question	a1	a2	a3	a4	ans
001	1	What is Python?	Programming Language	Snake	TV Show	Compiled Language	Programming Language
002	2	Python is the most popular for?	AI	Game Development	App Development	Desktop App Development	AI
003	3	Variables in Python?	_var	3myname	2var	name	name
004	4	Which option is not a Python Web Framework?	Django	Flask	Pyramid	Laravel	Laravel
005	5	Which python library provides the user with dat...	Pandas	Numpy	Matplotlib	Seaborn	Pandas
006	6	Which option is a python framework for creatin...	Kivy	Laravel	ReactJS	VueJS	Kivy
007	7	Python is .....?	Programming Language	Framework	Protocol	Hardware Componetn	Programming Language
008	9	1. What are the three main components of any ...	Processor, Memory, I/O	CPU, GPU, DSP	RAM, ROM, CPU	Power, Cost, Speed	Processor, Memory, I/O
009	8	1. What are the three main components of any ...	Processor, Memory, I/O	CPU, GPU, DSP	RAM, ROM, CPU	Power, Cost, Speed	Processor, Memory, I/O



id	qno	question	a1	a2	a3	a4	ans
001	1	What is Django?	Movie	Song	Python Framework	Programming Language	Python Framework
002	2	What is MVT?	Model View Template	Sport	Car Brand	TV Show	Model View Template

Design system

The first page the user sees is the Home page



Admin User

Administrator Sign In

wilsonrao@gmail.com

••••

Login

ONLINE EXAMINATION SYSTEM

Home Set Examination Check Marks Logout

Welcome Admin!

Welcome to OESI, an online examination system for colleges and universities to conduct their exams.

Please Select the Subject which you want to add questions to:

Python

Select Exam

Examination Subject to Edit:

Once you have selected an exam to edit click on "Set Examination" to either add questions to the examination or view the exam

Admin

OES provides Admin accounts which gives the user the privilege to create, manipulate or delete the exam

How does it Work?

OES makes use of JSP, Servlets, JSTL and a MySql database

Terms & Conditions

If your organization has any concerns regarding the system please read our Terms and Conditions.

ONLINE EXAMINATION SYSTEM

Home Set Examination Check Marks Logout

Python Exam Set

Question_ID	Question Number	Question	Choice 1	Choice 2	Choice 3	Choice 4	Answer
1	1	What is Python?	Programming Language	Snake	TV Show	Compiled Language	Programming Language
2	2	Python is the most popular for?	AI	Game Development	App Development	Desktop App Development	AI
3	3	Variables in Python?	_var	3myname	2var	name	name
4	4	Which option is not a Python Web Framework?	Django	Flask	Pyramid	Laravel	Laravel
5	5	Which python library provides the user with dataframes?	Pandas	Numpy	Matplotlib	Seaborn	Pandas
6	6	Which option is a python framework for creating mobile applications?	Kivy	Laravel	ReactJS	VueJS	Kivy
7	7	Python is .....?	Programming Language	Framework	Protocol	Hardware Componetn	Programming Language
8	9	1. What are the three main components of any computing system?	Processor, Memory, I/O	CPU, GPU, DSP	RAM, ROM, CPU	Power, Cost, Speed	Processor, Memory, I/O
9	8	1. What are the three main components of any computing system?	Processor, Memory, I/O	CPU, GPU, DSP	RAM, ROM, CPU	Power, Cost, Speed	Processor, Memory, I/O

Student Marks

ID	Username	Email	Marks
1	abhayshah	abhay@gmail.com	1
2	akshayshah	akshay@gmail.com	2
3	ajay	ajay@gmail.com	2
4	aamir	aamir@gmail.com	
5	ansh	ansh@gmail.com	
6	jimhalpert	jim@gmail.com	2
7	rahul	rahul@gmail.com	
8	abhayshah	abhay@gmail.com	1
9	ziadooz	ziad@gmail.com	4

ONLINE EXAMINATION SYSTEM

Home Set Examination Check Marks Logout

Add Questions

Add Questions to Python Examination

Question Number

Question

Choice 1

Choice 2

Choice 3

Choice 4

Correct Choice

Add Question


Thanks!