Pimpri-Chinchwad Education Trust's
Pimpri-Chinchwad College of Engineering
(An Autonomous Institute)
Affiliated to Savitribai Phule Pune University (SPPU)
ISO 21001:2018 Certified by TUV

DEPARTMENT OF INFORMATION TECHNOLOGY

MINI PROJECT

COURSE: OBJECT-ORIENTED PROGRAMMING LABORATORY

PROJECT TITLE:

ADVANCED QUIZ PLATFORM WITH CLASS BASED SYSTEM.

GROUP MEMBERS:

Name	PRN
1. Om Rajendra Bharambe	124B1F001
2. Atharva Shankar Lokhande	124B1F009
3. Madhav Suresh Khobare	124B1F023

UNDER THE GUIDANCE OF:

Mrs. Tanuja Patankar

YEAR: SECOND YEAR

SEMESTER: III

DIVISION: A

ACADEMIC YEAR: 2025-2026

TABLE OF CONTENT

1. INTRODUCTION

The goal of this project is to develop a **Quiz Application** in Java using Object-Oriented Programming (OOP) principles and modern frameworks. The application allows users to **register**, **log in**, and take multiple-choice quizzes. The backend is built with Spring Boot (Java) while the frontend is a Java client application. The motivation behind this project is to apply OOP concepts in a real-world scenario and to gain practical experience with Java-based web technologies. Java's OOP approach "organizes code using classes and objects"[1], making it well-suited for modeling entities such as *User*, *Quiz*, and *Question*.

Key OOP concepts are applied throughout the design. For example, **encapsulation** is used by defining classes with private fields and public methods; **inheritance** can be applied (e.g., a specialized Admin class inheriting from a generic User class); **abstraction** is achieved via interfaces or abstract classes for service layers; and **polymorphism** allows different question types (e.g., multiple-choice, true/false) to be handled uniformly. These principles make the code modular and reusable. Tutorials note that OOP "simplifies software development" by providing such concepts[2][1]. In summary, the project idea is to create a structured, maintainable quiz system in Java that demonstrates core OOP ideas while using Spring Boot for rapid backend development and integration with a Java-based client interface.

2. OBJECTIVES

- **Implement core OOP principles.** Design the application using classes and objects, and demonstrate encapsulation, inheritance, and polymorphism in Java.
- **Build a modular, user-friendly application.** Apply OOP concepts to create a clean, intuitive interface for registration, login, and quiz-taking.
- **Ensure data persistence and security.** Use Java Persistence API (JPA) for database access and Spring Security for login/authentication.
- **Practice integration of technologies.** Use Spring Boot for the backend API and a standalone Java frontend. Employ libraries like Gson for JSON parsing and PDFBox for report generation.
- **Enhance problem-solving skills.** Solve a practical, real-world problem (online quiz) using object-oriented design and Java frameworks.

3. SYSTEM REQUIREMENTS

Hardware: The application should run on a typical PC or laptop with at least an Intel i3 processor and 4 GB of RAM[3]. More RAM (e.g. 8 GB) is recommended for development with IDEs and multiple services.

Software: A 64-bit operating system (Windows, Linux, or macOS) is required. Java Development Kit (JDK) version 17 or higher must be installed[4]. A Java IDE such as Eclipse, IntelliJ IDEA, or NetBeans is recommended[4]. Maven (or Gradle) is used for building the project.

The backend server uses **Spring Boot 3.5.7** (as specified in the POM)[5]. Dependencies include Spring Data JPA for ORM, Spring Web for REST APIs, Spring Security for authentication, and Spring Actuator for monitoring[6][7]. Database connectivity requires **MySQL** (Connector/J driver) for runtime operation[8]; the POM also includes **PostgreSQL** driver support[9]. For PDF functionality, **Apache PDFBox 3.0.6** is included on both server and client sides[10][11]. The Java client uses the **Gson 2.13.2** library for JSON processing[12]. A running instance of MySQL or PostgreSQL is required as per configuration. In summary:

- **Java/JDK:** Java SE 17 or above[4]
- **Build Tools:** Maven (including Spring Boot Maven plugin)[5]
- **Frameworks:** Spring Boot (v3.5.7) with Spring Data JPA, Spring Web, Spring Security[6][7]
- **Databases:** MySQL 8 (with mysql-connector-j)[8], optionally PostgreSQL (with driver)[9]
- **Libraries:** Apache PDFBox (v3.0.6)[10], Google Gson (v2.13.2)[12]
- **Tools:** IDE (Eclipse/IntelliJ IDEA/NetBeans), web browser (for API testing)
- **Hardware:** Intel i3 CPU or better, 4+ GB RAM[3]

4. IMPLEMENTATION

Github Link

Server: https://github.com/MadhavK3/mcq_server

Client: https://github.com/MadhavK3/mcq_client

API Endpoint Summary

- **Authentication Endpoints**

Method	**Endpoint**	**Purpose**	**Request Example / Params**	**Response Example (Success)**
POST	/api/auth/register	Register a new user	{ "username": "john_doe", "email": "john@example.com", "password": "Password123!", "fullName": "John Doe" }	{ "message": "User registered successfully.", "user": { "id": 1, "username": "john_doe", "email": "john@example.com", "role": "STUDENT" } }

POST	/api/auth/login	Login and get **JWT token**	{ "username": "john_doe", "password": "Password123!" }	{ "message": "Login successful.", "accessToken": "<JWT>", "tokenType": "Bearer" }
POST	/api/auth/forgot-password	Send password reset link	{ "email": "john@example.com" }	{ "message": "Password reset link sent to registered email." }
POST	/api/auth/reset-password	Reset password using token	{ "token": "reset_token_xyz", "newPassword": "NewPassword123!" }	{ "message": "Password reset successfully." }
POST	/api/auth/logout	Logout and invalidate token	Header: Authorization: Bearer <JWT>	{ "message": "Logged out successfully." }

- **User Management Endpoints**

Method	**Endpoint**	**Purpose**	**Request Example / Params**	**Response Example (Success)**
GET	/api/users	Get all registered users	–	[{ "username": "john_doe", "email": "john@example.com", "role": "STUDENT" }]

GET	/api/users/{username}	Get user details by username	–	{ "username": "john_doe", "email": "john@example.com", "joinedClassrooms": ["MATH101"] }

- **Classroom Management Endpoints**

Method	**Endpoint**	**Purpose**	**Request Example / Params**	**Response Example (Success)**
GET	/api/classrooms	Get list of classrooms (filterable)	/api/classrooms?name=Math&filter=active	[{ "code": "MATH101", "name": "Mathematics 101", "status": "ACTIVE" }]
POST	/api/classrooms	**Create** a new classroom	{ "name": "Physics 101", "description": "Introductory physics." }	{ "message": "Classroom created successfully.", "classroom": { "code": "PHYS101", "studentsCount": 0 } }
GET	/api/classrooms/{code}	Get classroom details	–	{ "code": "MATH101", "name": "Mathematics

				101", "teacher": "amy_watson" }
PUT	/api/classrooms/{code}	**Update** classroom details	{ "name": "Math 101 (Updated)", "description": "Updated syllabus covering calculus." }	{ "message": "Classroom updated successfully." }
DELETE	/api/classrooms/{code}	**Delete** a classroom	–	{ "message": "Classroom MATH101 deleted successfully." }
POST	/api/classrooms/{code}/join	**Join** a classroom	Header: Authorization: Bearer <JWT>	{ "message": "Successfully joined classroom MATH101." }
DELETE	/api/classrooms/{code}/leave	**Leave** a classroom	–	{ "message": "You have left the classroom MATH101." }

DELETE	/api/classrooms/{code}/remove/{studentUsername}	**Remove** student from class	–	{ "message": "Student john_doe removed from classroom MATH101." }

- **Test and Submission Endpoints**

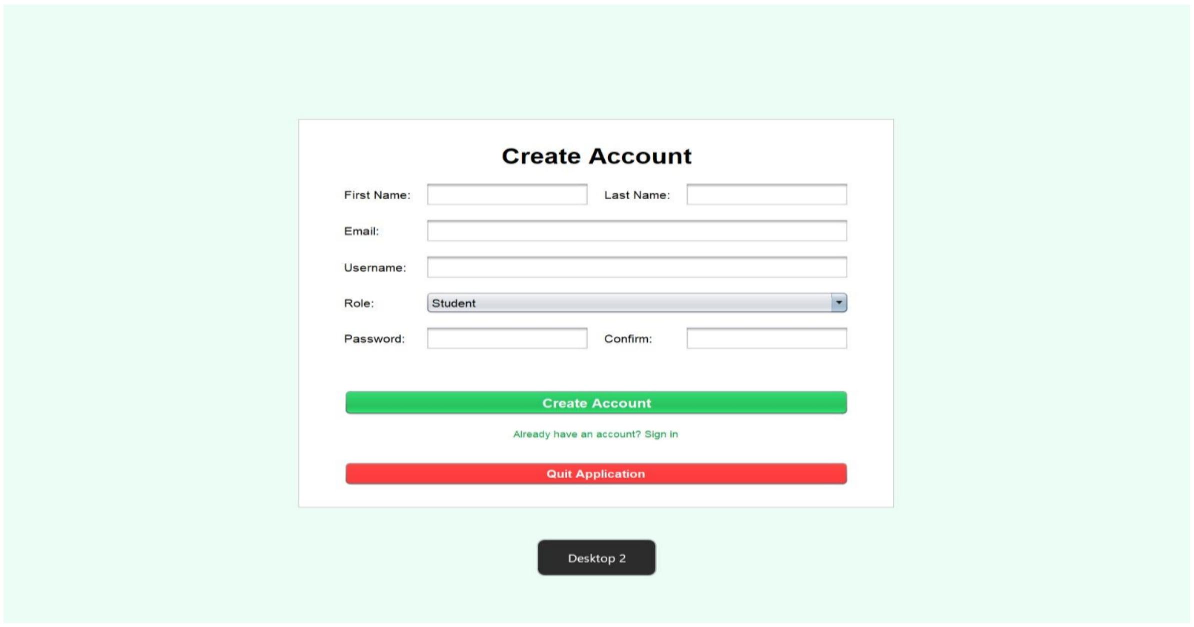
Method	**Endpoint**	**Purpose**	**Request Example / Params**	**Response Example (Success)**
POST	/api/classrooms/{classroomCode}/tests	**Create** a new test in classroom	Multipart: testname, pdfFile, correctAnswers	201 Created – Test created successfully
GET	/api/classrooms/student/active-test	Get **active test** for logged-in student	–	{ "classroomCode": "MATH101", "testName": "Algebra_Midterm", "status": "ACTIVE" }
GET	/api/classrooms/{classroomCode}/tests	Get all tests for a classroom	–	[{ "testname": "Algebra_Midterm", "status": "ACTIVE" }]

GET	/api/classrooms/{classroomCode}/tests/{testname}	Get specific test details	–	{ "testname": "Algebra_Midterm", "status": "ACTIVE", "questionCount": 10 }
GET	/api/classrooms/{classroomCode}/tests/{testname}/pdf	Download/view test PDF	–	Returns PDF file
DELETE	/api/classrooms/{classroomCode}/tests/{testname}	**Delete** a test and its submissions	–	204 No Content
POST	/api/classrooms/{classroomCode}/tests/{testname}/start	**Start test** and create submissions	–	{ "message": "Test started successfully. Submissions created for all students." }
POST	/api/classrooms/{classroomCode}/tests/{testname}/end	**End an active test**	–	{ "message": "Test ended successfully. Students can now submit answers." }
GET	/api/classrooms/{classroomCode}/tests/{testname}/submissions/my	Get current student's submission and result	Header: Authorization: Bearer <JWT>	{ "student": "john_doe", "score": 75, "status":

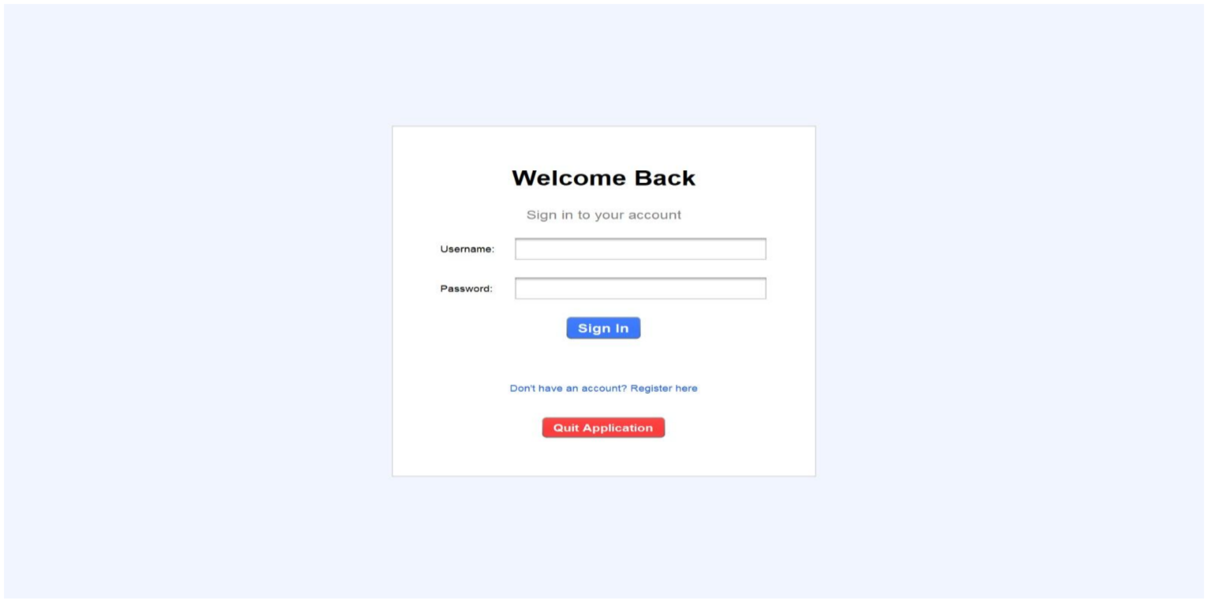
				"SUBMITTED" }
GET	/api/classrooms/{classroomCode}/tests/{testname}/submissions	View **all submissions** for a test	–	{ "correctAnswers": ["A","C"], "submissions": [{ "student": "john_doe", "score": 100 }] }
POST	/api/classrooms/{classroomCode}/tests/{testname}/submissions/update	**Update answers** while test is active	{ "userAnswers": ["A", "C", "B", "D"] }	{ "message": "Answers saved." }
POST	/api/classrooms/{classroomCode}/tests/{testname}/submissions/submit	**Submit final answers** after test ended	{ "userAnswers": ["A", "C", "B", "D"] }	{ "message": "Answers submitted successfully." }

5. OUTPUT PANELS

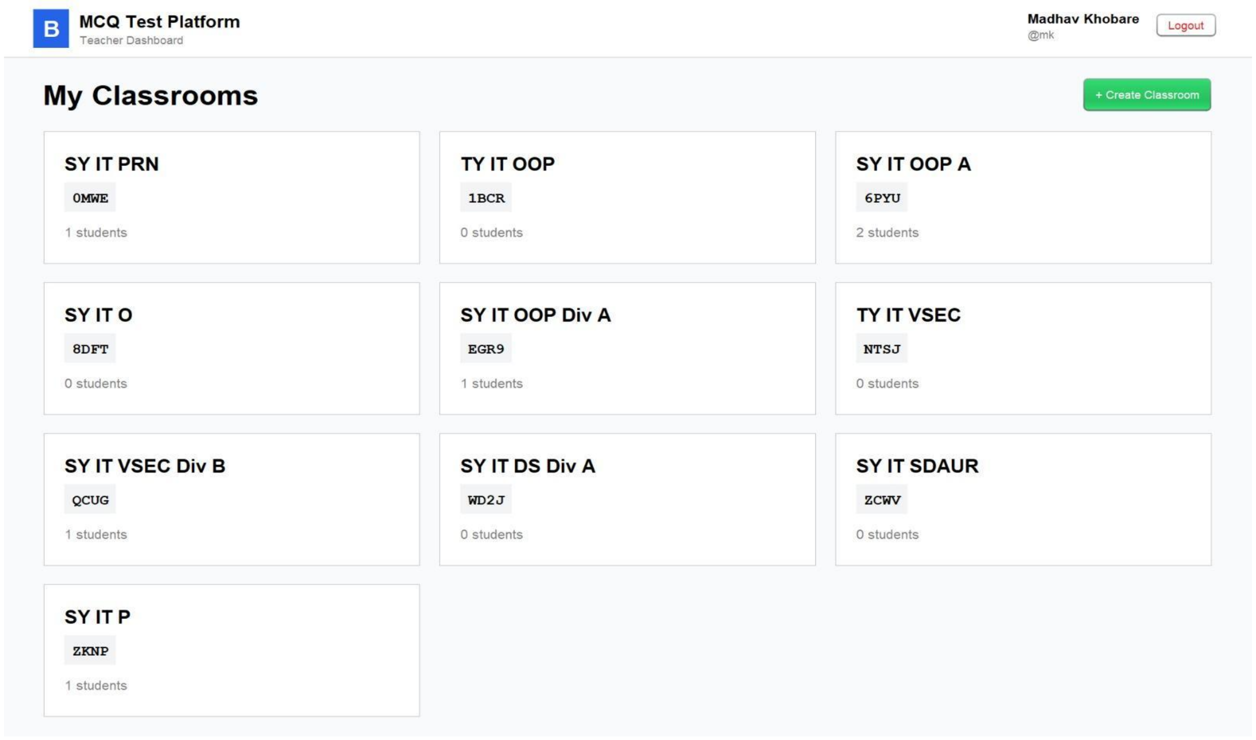
- LOGIN SCREEN



- REGISTER SCREEN



- TEACHER CLASSROOM PANEL



My Classrooms

SY IT PRN
0MWE
1 students

TY IT OOP
1BCR
0 students

SY IT OOP A
6PYU
2 students

SY IT O
8DFT
0 students

SY IT OOP Div A
EGR9
1 students

TY IT VSEC
NTSJ
0 students

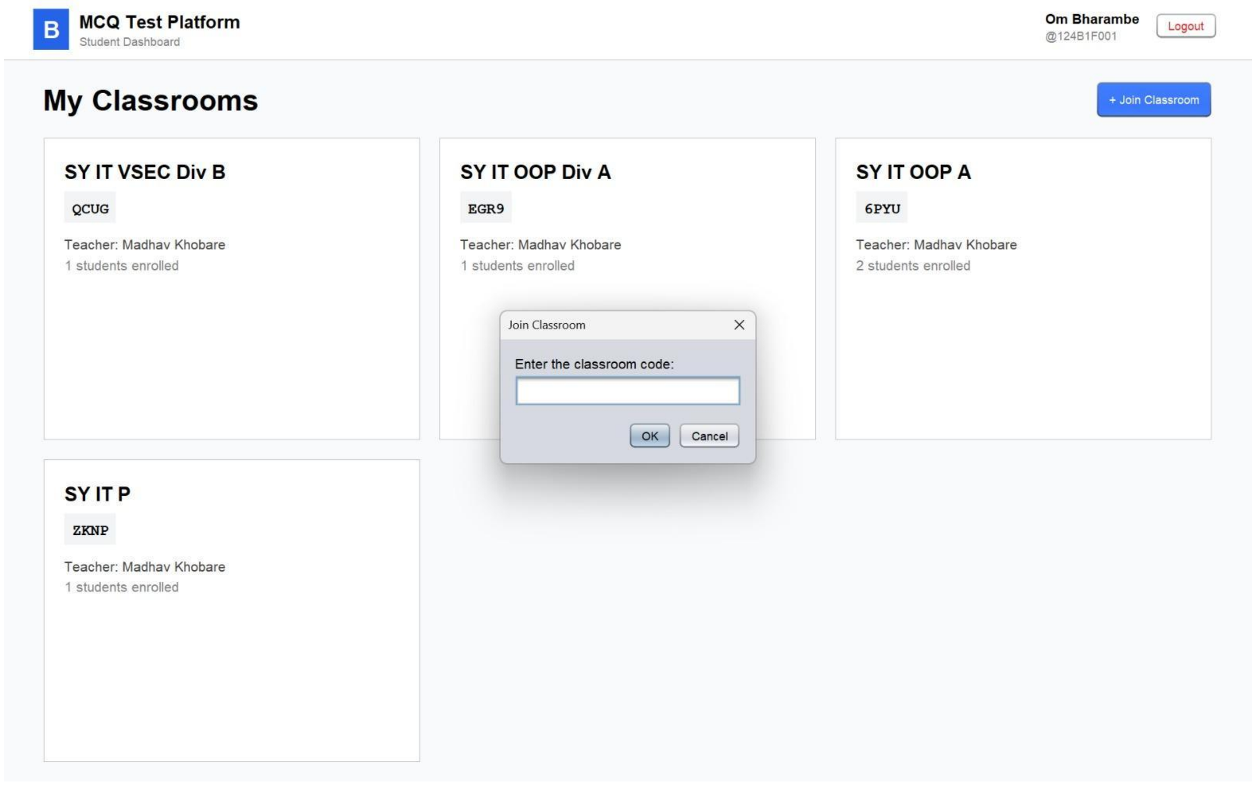
SY IT VSEC Div B
QCUG
1 students

SY IT DS Div A
WD2J
0 students

SY IT SDAUR
ZCWV
0 students

SY IT P
ZKNP
1 students

- STUDENT DASHBOARD TO JOIN THE CLASSROOM



My Classrooms

SY IT VSEC Div B
QCUG
Teacher: Madhav Khobare
1 students enrolled

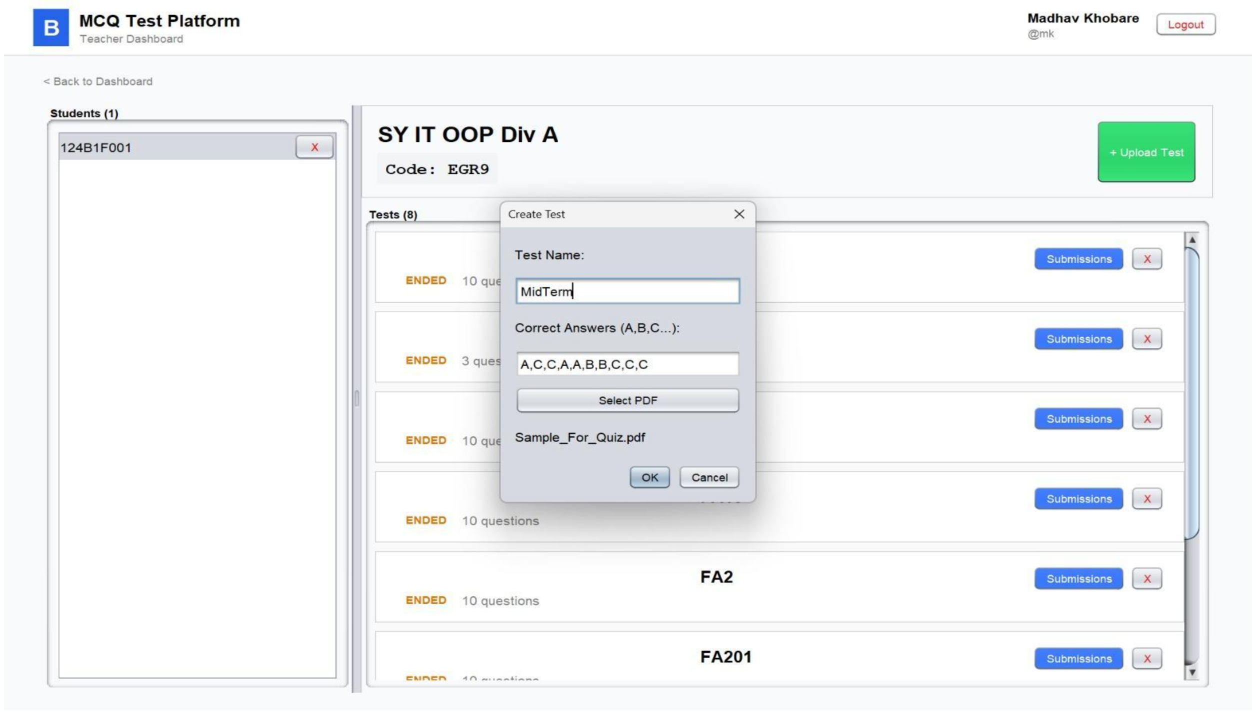
SY IT OOP Div A
EGR9
Teacher: Madhav Khobare
1 students enrolled

SY IT OOP A
6PYU
Teacher: Madhav Khobare
2 students enrolled

SY IT P
ZKNP
Teacher: Madhav Khobare
1 students enrolled

Enter the classroom code:

- TEACHER DASHBOARD TO CREATE AN TEST



B MCQ Test Platform
Teacher Dashboard

Madhav Khobare
@mk
Logout

< Back to Dashboard

Students (1)

124B1F001 X

SY IT OOP Div A
Code: EGR9

+ Upload Test

Tests (8)

Create Test

Test Name:
MidTerm

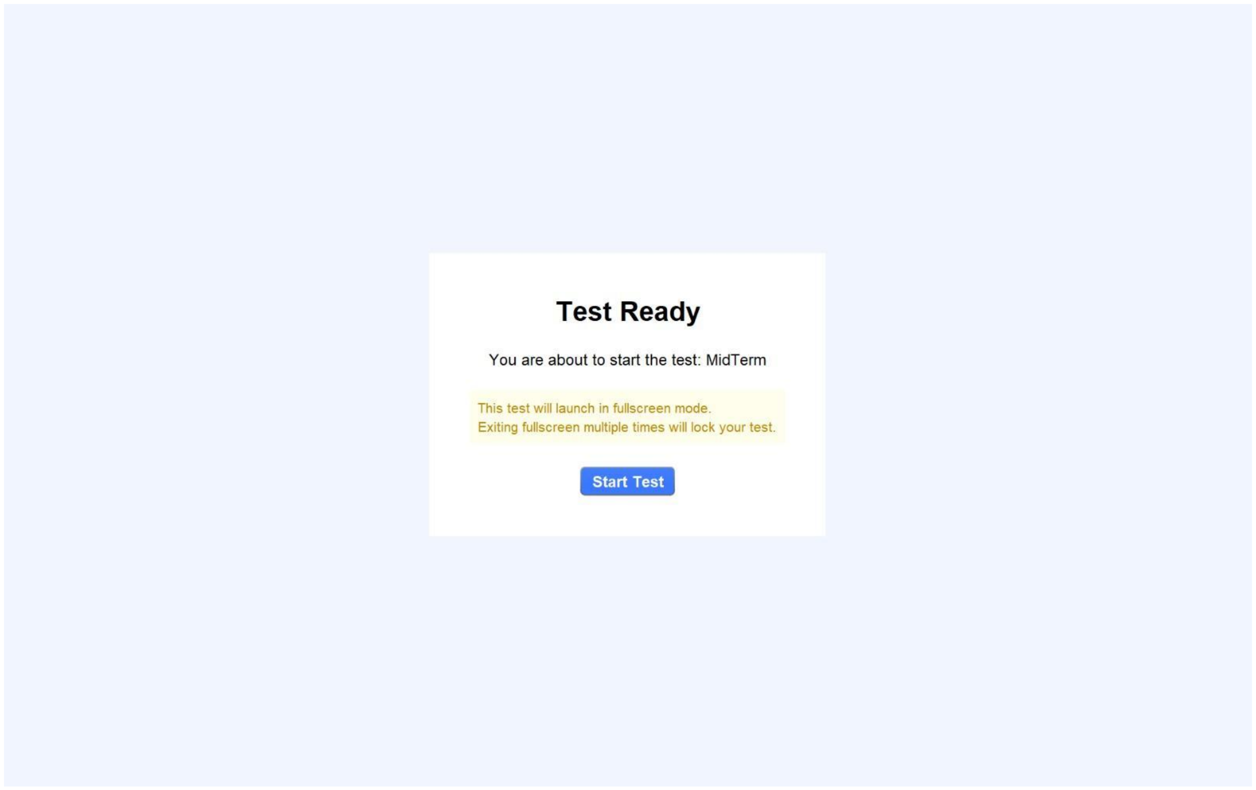
Correct Answers (A,B,C...):
A,C,C,A,A,B,B,C,C,C

Select PDF
Sample_For_Quiz.pdf

OK Cancel

ENDED 10 que Submissions X
ENDED 3 ques Submissions X
ENDED 10 que Submissions X
ENDED 10 questions Submissions X
FA2
ENDED 10 questions Submissions X
FA201
ENDED 10 questions Submissions X

- STUDENT SCREEN BEFORE TEST STARTS



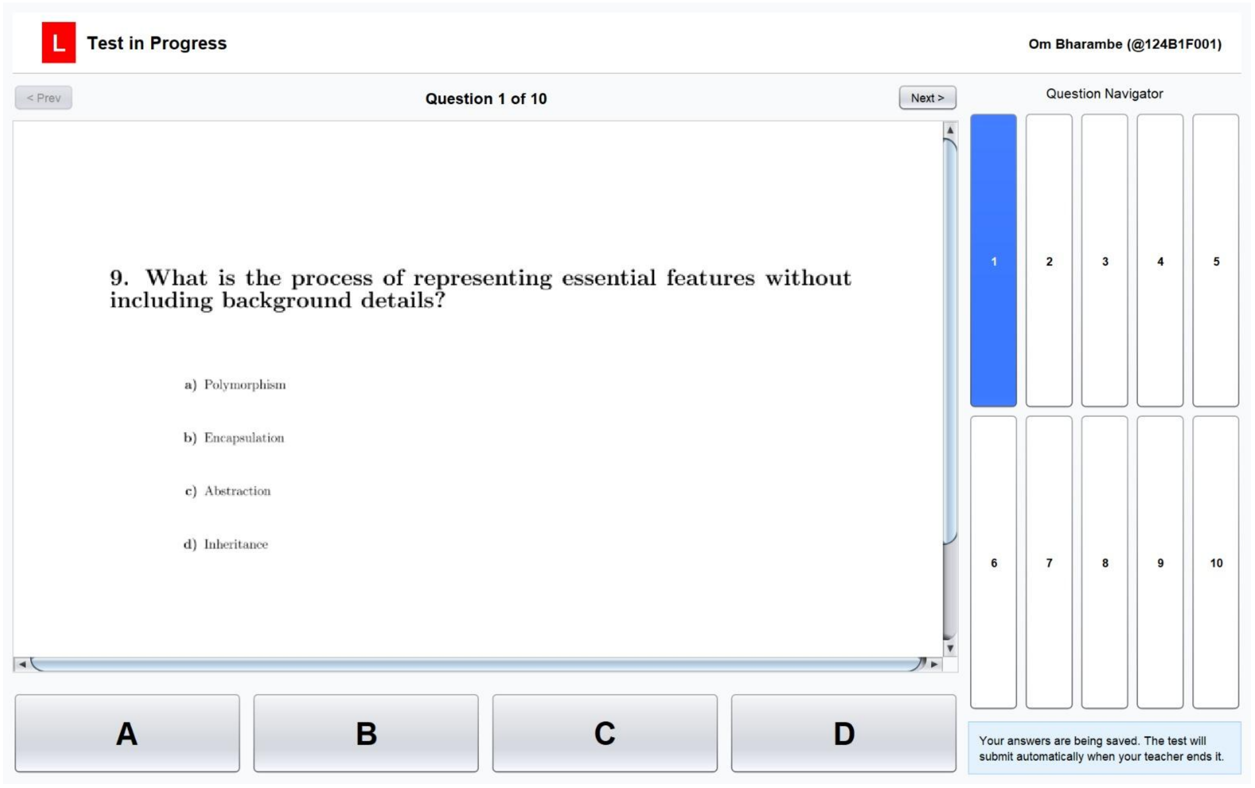
Test Ready

You are about to start the test: MidTerm

This test will launch in fullscreen mode.
Exiting fullscreen multiple times will lock your test.

Start Test

- STUDENT SCREEN FOR ASSESSMENT



L Test in Progress

Om Bharambe (@124B1F001)

< Prev Question 1 of 10 Next >

Question Navigator

9. What is the process of representing essential features without including background details?

a) Polymorphism

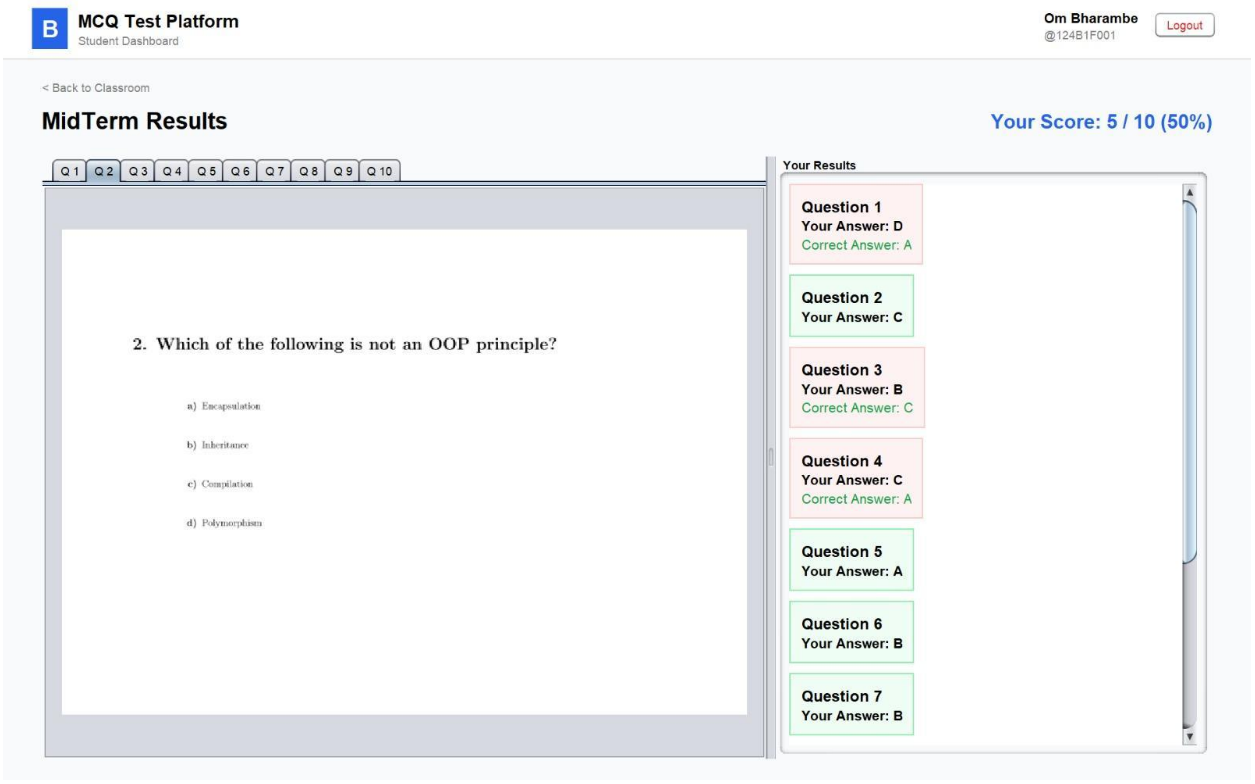
b) Encapsulation

c) Abstraction

d) Inheritance

Your answers are being saved. The test will submit automatically when your teacher ends it.

- STUDENT SCREEN TO CHECK THE SCORES



B MCQ Test Platform

Student Dashboard

Om Bharambe

@124B1F001

MidTerm Results

Your Score: 5 / 10 (50%)

2. Which of the following is not an OOP principle?

a) Encapsulation

b) Inheritance

c) Compilation

d) Polymorphism

Your Results

Question 1
Your Answer: D
Correct Answer: A

Question 2
Your Answer: C

Question 3
Your Answer: B
Correct Answer: C

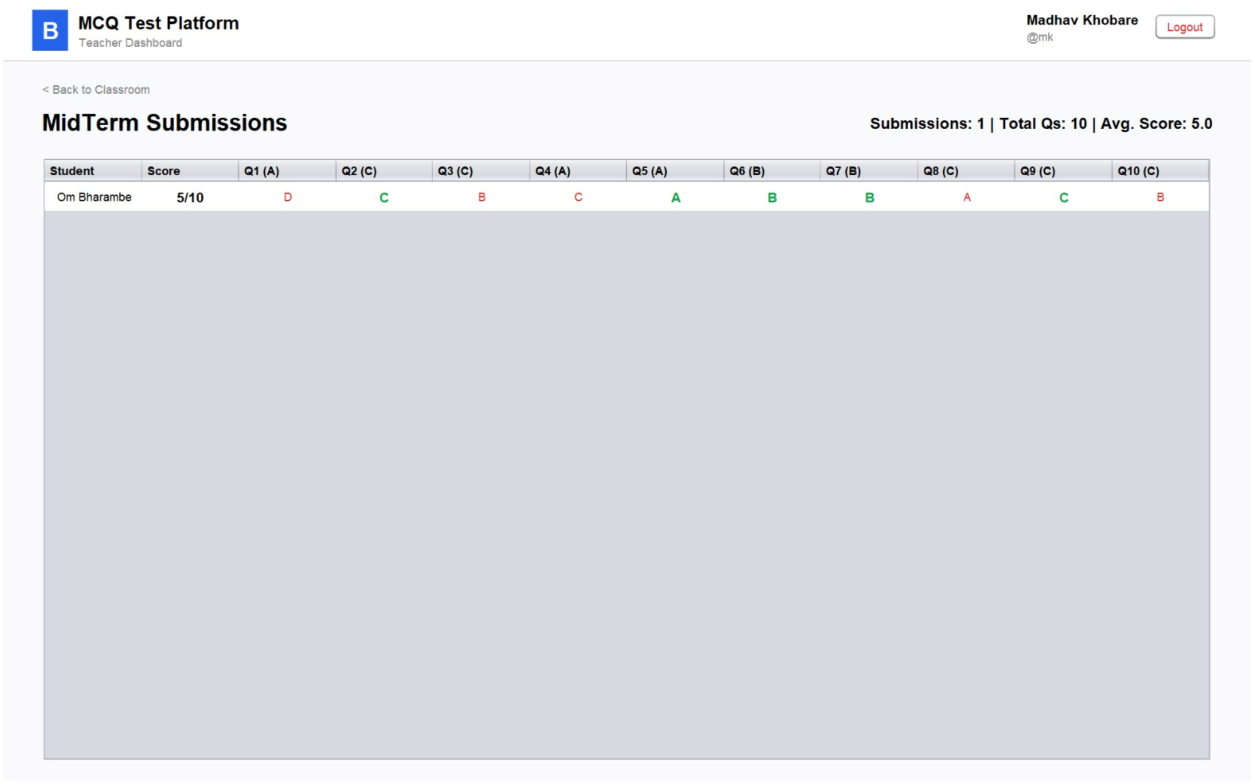
Question 4
Your Answer: C
Correct Answer: A

Question 5
Your Answer: A

Question 6
Your Answer: B

Question 7
Your Answer: B

- TEACHER SCREEN TO CHECK THE RESULTS



MCQ Test Platform
Teacher Dashboard

Madhav Khobare
@mk
Logout

< Back to Classroom

MidTerm Submissions

Submissions: 1 | Total Qs: 10 | Avg. Score: 5.0

Student	Score	Q1 (A)	Q2 (C)	Q3 (C)	Q4 (A)	Q5 (A)	Q6 (B)	Q7 (B)	Q8 (C)	Q9 (C)	Q10 (C)
Om Bharambe	5/10	D	C	B	C	A	B	B	A	C	B

6. CONCLUSION

The Quiz Application project provided hands-on experience with object-oriented design and Java technologies. Implementing this project reinforced OOP principles: for example, organizing entities into classes (**User**, **Question**, **Answer**), using inheritance or interfaces for flexibility, and encapsulating behavior within methods[1][2]. Working with Spring Boot simplified many aspects of web development; as Spring guides note, such an application can be "100% pure Java" without manual XML configuration[13]. We gained practical skills in building a REST API with Spring Data JPA for database operations and Spring Security for user authentication. On the client side, integrating libraries like Gson and PDFBox demonstrated data interchange and reporting capabilities.

As a learning outcome, this project solidified understanding of Java OOP and contemporary frameworks, bridging theoretical concepts with coding practice. In future work, the application could be enhanced by adding features such as timed quizzes, a richer question bank with categories, user role management (e.g. admin vs student), better UI/UX, and deployment to a cloud platform. Incorporating progress charts, email notifications, or mobile interface are also possible extensions. These enhancements would further leverage Java's ecosystem and design patterns to create a more robust e-quiz platform.

7. REFERENCES

- GeeksforGeeks – *"Java OOP (Object Oriented Programming) Concepts"*[1]. (Explains core OOP features and rationale.)
- Spring.io Guides – *"Building an Application with Spring Boot"*[13] and *"Accessing Data with MySQL"*[14]. (Official Spring tutorials for creating a Spring Boot app and configuring MySQL.)
- TutorialsPoint – *"PDFBox – Creating a PDF Document"*[16]. (Guide to using Apache PDFBox in Java.)
- Mkyong.com – *"How to parse JSON using Gson"*[15]. (Example of JSON–Java mapping with Gson.)
- TutorialsPoint – *"Java OOPs (Object-Oriented Programming) Concepts"*[17]. (Overview of OOP paradigms and benefits.)

---

[1] [2] Java OOP(Object Oriented Programming) Concepts - GeeksforGeeks

https://www.geeksforgeeks.org/java/object-oriented-programming-oops-concept-in-java/

[3] [4] OOP_Mini_Project_Report_Template.pdf

file://file_00000000b17c72089323f11454bd0b7e

[5] [6] [7] [8] [9] [10] pom.xml

file://file_000000001174720c8e4c8fb330d49fe8

[11] [12] pom1.xml

file://file_00000000afd472099f156fb574f361a6

[13] Getting Started | Building an Application with Spring Boot

https://spring.io/guides/gs/spring-boot/

[14] Getting Started | Accessing data with MySQL

https://spring.io/guides/gs/accessing-data-mysql/

[15] How to parse JSON using Gson - Mkyong.com

https://mkyong.com/java/how-to-parse-json-with-gson/

[16] PDFBox - Creating a PDF Document

https://www.tutorialspoint.com/pdfbox/pdfbox_creating_a_pdf_document.htm

[17] Java - OOPs (Object-Oriented Programming) Concepts

https://www.tutorialspoint.com/java/java_oops_concepts.htm