DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

CHANDIGARH UNIVERSITY

Discover. Learn. Empower.

# Report File FULL STACK


**Student Name:** Sukhjinder Singh

**UID:** 23BAI70078

**Branch:** BE-AIT-CSE

**Section/Group:** 23AIT-KRG-G2


**Semester:** 5th

**Date of Performance:** 25th Aug, 2025

**Subject Name:** Full Stack

**Subject Code:** 23CSP-339

---

## Project Report: Online Quiz Application

---

## 1. AIM

The aim of this project is to build an **Online Quiz Application** where users can:

- Attempt quizzes with multiple-choice questions.
- Get instant feedback on their answers.
- View their final score at the end of the quiz.
- Store quiz questions and user responses in a database.

This project helps students understand **frontend interaction, backend logic, and database management**.

---

## 2. TECH STACK

### Frontend

- **HTML5** → Structure of quiz pages.
- **CSS3** → Styling quiz interface and results.
- **JavaScript (Vanilla/React)** → Rendering questions dynamically, tracking score.

### Backend

- **Node.js** → Backend runtime.
- **Express.js** → API handling and routing.

### Database

- **MongoDB** → Stores questions, options, correct answers, and user results.

### Tools

- **VS Code** → Editor.
- **Postman** → Testing quiz APIs.
- **Browser** → Running the quiz.

## 3. THEORY / BACKGROUND

The Online Quiz Application uses the **client-server architecture**:

- **Client (HTML, CSS, JS)** → Displays quiz questions, accepts user answers, shows score.
- **Server (Node + Express)** → Provides quiz questions, checks answers, stores results.
- **Database (MongoDB)** → Stores quizzes, answers, and scores.

**REST API Endpoints:**

- `GET /quiz` → Fetch quiz questions.
- `POST /quiz/submit` → Submit answers and calculate score.
- `GET /results/:userId` → Fetch past results for a user.

---

## 4. PROCEDURE / IMPLEMENTATION

### Step 1: Setup Environment

1. Install Node.js.
2. Create folder quiz-app.
3. Run:

   npm init -y

   npm install express mongoose body-parser cors

### Step 2: Backend

- Setup server.js with Express and MongoDB connection.
- Create APIs for fetching questions and submitting answers.

### Step 3: Frontend

- Create index.html with quiz UI.
- Use JavaScript to display questions one by one.
- Use fetch() to send answers to backend.

### Step 4: Run Application

- Start backend:

  node server.js
- Open index.html in browser.
- Attempt quiz, submit, and view results.

---

## 5. CONCLUSION

This project demonstrates how to build a **full-stack Online Quiz Application**.
It provides experience in:

- **Frontend** → Dynamic quiz interface with HTML, CSS, JS.
- **Backend** → REST APIs using Node.js and Express.
- **Database** → MongoDB for quiz storage and results.

The project showcases **real-time user interaction, validation, and data persistence**.