



Tribhuvan University
Institute of Science and Technology
An Internship Report
on
“Chess MCQuiz Application”
Backend Nodejs Development
At
Swift Technology Pvt. Ltd.

Submitted To:
Department of Information Technology
National College of Computer Studies

**In partial fulfillment of the requirement for the Bachelor Degree in Computer
Science and Information Technology**

Submitted By:
Pramit Amatya (26327/077)

Under the Supervision of
Mr. Sumit Ghising

May, 2025

STUDENT DECLARATION

This is to certify that I have completed the report entitled “**Internship Report on Backend Nodejs at Swift Technology Pvt. Ltd.**” under the guidance of “**Mr. Sumit Ghising**” in partial fulfillment of requirements for the **Bachelor of Science in Computer Science and Information Technology “BSc. CSIT”** at Institute of Science and Technology.

This is my original work, and I have not submitted it earlier elsewhere.

Date: May 2025

Signature:

Pramit Amatya

ACKNOWLEDGEMENT

I would like to express my heartfelt gratitude to all individuals who supported me throughout my internship. Focusing mainly on my mentor, **Mr. Anil Yogi**, for giving me the opportunity to learn and improve my skills as a backend developer. His mentorship, advice, and assistance during the internship period has greatly contributed towards my learning and professional development.

I also appreciate my internship supervisor, **Mr. Sumit Ghising**, for his valuable supervision, encouragement, support and guidance throughout the internship. His expertise and feedback have been crucial in shaping the direction of the work. I would also like to thank the Project Supervisor, **Mr. Anil Yogi**, for his constant support and enlightening suggestions during the internship.

Lastly, I would like to acknowledge the support and collaboration environment of my fellow department as well as office colleagues for their contribution to my improvement in professional and problem-solving domain.

ABSTRACT

This project outlines my experience as an intern at Swift Technology Pvt. Ltd., focusing on the development of a Chess MCQuiz application. This web application is designed to enhance chess learning by providing interactive puzzles and multiple-choice questions that test users' tactical and strategic understanding. Developed using Node.js and Express for the backend and React for the frontend, the application ensures seamless data flow, efficient user interactions, and a secure system architecture. The report explores the project's objectives, technical strategies, and the challenges encountered during development. The report concludes with an evaluation of how this internship contributed to my growth in Backend development and backend expertise. It also provides insights into potential future enhancements for the project and reflects on the valuable experience gained at Swift Technology Pvt. Ltd.

Keywords: Multiple Choice Questions, Nodejs, Expressjs, API Integration

TABLE OF CONTENTS

STUDENT DECLARATION.....	ii
ACKNOWLEDGEMENT.....	iii
ABSTRACT.....	iv
LIST OF TABLES	vii
LIST OF FIGURES.....	viii
LIST OF ABBREVIATIONS	ix
CHAPTER 1: INTRODUCTION	1
1.1 INTRODUCTION	1
1.2 PROBLEM STATEMENT	1
1.3 OBJECTIVES	2
1.4 SCOPE AND LIMITATIONS	2
1.5 REPORT ORGANIZATION	3
CHAPTER 2: ORGANIZATION DETAILS	4
2.1 ORGANIZATION DETAILS	4
2.2 ORGANIZATIONAL HIERARCHY	5
2.3 WORKING DOMAINS OF ORGANIZATION	5
2.4 DESCRIPTION OF INTERN DEPARTMENT/UNIT	6
2.5 LITERATURE REVIEW	7
CHAPTER 3: INTERNSHIP ACTIVITIES	9
3.1 ROLES AND RESPONSIBILITIES	9
3.2 WEEKLY LOG	9
3.3 DESCRIPTION OF PROJECTS INVOLVED DURING INTERNSHIP.....	11
3.3.1 JWT Authentication using ExpressJs.....	11
3.3.2 gRPC Mini-Project.....	13
3.3.3 Chess MCQuiz.....	15

3.4 TASKS/ACTIVITIES PERFORMED.....	16
3.4.1 System Analysis.....	19
3.4.2 Implementation Tools	20
3.4.3 Implementation Details	20
3.4.4 Testing	21
CHAPTER 4: CONCLUSION AND LEARNING OUTCOMES	23
4.1 CONCLUSION	23
4.2 LEARNING OUTCOMES	23
REFERENCES.....	24
APPENDICES	

LIST OF TABLES

Table 2.1. 1: Organizational Details.....	4
Figure 2.2.1 Organizational Hierarchy	5
Table 2.3.1: Working Domains of Organization.....	5
Table 2.4.1: Contact Details of Mentor	6
Table 3.2.1: Weekly Log	9
Table 4.4.4. 1: Test Cases for User Authentication Testing	21
Table 4.4.4. 2: Test Cases for Profile Testing	21
Table 4.4.4. 3: Test Cases for Admin Dashboard Testing	22

LIST OF FIGURES

Figure 2.2.1 Organizational Hierarchy	Error! Bookmark not defined.
Figure 3.3.1.1 Login	Error! Bookmark not defined.
Figure 3.3.1.2 Blog	Error! Bookmark not defined.
Figure 3.3.2.1 Auth proto	Error! Bookmark not defined.
Figure 3.3.2.2 gRPC client.js	Error! Bookmark not defined.
Figure 3.3.2.3 gRPC login	Error! Bookmark not defined.
Figure 3.4.1 User Authentication	Error! Bookmark not defined.
Figure 3.4.2 User Details	Error! Bookmark not defined.
Figure 3.4.3 Admin role and authorization	Error! Bookmark not defined.
Figure 3.4.4 Admin CRUD	Error! Bookmark not defined.

LIST OF ABBREVIATIONS

API	Application Programming Interface
CRUD	Create, Read, Update, Delete
CSS	Cascading Style Sheets
DB	Database
JSON	JavaScript Object Notation
JWT	JSON Web Token
MVC	Model-View-Controller
MySQL	My Structured Query Language
Node.js	Node JavaScript
REST	Representational State Transfer
SQL	Structured Query Language
UI	User Interface

CHAPTER 1: INTRODUCTION

1.1 INTRODUCTION

With the increasing demand for interactive and educational chess platforms, the development of a Chess MCQuiz application using Node.js and React presents an opportunity to enhance chess learning through technology. This web application is designed to help players of all skill levels improve their tactical and strategic understanding by solving puzzles and answering chess-related multiple-choice questions. The platform allows users to engage with chess puzzles, track their progress, and receive instant feedback on their answers. Key features include real-time scoring, difficulty-based puzzles, leaderboards, and user authentication for personalized learning. Additionally, the system incorporates a ranking system to encourage user engagement and continuous improvement. Node.js and Express serve as the backend technologies, providing a scalable and efficient system for handling user data, game logic, and authentication. MongoDB is used as the database to store puzzles, questions, and user statistics. React plays a crucial role in delivering a dynamic, responsive, and interactive front-end, ensuring a smooth user experience. Security and performance are key considerations, with JWT-based authentication, role management, and data encryption ensuring a secure and reliable platform. This project aims to create an engaging and educational chess learning experience by combining game-based learning with structured assessments, ultimately enhancing players' tactical thinking and decision-making abilities.

1.2 PROBLEM STATEMENT

Chess learners often struggle to find structured and interactive platforms to practice tactics and strategies effectively. Traditional methods lack immediate feedback, progress tracking, and engagement, making learning less efficient. The Chess MCQuiz addresses these challenges by providing an interactive platform where users can solve chess puzzles, answer MCQs, and track their progress. However, developing such a system presents challenges like efficient database management, seamless backend-frontend integration, real-time user engagement, and data security. This project aims to tackle these challenges by implementing optimized backend services, secure authentication, and a structured learning experience, ultimately enhancing users' tactical and strategic skills in chess.

1.3 OBJECTIVES

- Develop a Chess MCQuiz application using Node.js (Express) and React, featuring user authentication, role-based access control, real-time scoring, leaderboards, and progress tracking.
- Provide a secure, engaging, and personalized platform for users to enhance their chess knowledge interactively and encourage competitive learning, and support long-term chess skill development.

1.4 SCOPE AND LIMITATIONS

Scopes:

- Develop a front-end solution for chess puzzle solving and MCQ interaction using React.
- Implement features such as user authentication, progress tracking, leaderboards, and puzzle management.
- Integrate real-time scoring and feedback mechanisms to enhance user engagement and learning.
- Develop a backend using Node.js (Express) to handle user data, puzzles, and MCQ options efficiently.

Limitations:

- Limited access to external APIs for integrating third-party chess resources or services.
- Full real-time multiplayer interaction was restricted due to platform scope and technical constraints.
- The project was focused primarily on puzzle-solving and MCQ features, leaving out advanced AI-based chess analysis or real-time game play.

1.5 REPORT ORGANIZATION

- **Chapter 1: Introduction**

Provides an overview of the internship project, including the problem statement, objectives, scope, limitations, and structure of the report.

- **Chapter 2: Organization Details and Literature Review**

Describes the organization where the internship was completed. It includes the organizational structure, working domains, the department/unit where the internship was carried out, and a review of related studies.

- **Chapter 3: Internship Activities**

Details the intern's roles and responsibilities, weekly log of tasks, description of the projects involved, and the technical activities performed during the internship.

- **Chapter 4: Conclusion and Learning Outcomes**

Summarizes the overall experience of the internship and highlights the key skills and knowledge gained throughout the period.

CHAPTER 2: ORGANIZATION DETAILS

2.1 ORGANIZATION DETAILS

Swift Technology Pvt. Ltd. is a prominent technology company focused on delivering cutting-edge solutions in the field of software development. Established in [year], Swift Technology has rapidly grown into a leader in providing innovative, high-performance web and mobile applications across various industries. The company has earned a strong reputation for delivering scalable and secure solutions, particularly in the education, gaming, and e-commerce sectors.

Swift Technology specializes in full-stack development, with a focus on Node.js, React, and other modern technologies that enable the creation of dynamic, user-centric applications. The company also offers backend solutions like database management, cloud integration, and real-time systems.

With a strong focus on innovation and quality, Swift Technology strives to provide its clients with intuitive and scalable products that drive business success. The team at Swift Technology is made up of highly skilled professionals who excel in both frontend and backend development, ensuring that projects are completed with precision and on time. The company prides itself on its collaborative work culture, which encourages continuous learning, problem-solving, and a proactive approach to tackling challenges.

As a forward-thinking organization, Swift Technology is committed to expanding its market presence by exploring new opportunities and adopting the latest technologies, ensuring it remains a leader in the competitive software development landscape.

Table 2.1. 1: Organizational Details

CEO	Neeraj Dhungana
Department	Tech
Address	3rd Floor, IME Complex, Kathmandu
Phone	01-4002535
Website	https://www.swifttech.com.np/

2.2 ORGANIZATIONAL HIERARCHY



Figure 2.2.1 Organizational Hierarchy

2.3 WORKING DOMAINS OF ORGANIZATION

Table 2.3.1: Working Domains of Organization

Domain	Description	Key Aspects
Executive Leadership	Sets the company's overall vision, strategy and governance.	<ul style="list-style-type: none"> • Define mission & long-term goals • Allocate major resources & budgets • Engage with board, investors and key partners • Track high-level KPIs

Development	Designs, builds, tests and maintains the company's software products or platforms.	<ul style="list-style-type: none"> • Requirements gathering & analysis • System architecture & design • Coding standards & best practices • Automated testing & continuous integration
HR & Administration	Manages the employee lifecycle and keeps the office running smoothly.	<ul style="list-style-type: none"> • Recruitment & onboarding • Performance management & training • Payroll, benefits & compliance • Facilities, vendor & office-support services

2.4 DESCRIPTION OF INTERN DEPARTMENT/UNIT

The Tech (the involved internship department) department at Swift Technology. First, they gather and prioritize requirements, and they design system architectures and craft prototypes that bring stakeholders' visions to life. Developers then write and review clean, maintainable code, while automated tests and continuous integration ensure quality and stability at every step. Once features pass testing, the team automates deployment to staging and production environments, monitors performance, and addresses any issues.

Table 2.4.1: Contact Details of Mentor

Name of Mentor	Mr. Anil Yogi
Position	Sr. Software Engineer
Email	anil.yogi@swifttech.com.np
Phone	9843263799

2.5 LITERATURE REVIEW

This section explores the core technologies relevant to the development of a web-based academy management system using a Node.js-based technology stack. It highlights modern practices in backend development, database integration, API testing, authentication, and software development tools.

Backend Development: Node.js

Node.js is an open-source, cross-platform JavaScript runtime environment that executes code outside a web browser. It is event-driven and non-blocking, making it ideal for developing scalable and efficient web applications. Node.js supports asynchronous operations and is well-suited for backend services that require high performance, such as real-time user data processing in academy management systems (Next.js Documentation, 2023).

Web Framework: Express.js

Express.js is a lightweight and flexible web framework for Node.js that simplifies the development of web servers and APIs. It supports middleware architecture, making it easy to manage routing, request validation, and error handling. Express.js is commonly used in backend systems to serve RESTful APIs, essential for CRUD operations in applications such as managing students, courses, and attendance records (Express.js documentation, 2025).

Database Management: MySQL

MySQL is a widely-used open-source relational database management system known for its reliability and performance. It uses structured query language (SQL) for managing and querying relational data. In the context of academy systems, MySQL is suitable for handling structured records such as student information, course schedules, and payment transactions (MySQL reference manual, 2025).

ORM Integration: Sequelize

Sequelize is a promise-based ORM (Object-Relational Mapping) tool for Node.js applications that supports MySQL and other SQL dialects. It allows developers to work with databases using JavaScript objects instead of raw SQL queries. Sequelize simplifies complex database operations and supports model relationships, validations, and migrations,

which are useful in building a scalable academic management system (Sequelize ORM documentation, 2025)

Authentication and Authorization: JSON Web Tokens (JWT)

JWT is a compact and self-contained method for securely transmitting information between parties. It is commonly used for user authentication and session management in web applications. JWT allows for role-based access control, ensuring that different user types (e.g., students, trainers, administrators) can securely access only the features they are authorized to use (JWT.io introduction, 2025).

API Development and Testing: Postman

Postman is a popular tool for developing, testing, and documenting APIs. It allows developers to simulate HTTP requests, test endpoints, and monitor API responses efficiently. In an academy management system, Postman is essential for testing functions such as student registration, attendance tracking, or retrieving academic records (Postman API platform documentation, 2025).

Version Control: GitHub

GitHub is a cloud-based platform for version control using Git. It supports collaborative development, continuous integration, and project documentation. GitHub enables teams to track code changes, manage pull requests, and maintain backups of their source code, which is vital for maintaining the integrity and scalability of academy management applications (GitHub documentation, 2025).

Remote Procedure Calls: gRPC

gRPC is a high-performance, open-source framework developed by Google for making remote procedure calls across distributed systems. It uses Protocol Buffers (Protobuf) for efficient serialization and supports bi-directional streaming and strong typing. In a modular academy management system, gRPC enables fast, secure, and efficient communication between services — for example, between the user service, notification service, and scheduling modules. It is especially useful for large systems that need scalable microservice communication (gRPC documentation, 2025).

CHAPTER 3: INTERNSHIP ACTIVITIES

3.1 ROLES AND RESPONSIBILITIES

As a Backend Developer for Swift Technology Pvt. Ltd., my core responsibilities include:

- **Building API:** Designed and developed RESTful APIs using Node.js and Express for managing users, puzzles, and MCQs.
- **Database Management:** Structured and managed MySQL collections for storing and retrieving puzzle and user data efficiently.
- **Integration:** Connected backend APIs with frontend components to ensure smooth data flow and functionality.
- **Testing:** Performed unit and integration testing using Postman and Jest to ensure API reliability and performance.
- **UI/ UX Implementation:** Supported frontend design needs by providing structured and consistent API responses.
- **State Management:** Provided optimized backend data formats to support frontend state handling and user session tracking.
- **Version Control:** Used Git and GitHub for code management, following proper branching and commit practices.

3.2 WEEKLY LOG

Table 3.2.1: Weekly Log

Week 1	<ul style="list-style-type: none">• Learning and doing CRUD operation using ExpressJs and MySQL.• Learned Graceful Shutdown.• JWT Authentication and validation.
Week 2	<ul style="list-style-type: none">• Implemented user authentication.• Learned about middleware for session and implemented it.• Created JWT authentication in ExpressJs.

Week 3	<ul style="list-style-type: none"> • JWT authentication with MySQL. • Learning ORM using Sequelize. • Integration of backend with frontend. • Create Frontend Quiz page as well. • Implementing Sequelize in my project.
Week 4	<ul style="list-style-type: none"> • Learning Sequelize-cli, migrations and seeders. • Implementing into our project.
Week 5	<ul style="list-style-type: none"> • Learning gRPC. • Implementing into dummy project. • Created a user and blog mini-project from organizational point of view. • Quiz in array form in frontend.
Week 6	<ul style="list-style-type: none"> • Created Profile page in frontend. • Fetch user details in frontend from backend. • Login and Signup when unauthenticated and Home page and profile when authenticated.
Week 7	<ul style="list-style-type: none"> • Learned about session expiration in frontend. • Moved to Login when session expired. • Quiz result stored in database and fetched in frontend profile.
Week 8	<ul style="list-style-type: none"> • Quiz question is stored in database and it is fetched in frontend. • Implemented role-based access control (Admin vs. User). • Created API to post questionnaires.
Week 9	<ul style="list-style-type: none"> • Developed Frontend to add questions as admin. • Edit, Delete the questions.(Both backend and basic frontend).

Week 10	<ul style="list-style-type: none"> • Customize User Details and Change password • Did testing and implemented private keys in environment to make it safe.
---------	--

3.3 DESCRIPTION OF PROJECTS INVOLVED DURING INTERNSHIP

I was primarily involved in a backend-focused project for a web-based Chess MCQuiz. This project offered valuable exposure to backend development, API design, and system architecture within a full-stack development context.

3.3.1 JWT Authentication using ExpressJs

JWT Auth Blog API is a practice backend project built using Node.js and Express to understand and implement user authentication using JWT and basic CRUD operations. The project includes APIs for managing users and blog posts.

Key Features:

- **JWT Authentication:** Simple login and registration system using JSON Web Tokens for secure access.
- **User API:** Users can register with their email, username, password, and age.
- **Blog API:** Users can create, read, update, and delete blog posts with a title, description, and author.
- **Practice Project:** This was built for learning purposes to get hands-on experience with authentication and REST APIs in Express.

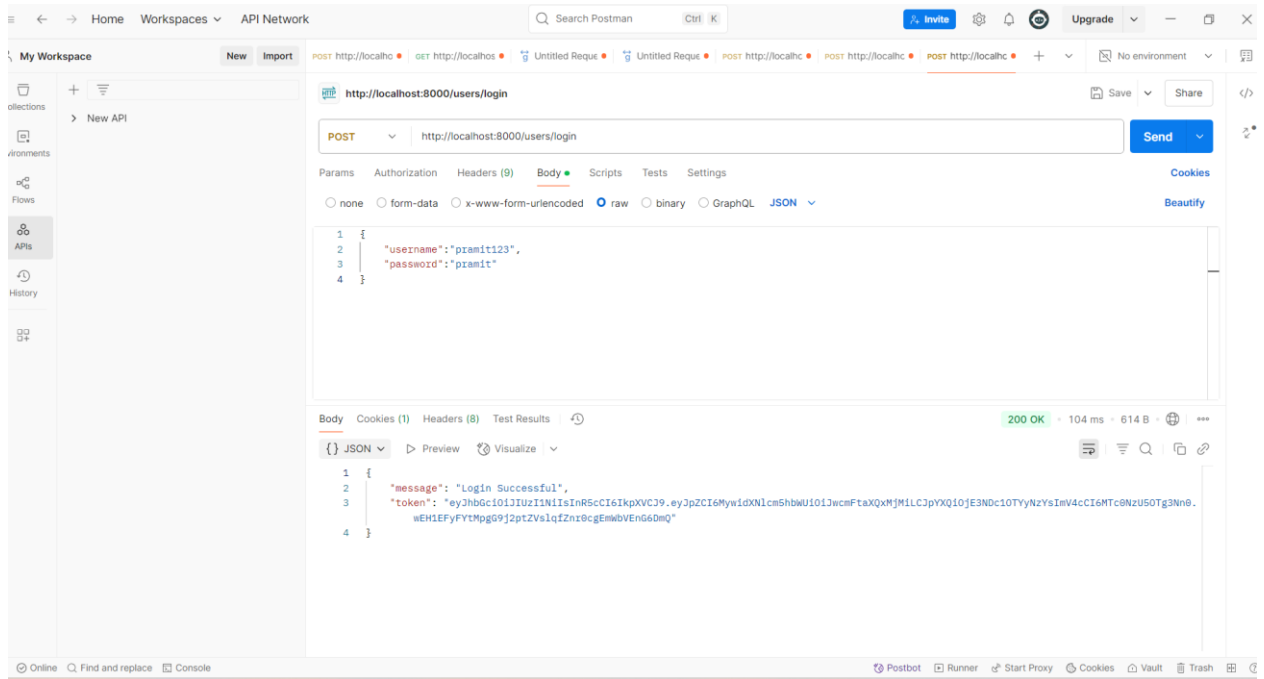


Figure 3.3.1.1 Login

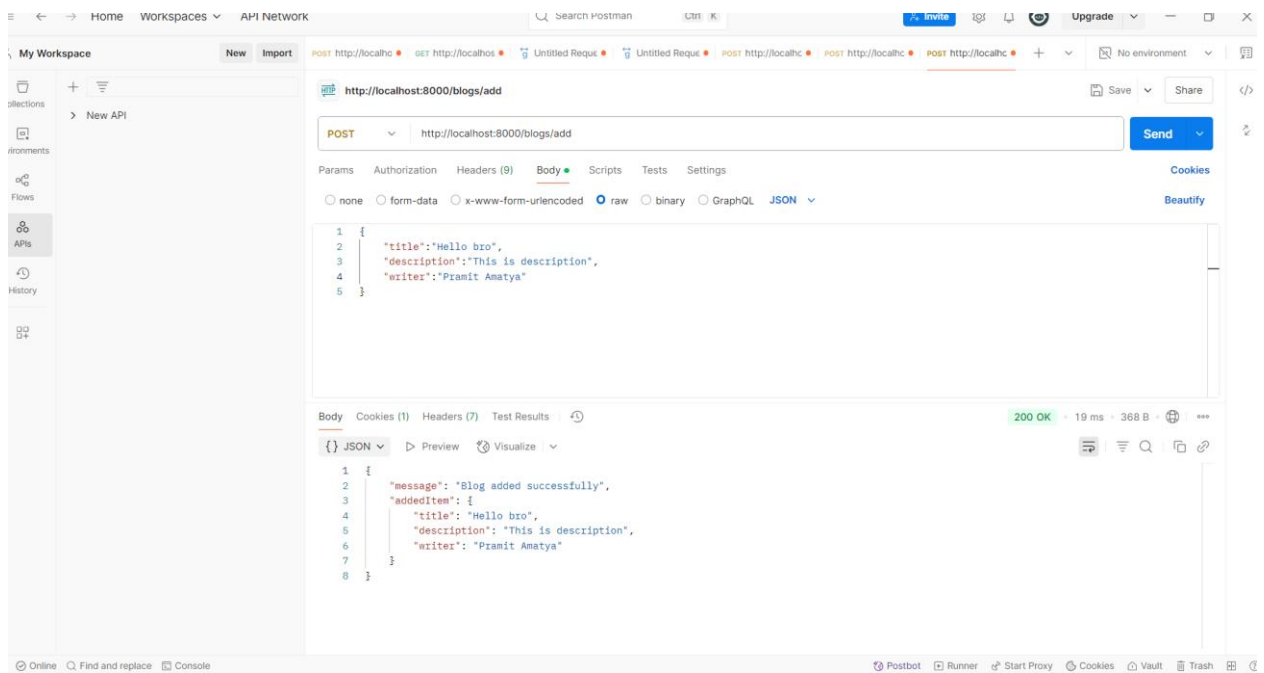


Figure 3.3.1.2 Blog

3.3.2 gRPC Mini-Project

gRPC auth blog service is a practice backend project built using Node.js and gRPC to learn how to implement authentication with JWT and manage user and blog data using protocol buffers and gRPC services instead of REST APIs.

Key Features:

- **JWT Authentication:** Secure login and registration system using JSON Web Tokens for user identity management.
- **User Service:** Handles user registration and login via gRPC with fields like email, username, password, and age, defined in auth.proto.
- **Blog Service:** Authenticated users can create, read, update, and delete blog posts with title, description, and author, defined in blog.proto.
- **Protected RPC Methods:** Blog-related gRPC methods are protected using JWT, ensuring only authenticated users can access them.
- **Protocol Buffers:** Used .proto files for defining structured and strongly typed messages and services.
- **Client-Server Communication:** Set up gRPC clients and servers to handle service calls across modules.
- **Modular Project Structure:** Cleanly separated into config, middleware, models, proto files, routes, and service implementations.
- **Practice Project:** Built to gain hands-on experience with gRPC in Node.js and understand how to build service-based architecture beyond REST APIs.

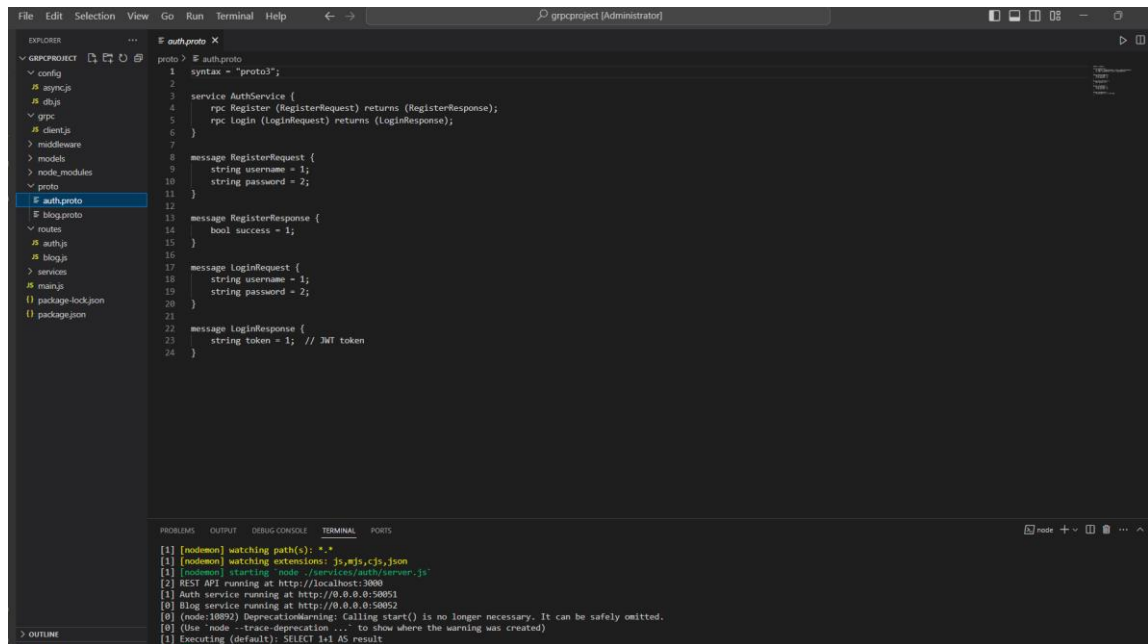


Figure 3.3.2.1 auth.proto

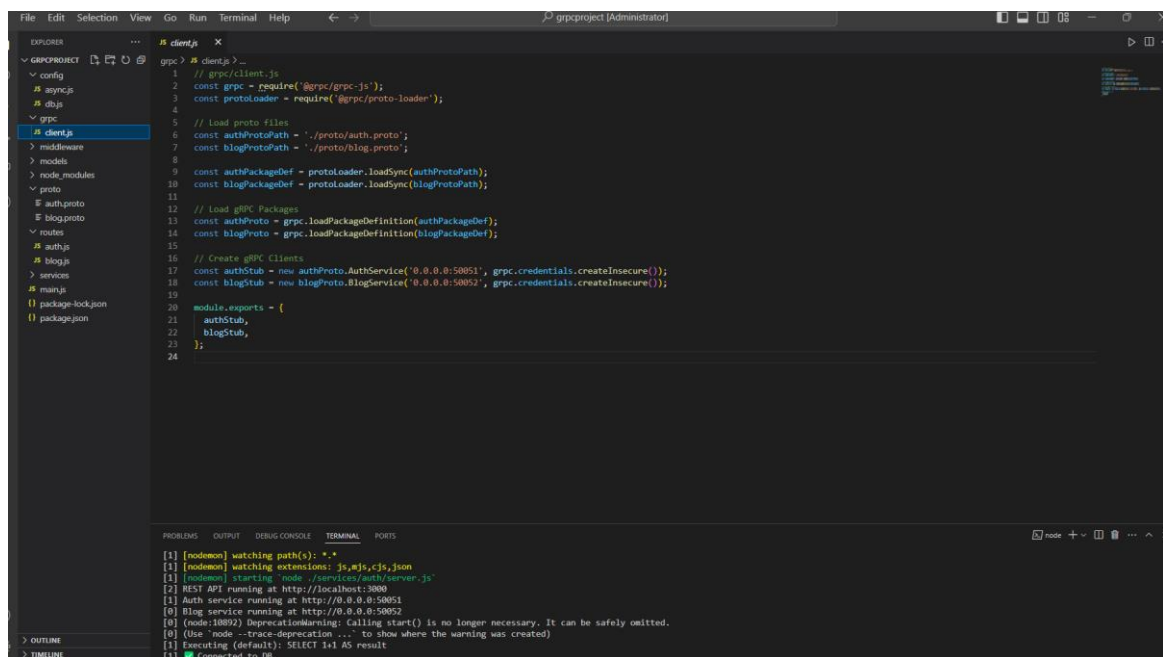


Figure 3.3.2.2 gRPC client.js

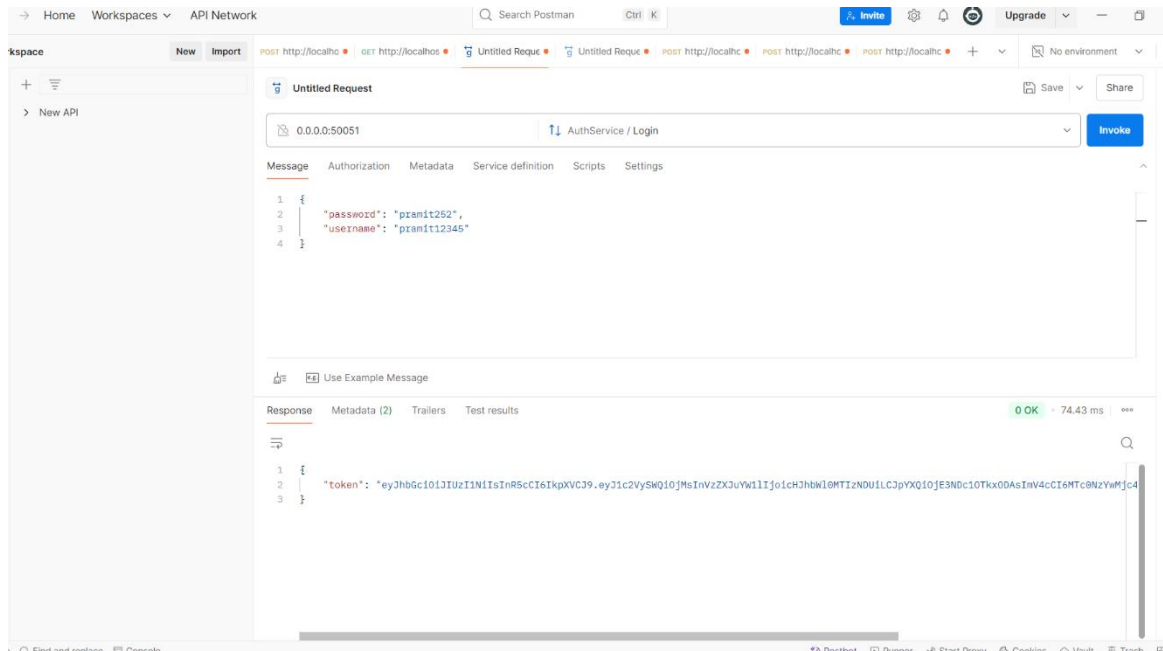


Figure 3.3.2.3 login

3.3.3 Chess MCQuiz

Chess MCQuiz is a web application designed to enhance chess learning through interactive puzzles and multiple-choice questions. It enables users—especially beginners and intermediate players—to test their tactical and strategic knowledge through structured quizzes and puzzle-solving exercises. The platform supports user authentication, personalized progress tracking, and puzzle management.

Key Features:

- **Interactive Chess Puzzles:** Users solve chess puzzles with MCQ-based answers to reinforce tactical learning.
- **User Authentication:** Secure login and registration system using JWT for session management.
- **Progress Tracking:** Users can view quiz results and track their performance history.
- **Admin Functionality:** Admins can add, update, and delete puzzles, MCQs, and manage users.

Technologies Used:

- **Frontend:** React with JavaScript
- **Backend:** Node.js with Express.js
- **Database:** MySQL with Sequelize
- **API Communication:** Fetch API / Axios
- **Authentication:** JSON Web Token (JWT)
- **API Testing:** Postman
- **Version Control:** Git and Github
- **Tools:** Postman, VS Code, Nodemon

My Contributions:

- Designed and developed RESTful APIs for user authentication, puzzle handling, and quiz results.
- Structured MySQL collections and implemented schema validation with Sequelize.
- Integrated backend APIs with frontend team using standardized response formats.
- Performed testing and debugging of API endpoints using Postman.
- Collaborated with the frontend team to align API functionality with UI/UX needs.

3.4 TASKS/ACTIVITIES PERFORMED

During my internship at Swift Technology Pvt. Ltd., I was primarily responsible for backend development of the Chess MCQuiz platform. My tasks involved designing backend architecture, building secure APIs, integrating with the frontend team, and optimizing data operations. Below are the major tasks and activities performed:

A. Learning and Onboarding

- **Project Understanding:** Reviewed the overall project goals, functional flow, and user types (admin, player).
- **Backend Stack Setup:** Set up the Node.js, Express, and MySQL environment along with essential tools like Postman and Git.

- **Codebase Exploration:** Studied routing structure, middleware usage, and modular folder organization.

B. Authentication & User Management

- **JWT-based Auth System:** Implemented secure user login, registration, and session management using JWT.
- **Role-based Access Control:** Restricted admin-only routes for puzzle and user management features.
- **User Profiles:** Enabled retrieval of user data and quiz history for performance tracking.

```

// userRoutes.js
const bcrypt = require("bcrypt");
const jwt = require("jsonwebtoken");
const con = require("../config/db");

const SECRET_KEY = "your_secret_key"; // Secret key for signing the JWT

// Signup Route
router.post("/signup", (req, res) => {
  const { name, age, username, password, cpassword } = req.body;

  if (!name || !age || !username || !password || !cpassword) {
    return res.status(400).json({ message: "All fields are required" });
  }

  if (!isNaN(age) || Number(age) <= 0) {
    return res
      .status(400)
      .json({ message: "Age must be a valid positive number" });
  }

  if (password !== cpassword) {
    return res.status(400).json({ message: "Passwords do not match" });
  }

  // Check if username already exists
  const checkQuery = "SELECT * FROM users WHERE username = ?";
  con.query(checkQuery, [username], async (err, results) => {
    if (err) {
      console.error("Error checking username:", err);
      return res.status(500).json({ message: "Database error" });
    }

    if (results.length > 0) {
      return res.status(409).json({ message: "Username already taken" });
    }

    const hashedPassword = await bcrypt.hash(password, 10);

    // Insert user if username is unique
    const insertQuery = "INSERT INTO users (name, age, username, password) VALUES (?, ?, ?, ?)";
    con.query(
      insertQuery,
      [name, age, username, hashedPassword],
      (err, result) => {
        if (err) {
          console.error("Error inserting user:", err);
          return res.status(500).json({ message: "Database error" });
        }
      }
    );
  });
});

// Login Route
router.post("/login", (req, res) => {
  const { username, password } = req.body;

  if (!username || !password) {
    return res
      .status(400)
      .json({ message: "Username and password are required" });
  }

  const loginQuery = "SELECT * FROM users WHERE username = ?";
  con.query(loginQuery, [username], async (err, results) => {
    if (err) {
      console.error("Error during login:", err);
      return res.status(500).json({ message: "Database error" });
    }

    if (results.length === 0) {
      return res.status(401).json({ message: "Invalid username or password" });
    }

    const user = results[0];

    // Verify password using bcrypt
    const isMatch = await bcrypt.compare(password, user.password);
    if (!isMatch) {
      return res.status(401).json({ message: "Invalid username or password" });
    }

    const token = jwt.sign(
      { id: user.id, username: user.username },
      SECRET_KEY,
      { expiresIn: "1h" }
    );

    res.status(200).json({ message: "Login successful", token });
  });
});

module.exports = router;

```

Figure 3.4.1 User Authentication

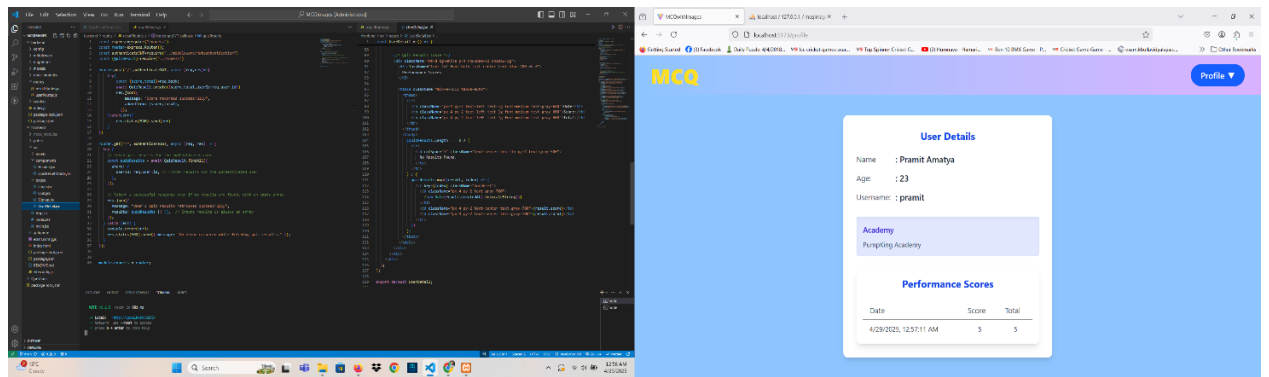


Figure 3.4.2 User Details

C. Puzzle & Quiz Functionality

- **Puzzle CRUD APIs:** Built routes for adding, editing, deleting, and retrieving puzzles with MCQs.
- **Quiz Attempt Flow:** Created endpoints to submit answers, evaluate responses, and store results.
- **Admin Controls:** Enabled bulk upload and management of puzzles by admin users.

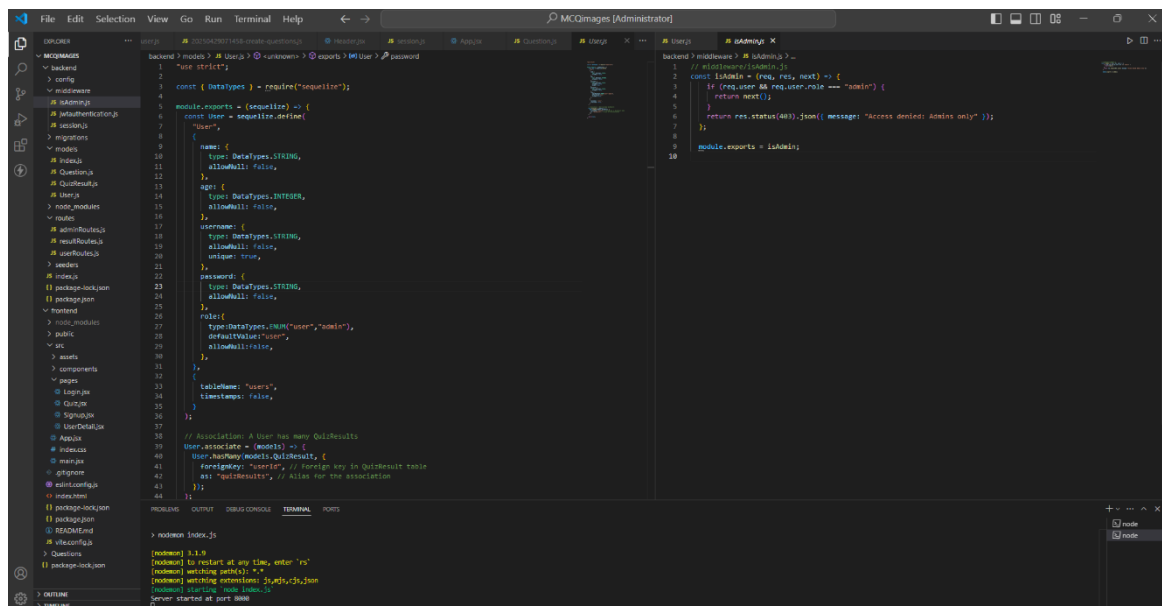


Figure 3.4.3 Admin role and authorization

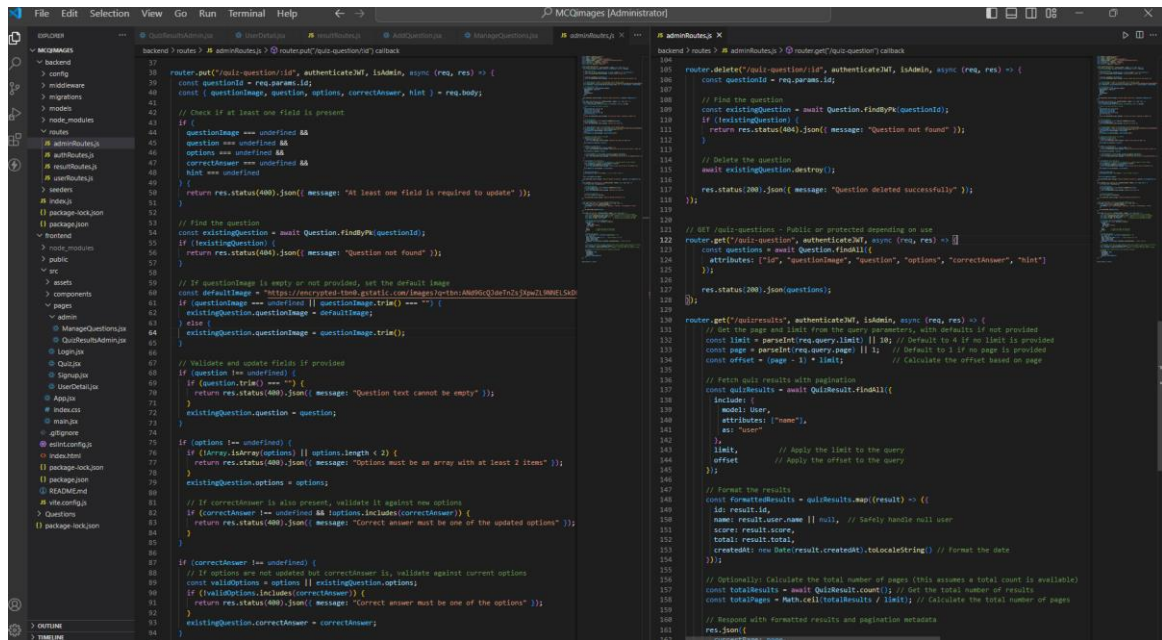


Figure 3.4.4 Admin CRUD

D. Performance, Testing & Documentation

- **API Testing:** Used Postman for testing all endpoints with both positive and edge cases.
- **Performance Improvements:** Optimized MongoDB queries and added pagination for puzzle listings.
- **Documentation:** Wrote detailed API documentation and usage instructions for frontend integration.

3.4.1 System Analysis

Functional Requirements:

- **User Authentication and Management:** JWT-based login, signup, and role-based access (user/admin).
- **Puzzle Management:** Admins can create, update, delete puzzles and attach MCQs.
- **User Quiz Flow:** Users can attempt puzzles, submit answers, and view results and progress.
- **Result Tracking:** Maintain user history for performance analysis and feedback.

Non-Functional Requirements:

- **Scalability:** Designed the API and database structure to support increasing puzzle and user load.
- **Security:** Secured endpoints with JWT authentication and validated inputs.
- **Reliability:** Maintained consistent and predictable API responses with error handling.
- **Maintainability:** Modular folder structure for routes, controllers, and models to ease future updates.

3.4.2 Implementation Tools

The implementation of the Chess MCQuiz involved a range of tools and technologies:

- **Backend Development:** Node.js with Express.js for building scalable APIs.
- **Version Control and Collaboration:** Git and GitHub for source code tracking and collaboration.
- **Database:** MySQL with Sequelize for schema modeling and query handling.
- **API Testing:** Postman for testing API endpoints during development and debugging.

3.4.3 Implementation Details

The following modules are part of the Chess MCQuiz implementation details:

A. API Structure:

Routes are organized by functionality (auth, puzzle, quiz), and each route uses controller functions for clean separation of logic.

B. State & Session Handling:

Session data is managed via JWT tokens stored client-side. Server only verifies tokens without maintaining user sessions in memory.

C. Routing & Middleware:

Used Express routing with middleware for authentication checks, input validation, and error handling across endpoints.

D. Data Modeling:

MySQL schemas are designed using Sequelize with proper references between users, puzzles, and quiz results to ensure relational consistency.

E. Input Validation:

Input data is validated at the backend using custom middleware functions to prevent malformed or malicious entries.

3.4.4 Testing

A. User Authentication Testing

Table 4.4.4. 1: Test Cases for User Authentication Testing

S.N.	Test Case	Steps	Expected Outcome	Actual Outcome	Status
1	User Registration	Fill registration form with valid details.	User registered successfully	User registered successfully	Pass
2	User, Admin Registration	Fill login form with valid details.	Logged in successfully	Logged in successfully	Pass

B. Profile Testing

Table 4.4.4. 2: Test Cases for Profile Testing

S.N.	Test Case	Steps	Expected Outcome	Actual Outcome	Status
1	Open profile	Get view-detail	User detail shown	User detail shown	Pass

C. Admin Dashboard Testing

Table 4.4.4. 3: Test Cases for Admin Dashboard Testing

S.N.	Test Case	Steps	Expected Outcome	Actual Outcome	Status
1	Add Question	Admin logged in & post add-question with valid json message	Artwork visible on marketplace	Artwork approved	Pass
2	View records of other users	Get score-users	Account access suspended	User deactivated	Pass

CHAPTER 4:

CONCLUSION AND LEARNING OUTCOMES

4.1 CONCLUSION

The Chess Puzzle MCQuiz system marks a significant advancement in combining technology with chess learning. It successfully delivers a gamified experience where users can engage with tactical puzzles in MCQ format, track their improvement, and learn from instant feedback. My backend development work ensured the system's core features—secure login, smooth data handling, and admin-level controls—ran efficiently, contributing to a platform that is both reliable and learner-friendly. With a well-structured API design and optimized database interactions, the platform is built to scale with future enhancements. This project not only empowers users but also supports educators in creating and managing high-quality chess content.

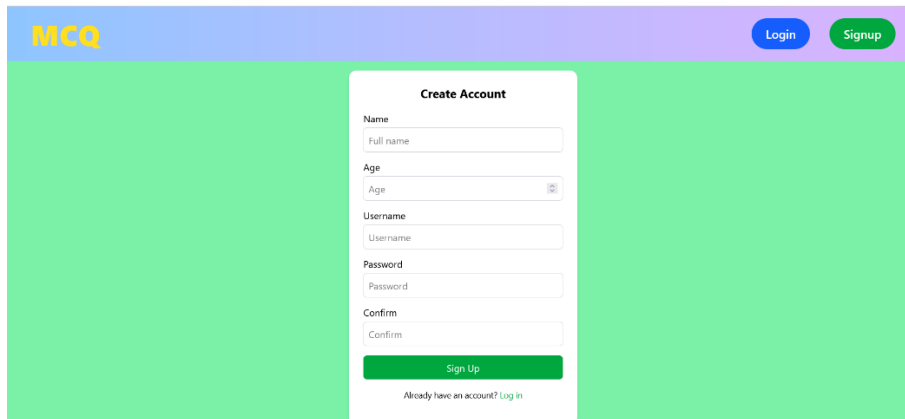
4.2 LEARNING OUTCOMES

This project allowed me to translate backend concepts into real-world solutions, bridging the gap between theory and practical application. I learned to design scalable and modular APIs, implement secure and efficient authentication mechanisms, and optimize database queries for performance and reliability. Working with real-time user interactions, quiz flow management, and admin-level operations improved my problem-solving skills and deepened my understanding of how to architect systems for user-centric platforms. Additionally, the experience gave me valuable insight into building educational tech products that are not only functional and efficient but also meaningful and impactful for learners and educators alike.

REFERENCES

- Nodemailer Documentation*. (2023). Retrieved from <https://nodemailer.com/about/>
- Postman API platform documentation*. (2025). Retrieved from Postman: <https://learning.postman.com/docs/>
- Abramov, D. &. (2015). *Redux*. Retrieved from <https://redux.js.org/>
- Atlassian using protocol*. (2025). Retrieved from Atlassian : <https://www.atlassian.com/blog/atlassian-engineering/using-protobuf-to-make-jira-cloud-faster>
- Express.js documentation*. (2025). Retrieved from ExpressJs: <https://expressjs.com/en/>
- GitHub documentation*. (2025). Retrieved from GitHub Docs: <https://docs.github.com/en/>
- gRPC documentation*. (2025). Retrieved from gRPC: <https://grpc.io/docs/>
- JWT.io introduction*. (2025). Retrieved from JSON Web Tokens (JWT): <https://jwt.io/introduction>
- MySQL reference manual*. (2025). Retrieved from MySQL: <https://dev.mysql.com/doc/>
- Next.js Documentation*. (2023). Retrieved from <https://nextjs.org/docs>
- Node.js documentation*. (2025 May). Retrieved from Node.js: <https://nodejs.org/en/>
- OECD. (2023). *Education and student information systems*. Retrieved from https://www.oecd.org/en/publications/oecd-digital-education-outlook-2023_c74f03de-en/full-report/education-and-student-information-systems_ef9f7b25.html
- Ozer, C. B. (2021). *Introduction to MongoDB*. Retrieved from <https://medium.com/codex/introduction-to-mongodb-16098b30d32b>
- Sequelize ORM documentation*. (2025). Retrieved from Sequelize: <https://sequelize.org/docs/v6/>
- VS Code documentation*. (2025). Retrieved from Visual Studio Code: <https://code.visualstudio.com/docs>

APPENDICES



The screenshot shows the 'Create Account' form on the MCQ website. The header is purple with the 'MCQ' logo in yellow on the left and 'Login' and 'Signup' buttons on the right. The background is green. The form is white and contains the following fields: 'Name' (Full name), 'Age' (with a dropdown arrow), 'Username', 'Password', and 'Confirm'. A green 'Sign Up' button is at the bottom of the form. Below the button, it says 'Already have an account? Log in'.

MCQ Login Signup

Create Account

Name
Full name

Age
Age

Username
Username

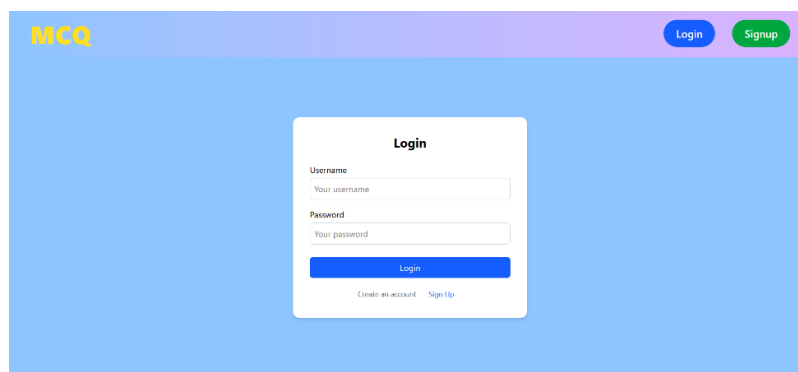
Password
Password

Confirm
Confirm

Sign Up

Already have an account? Log in

Sign in Page



The screenshot shows the 'Login' page on the MCQ website. The header is purple with the 'MCQ' logo in yellow on the left and 'Login' and 'Signup' buttons on the right. The background is blue. The form is white and contains the following fields: 'Username' (Your username) and 'Password' (Your password). A blue 'Login' button is at the bottom of the form. Below the button, it says 'Create an account' and 'Sign Up'.

MCQ Login Signup

Login

Username
Your username

Password
Your password

Login

Create an account Sign Up

Login Page



The screenshot shows the 'Quiz Home Page' on the MCQ website. The header is purple with the 'MCQ' logo in yellow on the left and a 'Profile' button with a dropdown arrow on the right. The background is dark blue. The page displays 'Your Score: 0 / 5' in yellow. Below this, there is a chessboard with a puzzle labeled 'Q. 1' and 'Mate in 3'. The chessboard shows a white king on e1, a white queen on d1, a white rook on a1, a white bishop on c1, a white knight on f1, a white pawn on g2, a black king on e8, a black queen on d8, a black rook on a8, a black bishop on c8, a black knight on f8, and a black pawn on g7. To the right of the chessboard, there are four blue buttons with yellow text: 'A: RxT7+', 'B: Rg6+', 'C: Ra6+', and 'D: Rh6+'.

MCQ Profile ▼

Your Score: 0 / 5

Q. 1

Mate in 3

A: RxT7+ B: Rg6+

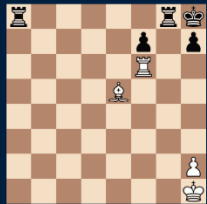
C: Ra6+ D: Rh6+

Quiz Home Page

MCQ

Profile ▼

Q. 1



Your Score: 0 / 5

Mate in 3

A: Rxf7+

B: Rg6+

C: Ra6+

D: Rh6+

Correct!

1. Ra6+ f6 2. Bxf6+ Rg7 3. Rxa8# checkmate.


Next Question

Quiz Page Correct Answer

MCQ

Profile ▼

Q. 2



Your Score: 1 / 5

Mate in 3

A: Rxf7+

B: Rg6+

C: Ra6+

D: Rh6+

Wrong! Correct Answer: **Ra6+**

1. Ra6+ f6 2. Bxf6+ Rg7 3. Rxa8# checkmate.

Next Question

Quiz Page Wrong Answer

🎉

Awesome Job!

🎉

Your Score: 3 / 5

Submit

Submit Component

MCQ

Profile

User Details

Name : Pramit Amatya

Age : 23

Username : pramit

Academy

PumpKing Academy

Performance Scores

Date	Score	Total
5/2/2025, 8:32:48 PM	3	5

View Detail Page

MCQ

Profile

Quiz Results

SN	Name	Score	Total	Date
1	Ram Shah	2	5	5/2/2025, 12:30:28 AM
2	Pramit Amatya	3	5	5/2/2025, 8:32:48 PM
3	PumpKing	4	5	5/3/2025, 12:31:57 AM
4	AYP	5	5	5/5/2025, 8:10:21 PM
5	Pramit Amatya	5	5	5/7/2025, 8:11:23 PM
6	Pramit Amatya	3	5	5/7/2025, 8:11:43 PM
7	Admin	1	5	5/9/2025, 6:06:01 PM
8	Admin	1	5	5/9/2025, 6:06:07 PM
9	Admin	2	5	5/9/2025, 6:06:14 PM
10	Admin	4	5	5/9/2025, 6:06:32 PM

Prev

Page 1 of 2

Next


Quiz Results

Close

Add New Question

What is this image?

rons_%28cropped%29.jpg/960px Tour_Eiffel_Wikimedia_Commons_%28cropped%29.jpg



Options:

Pirates

Eiffel Tower

Burj Khalifa

Swwayambhu

+ Add Option

This is Eiffel Tower

Add Question

Add Questions



- Italy
- China
- **Japan**
- Korea

Hint: This place is located in Japan.

[Delete](#) [Edit](#)

What is this image?



- Pirates
- **Eiffel Tower**
- Burj Khalifa
- Swwayambhu

Hint: This is Eiffel Tower

[Delete](#) [Edit](#)

Edit/Delete Questions