



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

Mentor's Recommendation

Supervisor's Recommendation

Approval Letter

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

Chess MCQuiz application 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

Mentor’s Recommendation.....	ii
Supervisor’s Recommendation	iii
Approval Letter	iv
Acknowledgement	v
Abstract	vi
List of Tables	ix
List of Figures.....	x
List of Abbreviations.....	xi
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.....	2
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	7
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.....	20
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.....	1

List of Tables

Table 2.1: Organizational Details	4
Table 2.2: Working Domains or Organization	5
Table 2.3: Contact Details of Mentor	6
Table 3.1: Weekly Log	9
Table 4.1: Test Cases for User Authentication Testing	21
Table 4.2: Test Cases for Profile Testing	22
Table 4.3: Test Cases for Admin Dashboard Testing	22

List of Figures

Figure 2.1: Organizational Hierarchy	5
Figure 3.1: Login Postman	12
Figure 3.2: Blog Postman	12
Figure 3.3: Auth.proto	14
Figure 3.4: gRPC client.js.....	14
Figure 3.5: gRPC Login Postman	15
Figure 3.6: User Authentication.....	17
Figure 3.7: User Details.....	18
Figure 3.8: Admin role and authorization	19
Figure 3.9: Admin CRUD.....	19

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

- To 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.
- To 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:

- Integrates image-based MCQs with chess puzzles to enhance learning through visual interaction.
- Tracks user progress and displays leaderboards based on MCQ and puzzle performance.
- Provides real-time scoring and feedback for MCQs to improve user engagement.

Limitations:

- Limited access to external APIs for integrating third-party chess resources or services.
- 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: 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 2.1: Organizational Hierarchy

2.3 Working Domains of Organization

Table 2.2: Working Domains or 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 2.3: Contact Details of Mentor

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

2.5 Literature Review

Recent studies have documented the growing use of modern web technologies in the development of digital management systems for educational institutions and specialized training centers, including chess academies. These systems commonly utilize backend frameworks, database solutions, and authentication mechanisms to support critical functionalities such as attendance tracking, session scheduling, and secure data management.

Node.js is frequently employed as the backend runtime environment due to its event-driven and non-blocking nature. It supports asynchronous operations, making it effective for handling real-time tasks such as user requests and data processing in educational platforms (OpenJS Foundation, 2023). To facilitate API development, Express.js is often used alongside Node.js. Its middleware architecture enables streamlined routing, request validation, and error handling—features essential for building robust and maintainable backend services (StrongLoop, 2025).

For data storage and management, MySQL is a widely adopted relational database management system. Its structured approach to data makes it well-suited for storing information such as student records, payment transactions, and training schedules (Oracle Corporation, 2025). To simplify interactions between the application and the database, many systems integrate Sequelize, a promise-based Object-Relational Mapping (ORM) tool. Sequelize enables developers to perform complex database operations using JavaScript objects, while also supporting model associations, validations, and migrations for scalability (Sequelize Contributors, 2025).

Security and user access control are typically managed through JSON Web Tokens (JWT). JWTs allow for secure, stateless user authentication and are commonly used to implement role-based access, ensuring that only authorized users—such as students, instructors, or administrators—can access specific system features (Auth0, 2025). In the development process, tools like Postman are used to test and document APIs. Postman helps simulate HTTP requests and responses, making it easier to verify the functionality of endpoints related to operations such as student registration, attendance tracking, and data retrieval (Postman Inc., 2025).

Collaborative software development practices are supported by platforms like GitHub, which enables teams to manage source code, track changes, and coordinate development through features like version control, pull requests, and issue tracking (GitHub, 2025). In systems built using a microservices architecture, gRPC is often used to enable efficient communication between different service modules. It supports fast, secure remote procedure calls using Protocol Buffers, making it suitable for integrating components such as scheduling, notifications, and user services in complex educational platforms (Google, 2025).

The adoption of these technologies reflects a broader shift toward web-based, modular, and cloud-friendly solutions in academic management. By leveraging these tools, institutions can develop systems that are scalable, maintainable, and responsive to the varied operational needs of modern educational environments.

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 3.1: Weekly Log

Week 1 (March 16 – March 22)	<ul style="list-style-type: none">• Learning and doing CRUD operation using ExpressJs and MySQL.• Learned Graceful Shutdown.• JWT Authentication and validation.
Week 2 (March 23 – March 29)	<ul style="list-style-type: none">• Implemented user authentication.• Learned about middleware for session and implemented it.• Created JWT authentication in ExpressJs.

<p>Week 3</p> <p>(March 30 – April 5)</p>	<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.
<p>Week 4</p> <p>(April 6 – April 12)</p>	<ul style="list-style-type: none"> • Learning Sequelize-cli, migrations and seeders. • Implementing into our project.
<p>Week 5</p> <p>(April 13– April 19)</p>	<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.
<p>Week 6</p> <p>(April 20 – April 26)</p>	<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.
<p>Week 7</p> <p>(April 27 – May 3)</p>	<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.
<p>Week 8</p> <p>(May 4–May 10)</p>	<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.

<p>Week 9</p> <p>(May 11 – May 17)</p>	<ul style="list-style-type: none"> • Developed Frontend to add questions as admin. • Edit, Delete the questions.(Both backend and basic frontend).
<p>Week 10</p> <p>(May 18 – May 24)</p>	<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.

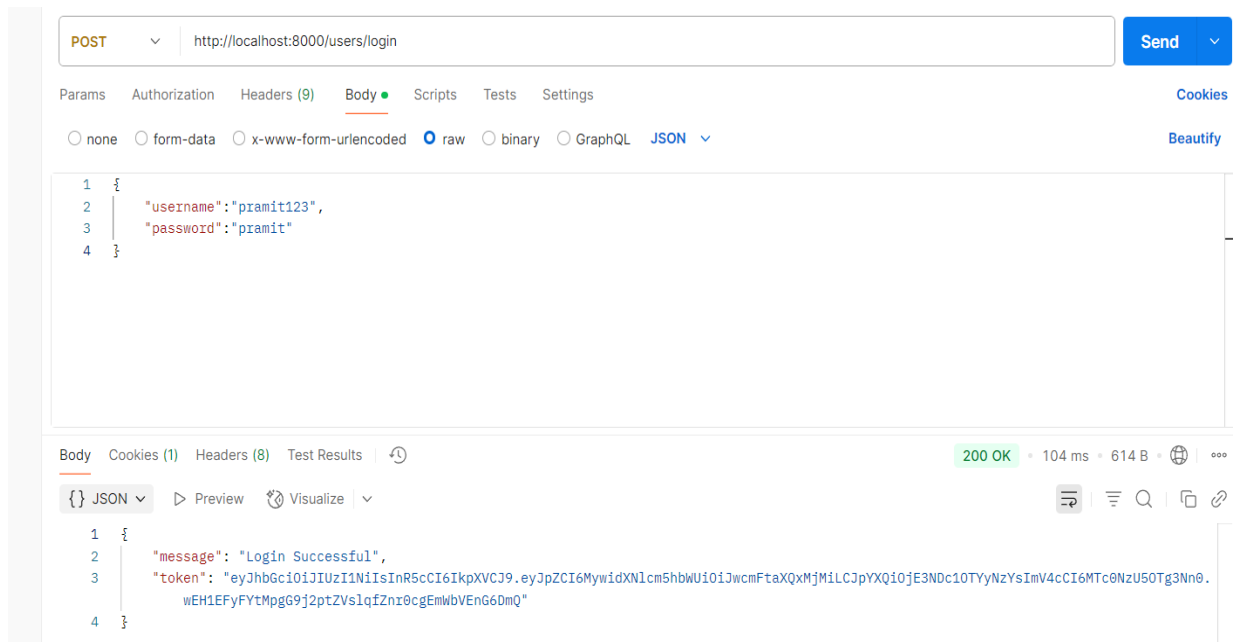


Figure 3.1: Login Postman

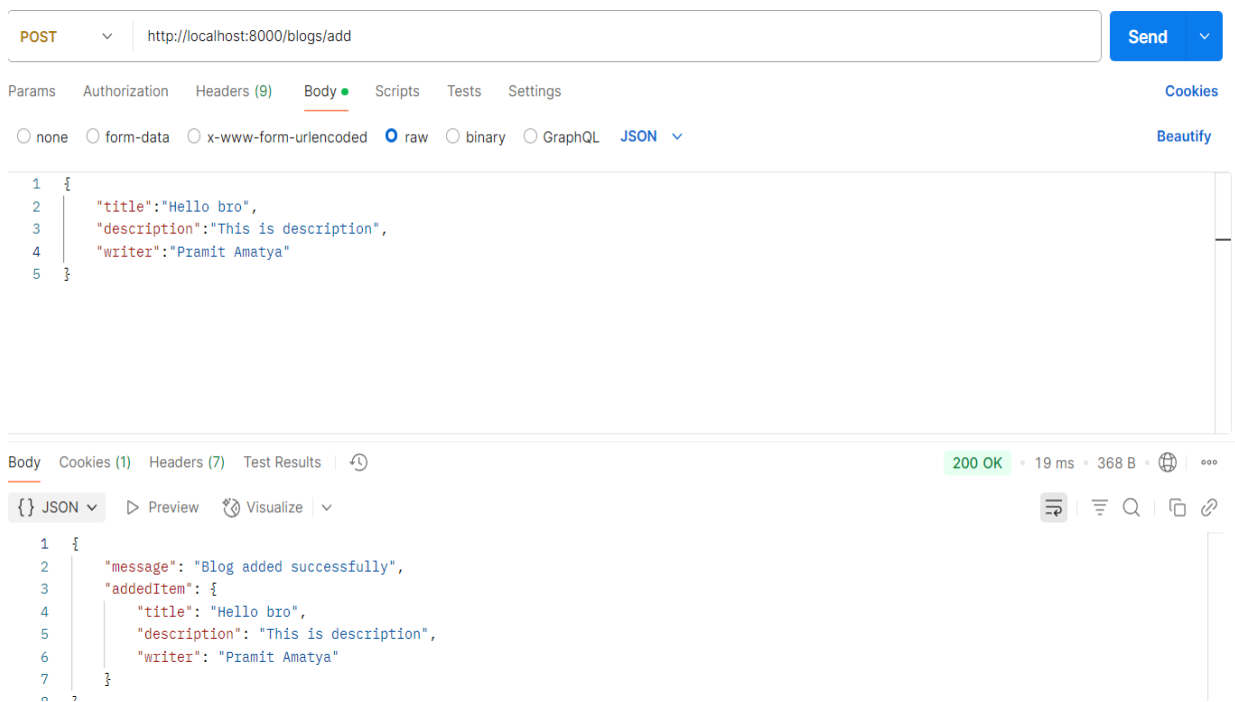


Figure 1.2: Blog Postman

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.


```

auth.proto ×
proto > auth.proto
1  syntax = "proto3";
2
3  service AuthService {
4      rpc Register (RegisterRequest) returns (RegisterResponse);
5      rpc Login (LoginRequest) returns (LoginResponse);
6  }
7
8  message RegisterRequest {
9      string username = 1;
10     string password = 2;
11 }
12
13 message RegisterResponse {
14     bool success = 1;
15 }
16
17 message LoginRequest {
18     string username = 1;
19     string password = 2;
20 }
21
22 message LoginResponse {
23     string token = 1; // JWT token
24 }

```

Figure 3.3: Auth.proto

```

JS client.js ×
grpc > JS client.js > ...
1  // grpc/client.js
2  const grpc = require('@grpc/grpc-js');
3  const protoLoader = require('@grpc/proto-loader');
4
5  // Load proto files
6  const authProtoPath = './proto/auth.proto';
7  const blogProtoPath = './proto/blog.proto';
8
9  const authPackageDef = protoLoader.loadSync(authProtoPath);
10 const blogPackageDef = protoLoader.loadSync(blogProtoPath);
11
12 // Load gRPC Packages
13 const authProto = grpc.loadPackageDefinition(authPackageDef);
14 const blogProto = grpc.loadPackageDefinition(blogPackageDef);
15
16 // Create gRPC Clients
17 const authStub = new authProto.AuthService('0.0.0.0:50051', grpc.credentials.createInsecure());
18 const blogStub = new blogProto.BlogService('0.0.0.0:50052', grpc.credentials.createInsecure());
19
20 module.exports = {
21     authStub,
22     blogStub,
23 };
24

```

Figure 3.4: gRPC client.js

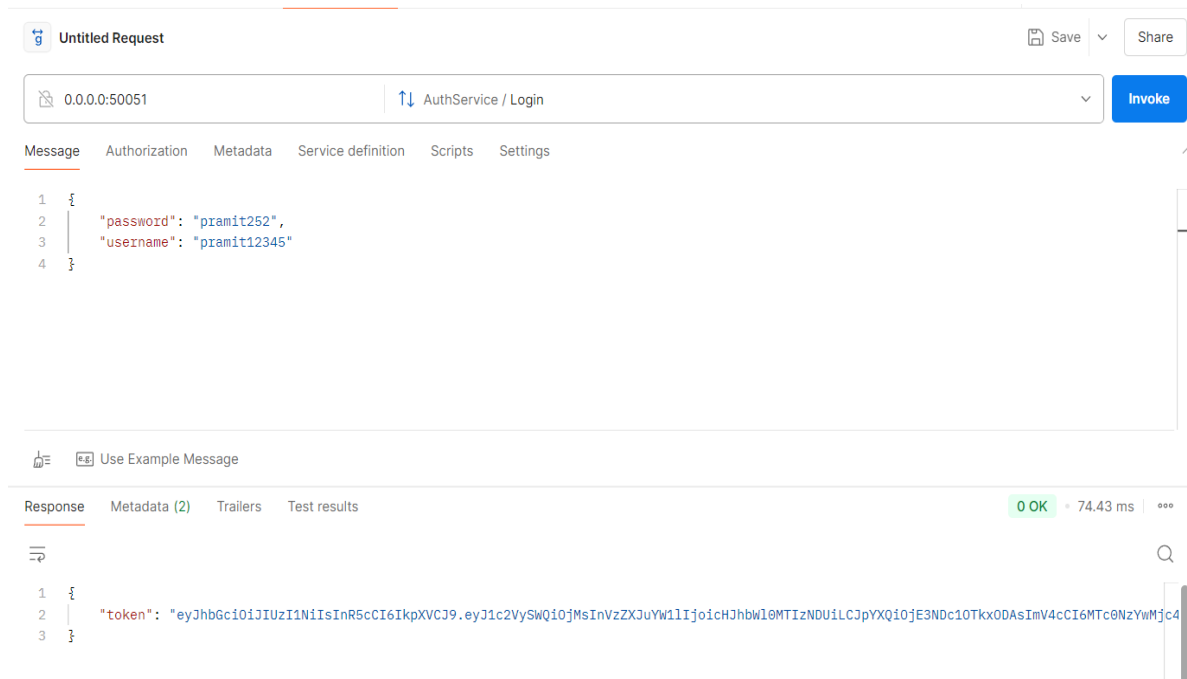


Figure 3.5: gRPC Login Postman

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.

```
exports.login = async (req, res) => {
  const { username, password } = req.body;

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

  const user = await User.findOne({ where: { username } });
  if (!user) {
    return res.status(401).json({ message: "Invalid username or password" });
  }

  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, role: user.role },
    SECRET_KEY,
    { expiresIn: "1h" }
  );

  req.session.token = token;

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

Figure 3.6: User Authentication

```

exports.getUserDetails = async (req, res) => {
  const user = await User.findOne({
    where: { id: req.user.id },
    attributes: ["name", "age", "username", "role"],
  });

  if (!user) return res.status(404).json({ message: "User not found" });

  res.status(200).json(user);
};

exports.updateUserDetails = async (req, res) => {
  const { name, age, username } = req.body;

  if (name === undefined && age === undefined && username === undefined) {
    return res
      .status(400)
      .json({ message: "At least one field is required to update" });
  }

  const user = await User.findByPk(req.user.id);
  if (!user) return res.status(404).json({ message: "User not found" });

  if (name !== undefined) {
    if (name.trim() === "") {
      return res.status(400).json({ message: "Name cannot be empty" });
    }
    user.name = name;
  }

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

```

Figure 3.7: 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.

```

JS isAdmin.js X
backend > middleware > JS isAdmin.js > ...
1 // middleware/isAdmin.js
2 const isAdmin = (req, res, next) => {
3   if (req.user && req.user.role === "admin") {
4     return next();
5   }
6   return res.status(403).json({ message: "Access denied: Admins only" });
7 };
8
9 module.exports = isAdmin;
10

```

Figure 3.8: Admin role and authorization

```

6
7 exports.addQuestion = async (req, res) => {
8   const { questionImage, question, options, correctAnswer, hint } = req.body;
9
10  if (!question || !Array.isArray(options) || options.length < 2 || !correctAnswer) {
11    return res.status(400).json({ message: "Invalid question format" });
12  }
13
14  if (!options.includes(correctAnswer)) {
15    return res.status(400).json({ message: "Correct answer must be one of the options" });
16  }
17
18  const imageToUse = questionImage?.trim() || defaultImage;
19
20  const newQuestion = await Question.create({
21    questionImage: imageToUse,
22    question,
23    options,
24    correctAnswer,
25    hint,
26  });
27
28  res.status(201).json({ message: "Question added successfully", question: newQuestion });
29 };

```

Figure 3.9: 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 4.1: 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 4.2: 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 4.3: 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.

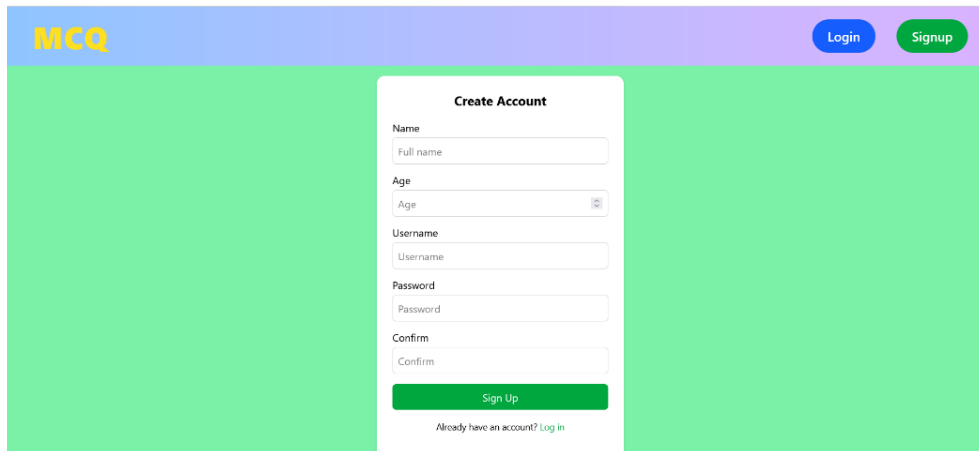
4.2 Learning Outcomes

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 image shows a web form titled "Create Account" for a platform called "MCQ". The form is centered on a light green background. It includes input fields for Name (Full name), Age, Username, Password, and Confirm. A green "Sign Up" button is at the bottom, with a link "Already have an account? Log in" below it. The top of the page has a purple header with the "MCQ" logo and "Login" and "Signup" buttons.

MCQ

[Login](#) [Signup](#)

Create Account

Name
Full name

Age
Age

Username
Username

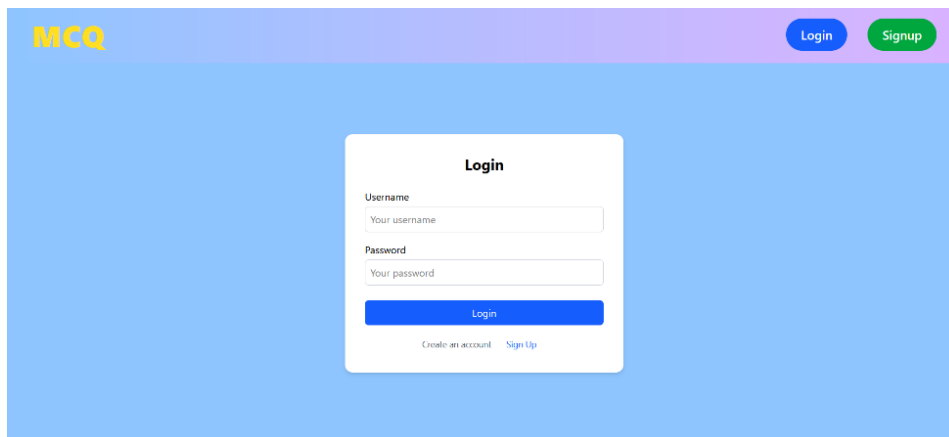
Password
Password

Confirm
Confirm

[Sign Up](#)

[Already have an account? Log in](#)

Sign in Page



The image shows a web form titled "Login" for a platform called "MCQ". The form is centered on a light blue background. It includes input fields for Username (Your username) and Password (Your password). A blue "Login" button is at the bottom, with links "Create an account" and "Sign Up" below it. The top of the page has a purple header with the "MCQ" logo and "Login" and "Signup" buttons.

MCQ

[Login](#) [Signup](#)

Login

Username
Your username

Password
Your password

[Login](#)

[Create an account](#) [Sign Up](#)

Login Page



The image shows a web page titled "Quiz Home Page" for a platform called "MCQ". The page has a dark blue background. On the left, there is a chessboard with a puzzle labeled "Q. 1". On the right, there is a section titled "Your Score: 0 / 5" and "Mate in 3". Below this, there are four buttons labeled A: Rx7+, B: Rg6+, C: Ra6+, and D: Rh6+. The top of the page has a purple header with the "MCQ" logo and a "Profile" button.

MCQ

[Profile](#)

Q. 1

Your Score: 0 / 5

Mate in 3

[A: Rx7+](#) [B: Rg6+](#)

[C: Ra6+](#) [D: Rh6+](#)

Quiz Home Page

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

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

Quiz Results

Close

Add New Question

What is this image?

ions_%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