Tribhuvan University

Institute of Science and Technology

An Internship Report

on

“Chess MCQuiz Application”

Full Stack 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

ACKNOWLEDGEMENT

I would like to express my heartfelt gratitude to all individuals who supported me throughout my internship. Focusing mainly on my mentor, **Mr. Kritan Thapa**, 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 full-stack 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

LIST OF TABLES

LIST OF ABBREVIATIONS

API – Application Programming Interface

CRUD – Create, Read, Update, Delete

CSS – Cascading Style Sheets

DB – Database

JWT– JSON Web Token

Node.js– Node JavaScript

JSON – JavaScript Object Notation

MVC – Model-View-Controller

MySQL – My Structured Query Language

UI– User Interface

REST – Representational State Transfer

SQL – Structured Query Language

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 for an interactive learning experience.
- To implement user authentication and role-based access control for secure and personalized learning.
- To integrate real-time scoring, leaderboards, and progress tracking to enhance user engagement.
- To optimize database management and API performance for seamless puzzle retrieval and response handling.
- To ensure data security and privacy through authentication, encryption, and access control measures.

1.4 SCOPE AND LIMITATIONS

1.4.1 Scope

- 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.

1.4.2 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.

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.

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

Table 2.1. 1: Organizational Details

2.2 ORGANIZATIONAL HIERARCHY

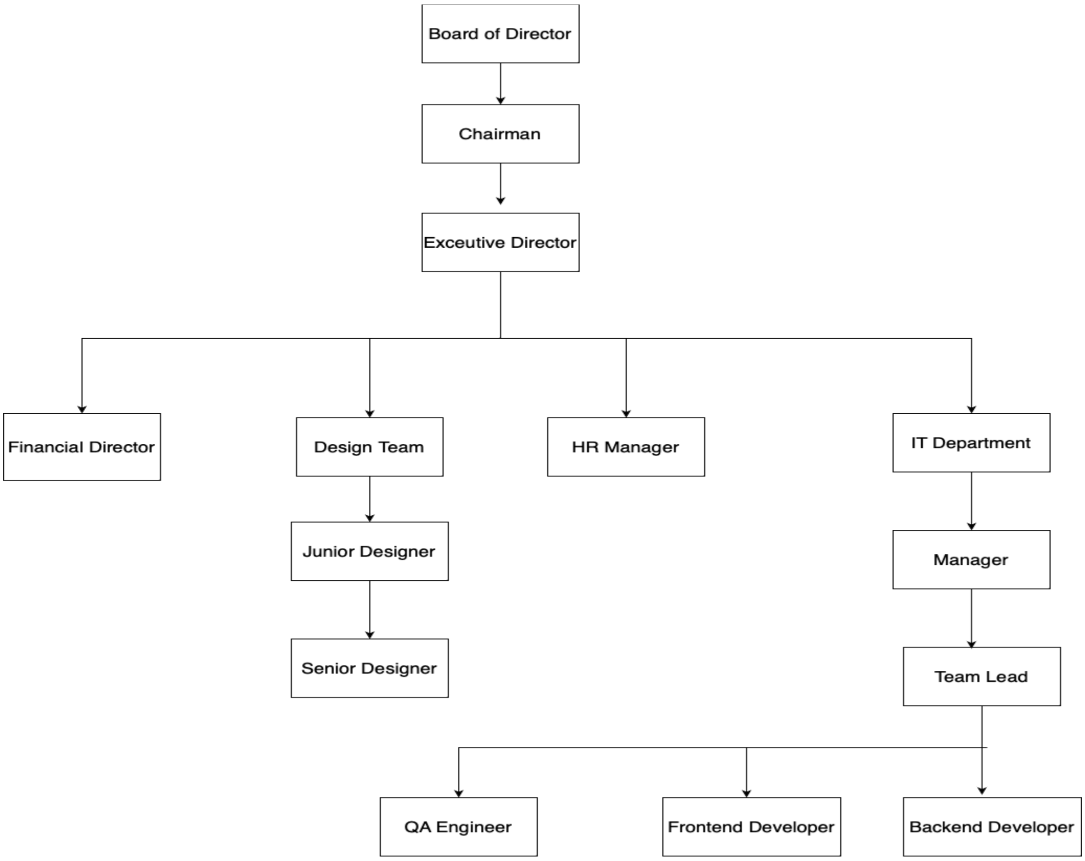


Figure 2.2. 1: Organizational Hierarchy

2.3 WORKING DOMAINS OF ORGANIZATION

Domain	Description	Key Aspects
Executive Leadership	Sets the company's overall vision, strategy and governance.	• 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	• Requirements gathering & analysis • System architecture &

	software products or platforms.	design • Coding standards & best practices • Automated testing & continuous integration
HR & Administration	Manages the employee lifecycle and keeps the office running smoothly.	• Recruitment & onboarding • Performance management & training • Payroll, benefits & compliance • Facilities, vendor & office-support services

Table 2.3. 1: Working Domains of Organization

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.

Name of Mentor	Mr. Kritan Thapa
Position	Software Engineer
Email	kritanthapa@swifttech.com
Phone	9860632193

Table 2.4. 1: Contact Details of Mentor

CHAPTER 3: INTERNSHIP ACTIVITIES

3.1 ROLES AND RESPONSIBILITIES

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

- **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

Week 1	• Learning and doing CRUD operation using expressjs and mysql • Learned Graceful Shutdown • JWT Authentication and validation
Week 2	• Implemented user authentication • Learned about middleware for session and implemented it. • Create Frontend for authentication

Week 3	• 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	• Learning Sequelize-cli, migrations and seeders. • Implementing into our project
Week 5	• 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	• 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	• Learned about session expiration in frontend • Moved to Login when session expired • Quiz result stored in database and fetched in frontend profile
Week 8	• 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.

Table 3.2. 1: Weekly Log

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 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.

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.

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

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

Table 4.4.4. 1: Test Cases for User Authentication Testing

B. 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

Table 4.4.4. 2: Test Cases for Profile Testing

C. 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

Table 4.4.4. 3: Test Cases for Admin Dashboard Testing

CHAPTER 4: CONCLUSION

The Chess Puzzle MCQuiz platform represents a meaningful step toward improving chess education through interactive, structured, and engaging puzzle-based learning. As a backend developer, my contributions focused on building a secure, scalable, and modular system that enables seamless quiz interactions, puzzle management, and performance tracking.

The project successfully achieves its primary goals of providing a platform for users to solve tactical puzzles through MCQ formats, track their progress, and receive instant feedback. The robust authentication system, structured data flow, and admin-level puzzle controls ensure a reliable and user-friendly experience. With efficient API architecture, optimized database operations, and maintainable code structure, the platform lays a strong foundation for future expansion and integration into broader chess learning ecosystems.

REFERENCES

Node.js. (2025). Node.js documentation. Retrieved March, 2025, from https://nodejs.org/en/docs/

Express.js. (2025). Express.js documentation. Retrieved March, 2025, from https://expressjs.com/en/starter/installing.html

MySQL. (2025). MySQL reference manual. Retrieved March, 2025, from https://dev.mysql.com/doc/

Sequelize. (2025). Sequelize ORM documentation. Retrieved March, 2025, from https://sequelize.org/docs/v6/

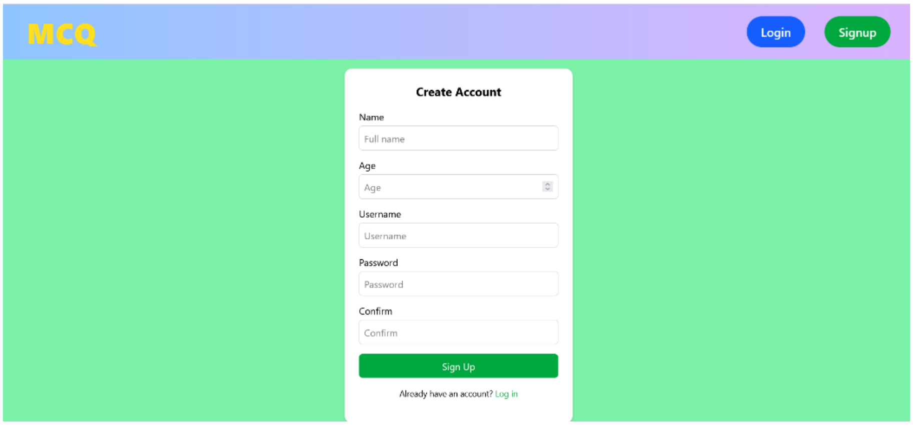
JSON Web Tokens (JWT). (2025). JWT.io Introduction. Retrieved March, 2025, from https://jwt.io/introduction

Postman. (2025). Postman API Platform Documentation. Retrieved March, 2025, from https://learning.postman.com/docs/

GitHub Docs. (2025). GitHub documentation. Retrieved March, 2025, from https://docs.github.com/en

Visual Studio Code. (2025). VS Code Documentation. Retrieved March, 2025, from https://code.visualstudio.com/docs

APPENDICES



Create Account

Name

Full name

Age

Age

Username

Username

Password

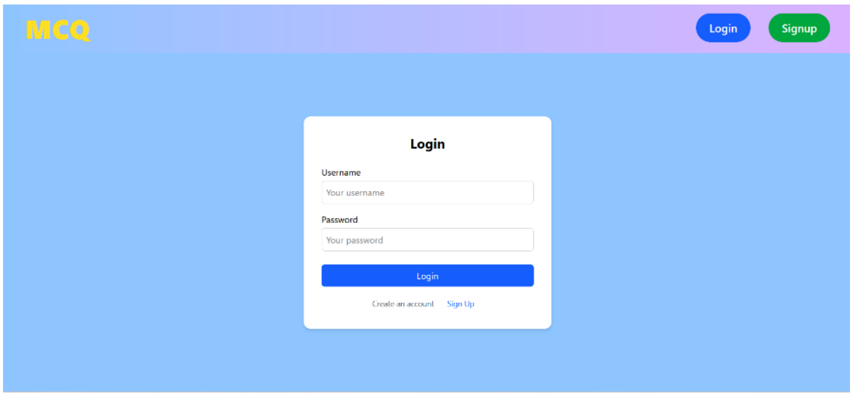
Password

Confirm

Confirm

Already have an account? Log in

Sign in Page



Login

Username

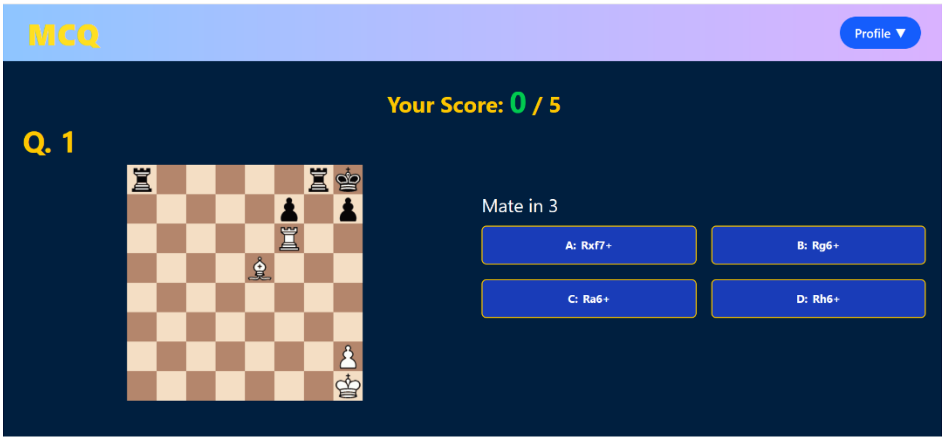
Your username

Password

Your password

Create an account Sign Up

Login Page

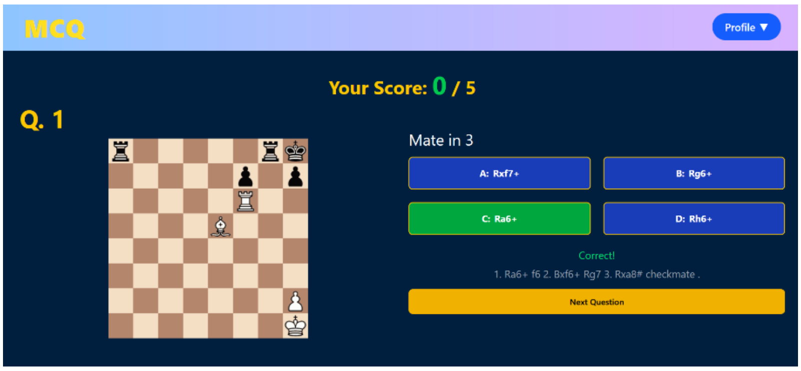


Your Score: 0 / 5

Q. 1

Mate in 3

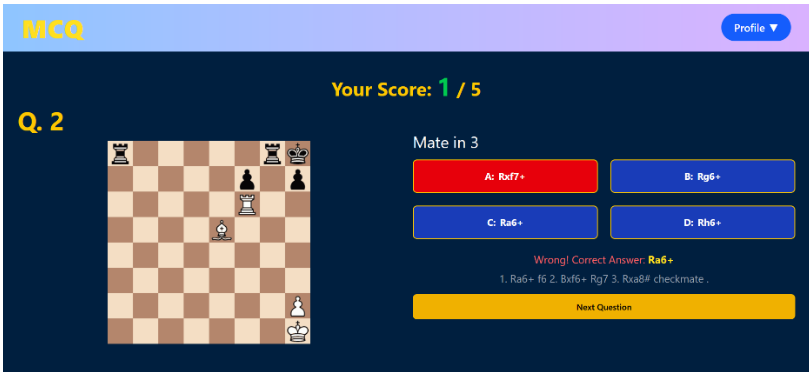
Quiz Home Page



Correct!

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

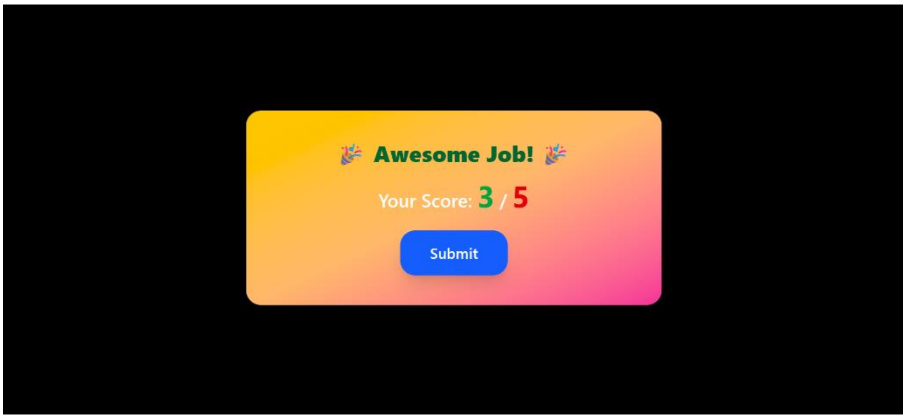
Quiz Page Correct Answer



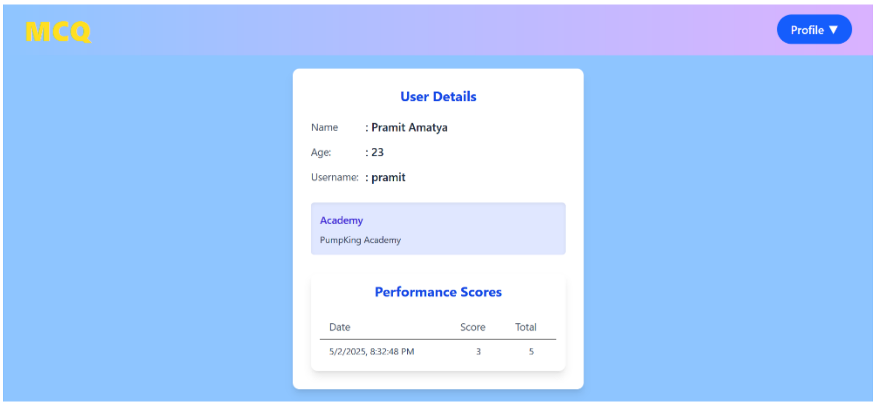
Wrong! Correct Answer: Ra6+

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

Quiz Page Wrong Answer



Submit Component



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