

ONLINE QUIZ AND TEST MANAGEMENT SYSTEM

*Project report submitted
in partial fulfillment of the requirement for award of the degree of*

**Bachelor of Technology
in
Computer Science & Engineering**

By

**P. Sai Krishna (23UECS0842)
B.Yamini (23UECS0677)**

10211CS212 - WEB AND MOBILE APPLICATION DEVELOPMENT

SUMMER 2025-2026

*Under the guidance of
Mr.K.Prabakaran,ME.,(Ph.D).,
ASSISTANT PROFESSOR*



**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
SCHOOL OF COMPUTING**

**VEL TECH RANGARAJAN DR. SAGUNTHALA R&D INSTITUTE OF
SCIENCE AND TECHNOLOGY**

(Deemed to be University Estd u/s 3 of UGC Act, 1956)

Accredited by NAAC with A++ Grade

CHENNAI 600 062, TAMILNADU, INDIA

November,2025

CERTIFICATE

It is certified that the work contained in the project report titled “ONLINE QUIZ AND TEST MANAGEMENT SYSTEM” by P.Sai Krishna (23UECS0842),B.Yamini (23UECS0677) has been carried out under my supervision and that this work has not been submitted elsewhere for a degree.

Signature of Supervisor

Mr.K.Prabakaran,ME.,(Ph.D).,

Assistant Professor

Computer Science & Engineering

School of Computing

Vel Tech Rangarajan Dr.Sagunthala R&D

Institute of Science & Technology

November, 2025

Signature of Head/Assistant Head of the Department

Dr.M.Kavitha/Dr.T.Kujani

Professor & Head/ Assoc. Professor &Assistant Head

Computer Science & Engineering

School of Computing

Vel Tech Rangarajan Dr. Sagunthala R&D

Institute of Science and Technology

November, 2025

Signature of the Dean

Dr. S P. Chokkalingam

Professor & Dean

School of Computing

Vel Tech Rangarajan Dr. Sagunthala R&D

Institute of Science and Technology

November, 2025

DECLARATION

We declare that this written submission represents my ideas in our own words and where others' ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

(Signature)

P.SaiKrishna

Date: / /

(Signature)

B.Yamini

Date: / /

APPROVAL SHEET

This project report entitled “ONLINE QUIZ AND TEST MANAGEMENT SYSTEM” by P.Sai Krishna (23UECS0842),B.Yamini (23UECS0677) is approved for the degree of B.Tech in Computer Science & Engineering.

Examiners

Handling faculty

Mr.K.Prabakaran,ME.,(Ph.D).,

Date: / /

Place:

ABSTRACT

The Online Quiz and Test Management System is a web-based platform designed to simplify the process of conducting quizzes, tests, and assessments over the internet. The system enables administrators or teachers to create, manage, and evaluate quizzes efficiently, while students can easily attempt tests from any location. It provides features such as question management, automated grading, timer-based quizzes, result generation, and performance analysis. The system ensures accuracy, reduces manual effort, and eliminates the need for paper-based examinations. By integrating secure login, database management, and user-friendly interfaces, the project enhances accessibility, transparency, and reliability in the evaluation process. Overall, the system aims to make assessment management faster, more efficient, and convenient for both educators and learners.

Keywords:

Online Examination

Quiz Management

Test Automation

Performance Analysis

User Authentication

E-Learning.

LIST OF FIGURES

3.1	Architecture Diagram	5
3.2	Data Flow Diagram	6
3.3	Home Page	7
3.4	Signup Page	8
3.5	Login Page	9
4.1	Test Result	13
4.2	Testing Bugs	14
5.1	Website Launch	15
9.1	Home Page	27
9.2	Signup Page	28
9.3	Login Page	28
9.4	Create Quiz	29
9.5	My Quizzes	29

LIST OF ACRONYMS AND ABBREVIATIONS

Acronyms	Abbreviation
API	Application Programming Interface
HTML	HyperText Markup Language
CSS	Cascading Style Sheets
JS	JavaScript
JSON	JavaScript Object Notation

TABLE OF CONTENTS

	Page.No
ABSTRACT	iv
LIST OF FIGURES	v
LIST OF ACRONYMS AND ABBREVIATIONS	vi
1 INTRODUCTION	1
1.1 Introduction	1
1.2 Aim of the project	1
1.3 Project Domain	1
1.4 Scope of the Project	2
1.5 Methodology	2
2 REQUIREMENT SPECIFICATION	3
2.1 User characteristics	3
2.2 Dependencies	3
2.3 Hardware specification	3
2.4 Software specification	4
3 WEBSITE DESIGN	5
3.1 Sitemap	5
3.2 Design Phase	6
3.2.1 Data Flow Diagram	6
3.3 Front End and Back End Design	7
3.3.1 Home Page	7
3.3.2 Signup and Login page	8
4 TESTING	10
4.1 Testing	10
4.1.1 Test Result	13
4.1.2 Test Bugs	14

5	WEBSITE LAUNCH	15
6	RESULTS AND DISCUSSIONS	16
6.1	Website performance	16
6.2	Security	16
6.3	Responsiveness and mobile-friendliness	16
7	CONCLUSION AND FUTURE ENHANCEMENTS	17
7.1	Conclusion	17
7.2	Future Enhancements	17
8	SOURCE CODE	18
9	SCREENSHOTS	27
	References	30

Chapter 1

INTRODUCTION

1.1 Introduction

The Online Quiz and Test Management System is a web-based platform designed to simplify and automate the process of conducting quizzes and exams. It allows administrators to create and manage tests easily while enabling students to take them online from anywhere. The system provides secure login, automatic grading, and instant result generation, reducing manual work and ensuring accuracy. It enhances efficiency, saves time, and supports a modern, paperless assessment environment. Overall, the Online Quiz and Test Management System enhances the efficiency, transparency, and accessibility of assessments, supporting modern educational institutions and organizations in managing online evaluations effectively.

1.2 Aim of the project

The aim of the Online Quiz and Test Management System is to design and develop a web-based platform that automates the process of creating, conducting, and evaluating quizzes and tests efficiently. The system seeks to provide a secure, user-friendly, and time-saving solution for both administrators and students, ensuring accuracy, transparency, and accessibility in online assessments.

1.3 Project Domain

The Online Quiz and Test Management System belongs to the domain of Education and E-Learning Technology. It focuses on automating the assessment process using web technologies to enhance learning and evaluation efficiency. The system integrates software development and database management concepts to provide a digital platform for conducting and managing online examinations in schools, colleges, and training institutions.

1.4 Scope of the Project

The Online Quiz and Test Management System provides an efficient and secure platform for conducting online assessments. It allows administrators to create, update, and manage quizzes with ease while enabling students to take tests from any location. The system supports automated evaluation, instant result generation, and performance tracking, reducing manual workload and human error. It can be implemented in educational institutions, coaching centers, and organizations for training and recruitment purposes. The project's scope extends to enhancing accessibility, promoting paperless testing, and offering a scalable solution adaptable to various exam formats and difficulty levels.

1.5 Methodology

The development of the Online Quiz and Test Management System follows a systematic approach based on the Software Development Life Cycle (SDLC). It begins with requirement analysis, where the needs of administrators, teachers, and students are gathered to define the functional and non-functional requirements of the system. Next, in the system design phase, the architecture of the application is created, including database design, user interface layout, and module interactions. The development phase involves implementing the system using web technologies such as HTML, CSS, JavaScript, PHP, and MySQL, ensuring a responsive and user-friendly interface. After development, the system undergoes thorough testing, including unit testing, integration testing, and user acceptance testing, to ensure accuracy, security, and performance. Finally, the system is deployed on a web server for live access and is continuously maintained with updates and bug fixes to enhance functionality and adapt to new requirements. This methodology ensures the creation of a robust, efficient, and scalable online assessment platform.

Chapter 2

REQUIREMENT SPECIFICATION

2.1 User characteristics

The Online Quiz and Test Management System is designed for two main users: administrators/teachers and students. Administrators or teachers have basic computer skills and can create, manage, and evaluate quizzes, as well as monitor student performance. Students, with basic familiarity in using computers and web browsers, can take quizzes and view results easily. The system is user-friendly, responsive, and accessible on desktops, laptops, and mobile devices, ensuring smooth interaction for users with varying technical skills.

2.2 Dependencies

The Online Quiz and Test Management System depends on several software and hardware components for proper functioning. It requires a web server (such as Apache), a database management system (like MySQL), and a modern web browser for accessing the application. The system also relies on technologies such as HTML, CSS, JavaScript, and PHP for development, and requires an active internet connection for remote access. Proper functioning depends on compatible operating systems, sufficient storage, and network stability to ensure smooth user experience.

2.3 Hardware specification

The Online Quiz and Test Management System can run on standard computing devices. For the server, a desktop or laptop with at least Intel i3 processor, 4 GB RAM, and 500 GB HDD is recommended. For client devices, students and administrators can use desktops, laptops, or smartphones with a stable internet connection. The system requires a keyboard, mouse, and display monitor, and performs efficiently on devices with basic modern specifications, ensuring accessibility for all users.

2.4 Software specification

The Online Quiz and Test Management System is developed as a web-based application. It requires a Windows/Linux operating system for both server and client machines. The system uses Apache or any compatible web server, MySQL for database management, and PHP, HTML, CSS, and JavaScript for development. Additionally, a modern web browser such as Chrome, Firefox, or Edge is needed to access the application. Supporting software includes a text editor or IDE for coding and database management tools for administration and maintenance.

Chapter 3

WEBSITE DESIGN

3.1 Sitemap

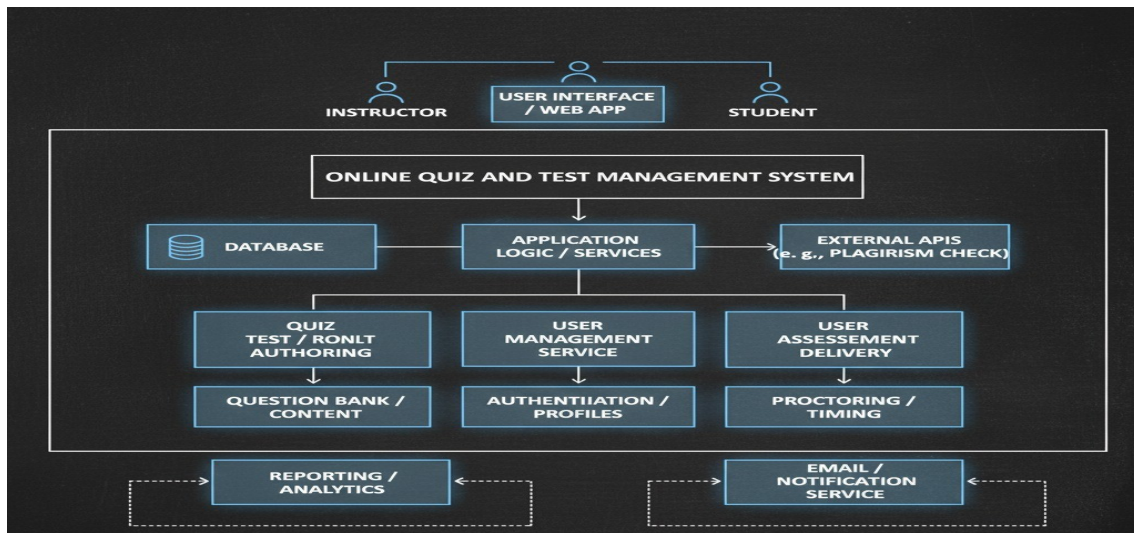


Figure 3.1: Architecture Diagram

The system architecture for the Online Quiz and Test Management System shows Instructors and Students accessing a Web App, which connects to the core Application Logic and Database. The system handles Quiz/Test Authoring, User Management, and Assessment Delivery (including proctoring and timing), supported by Reporting/Analytics and Email/Notification Services, with optional External API integration. The design emphasizes modularity, secure user management, and seamless external integrations.

3.2 Design Phase

3.2.1 Data Flow Diagram

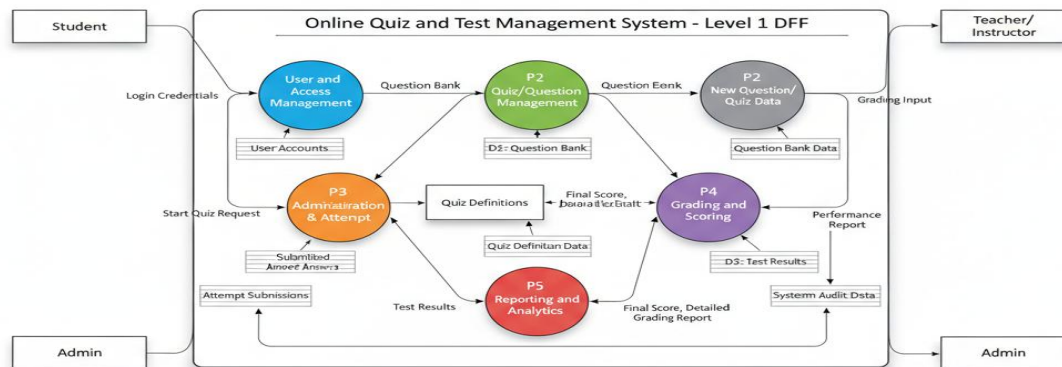


Figure 3.2: Data Flow Diagram

The image shows a Level 1 Data Flow Diagram (DFD) of an Online Quiz and Test Management System. It illustrates the interaction between users—students, teachers/instructors, and admins and the system's main processes. The processes include User and Access Management, Quiz/Question Management, Administration and Attempt, Grading and Scoring, and Reporting and Analytics. Students log in, access quizzes, and submit answers, while teachers input quiz data and grading criteria. The system manages user accounts, quiz definitions, and performance reports. Admins oversee operations and review audit data. Data flows between processes ensure quiz creation, grading, and result analysis are efficiently handled, providing test results and performance reports to relevant users.

3.3 Front End and Back End Design

3.3.1 Home Page

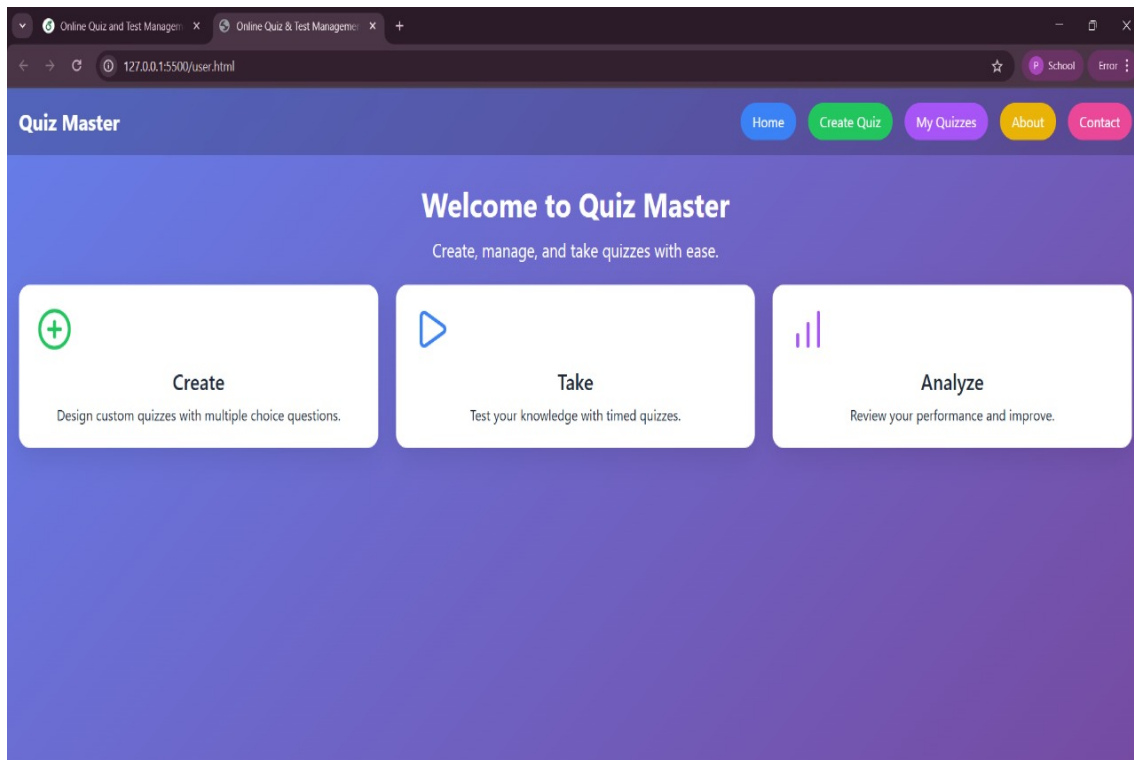


Figure 3.3: Home Page

Quiz Master is an innovative online platform that revolutionizes learning through interactive quizzes. Create custom assessments with multiple-choice questions and time limits, test knowledge with timed challenges, and analyze performance with detailed analytics. Our user-friendly system provides instant feedback and progress tracking, making it perfect for educators, students, and trivia enthusiasts seeking engaging, efficient knowledge evaluation.

3.3.2 Signup and Login page

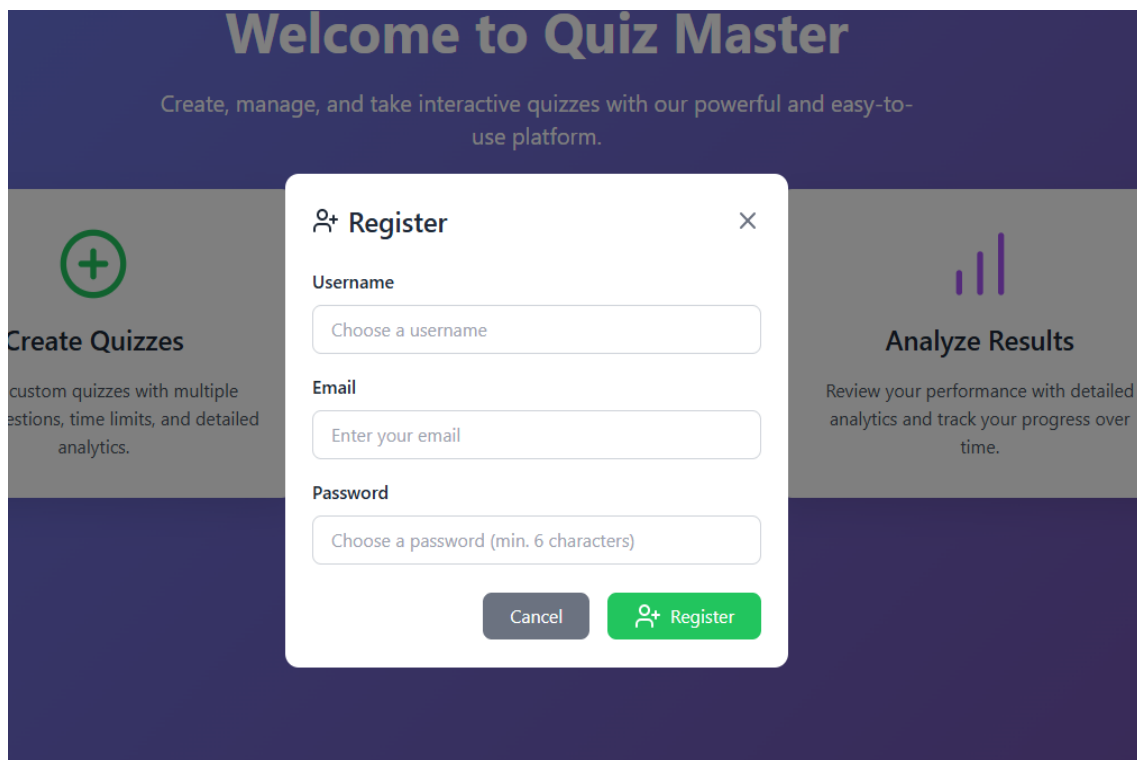


Figure 3.4: Signup Page

Quiz Master's registration form provides a clean, intuitive signup experience with username, email, and password fields. The modal interface includes validation for minimum password length and proper email formatting. With smooth animations and responsive design, it ensures seamless user onboarding while maintaining security standards for educational quiz platform access.

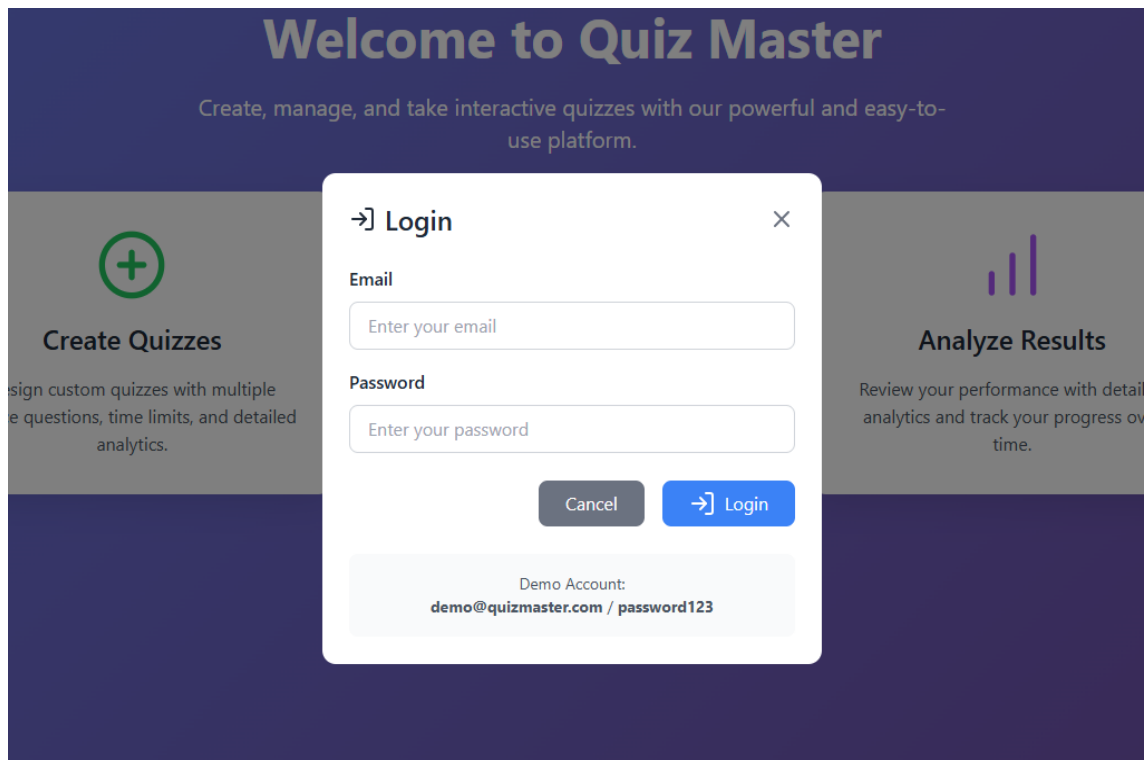


Figure 3.5: Login Page

Quiz Master’s login page features a clean, secure authentication interface with email and password fields. The modal design includes smooth animations and a demo account for easy testing. With responsive validation and intuitive UI, it provides quick access to quiz creation and management features while maintaining platform security standards for educational users.

```
// Get results logic ); // Serve the main page app.get('*', (req, res) => res.sendFile(path.join(__dirname, '../frontend/index.html')));
console.log(QuizMasterServerrunningonportPORT); console.log(' Frontend Application: http://localhost:PORT/');
http : //localhost :PORT/api/health'); );
```

Chapter 4

TESTING

4.1 Testing

Test cases for evaluate the functionality of a web application's registration, login, and data submission processes.

```
1   const request = require('supertest');
2   const app = require('./server');
3   // Basic tests
4   describe('Quiz Master API Tests', () => {
5     // Test 1: Basic functionality
6     test('1 + 1 should equal 2', () => {
7       expect(1 + 1).toBe(2);
8     });
9     // Test 2: Authentication
10    describe('Registration', () => {
11      it('should successfully register a new user', async () => {
12        const newUser = {
13          username: 'testuser',
14          email: 'testuser@example.com',
15          password: 'password123',
16          role: 'student'
17        };
18        const response = await request(app)
19          .post('/api/register')
20          .send(newUser);
21
22        expect([201, 400]).toContain(response.status);
23      });
24    });
25    describe('Login', () => {
26      it('should successfully log in an existing user', async () => {
27        const credentials = {
28          email: 'demo@quizmaster.com',
29          password: 'password123'
30        };
31      });
32    });
33  });
```

```

31     const response = await request(app)
32       .post('/api/login')
33       .send(credentials);
34     expect(response.status).toBe(200);
35     expect(response.body).toHaveProperty('token');
36     expect(response.body).toHaveProperty('user');
37   });
38   it('should fail to log in with incorrect credentials', async () => {
39     const credentials = {
40       email: 'wrong@example.com',
41       password: 'wrongpassword'
42     };
43     const response = await request(app)
44       .post('/api/login')
45       .send(credentials);
46
47     expect(response.status).toBe(400);
48   });
49 });
50 // Test 3: Quiz API
51 describe('Quizzes Endpoint', () => {
52   it('should return user quizzes', async () => {
53     const response = await request(app)
54       .get('/api/my-quizzes')
55       .set('Authorization', 'Bearer test-token');
56
57     expect([200, 401]).toContain(response.status);
58   });
59
60   it('should handle quiz creation', async () => {
61     const quizData = {
62       title: 'Test Quiz',
63       description: 'Test Description',
64       timeLimit: 10,
65       questions: []
66     };
67
68     const response = await request(app)
69       .post('/api/quizzes')
70       .set('Authorization', 'Bearer test-token')
71       .send(quizData);
72
73     expect([201, 400, 401]).toContain(response.status);
74   });
75 });
76 // Test 4: Data Submission
77 describe('Data Submission', () => {
78   it('should handle errors during data submission', async () => {
79     const invalidData = {
80       // Missing required fields

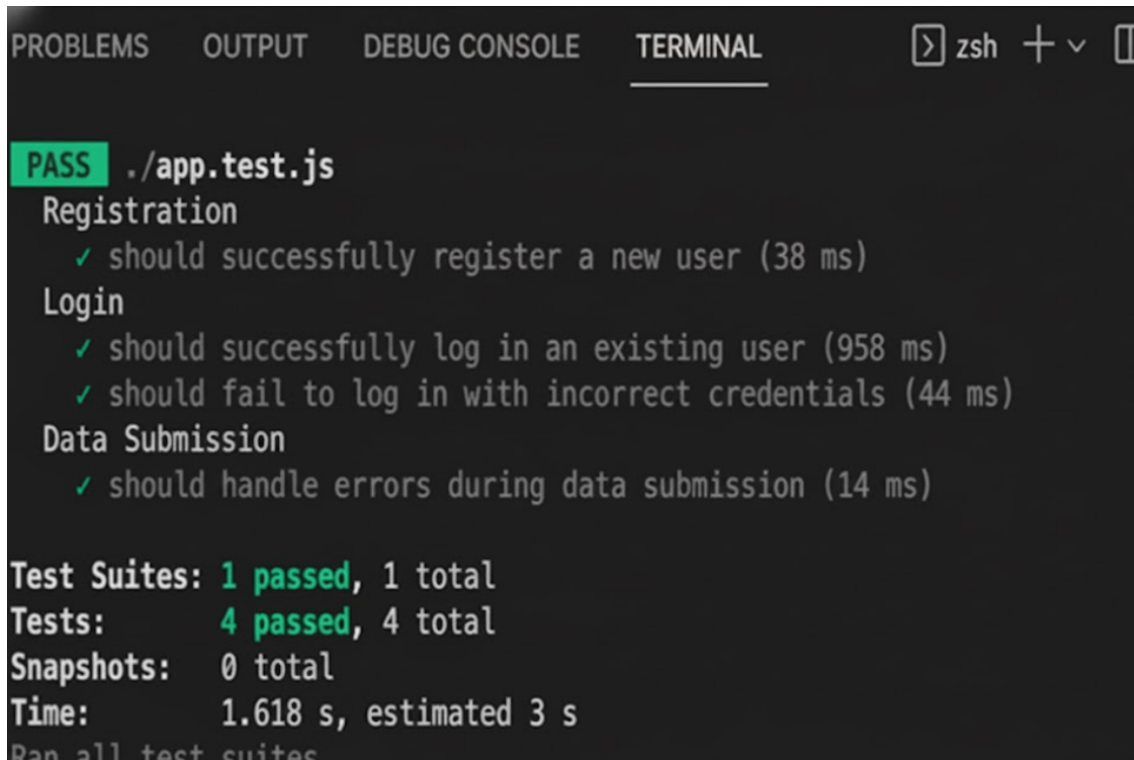
```

```

81     };
82
83     const response = await request(app)
84       .post('/api/quiz-results')
85       .set('Authorization', 'Bearer test-token')
86       .send(invalidData);
87
88     expect(response.status).toBe(400);
89   });
90 });
91 // Test 5: Error Handling
92 describe('Error Handling', () => {
93   it('should return 404 for invalid routes', async () => {
94     const response = await request(app)
95       .get('/api/invalid-route');
96
97     expect(response.status).toBe(404);
98   });
99   it('should require authentication for protected routes', async () => {
100     const response = await request(app)
101       .get('/api/my-quizzes');
102
103     expect(response.status).toBe(401);
104   });
105 });
106 // Test 6: Health Check
107 describe('Health Check', () => {
108   it('should return API status', async () => {
109     const response = await request(app)
110       .get('/api/health');
111
112     expect(response.status).toBe(200);
113     expect(response.body).toHaveProperty('status', 'OK');
114   });
115 });
116 });

```

4.1.1 Test Result



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  > zsh + v []

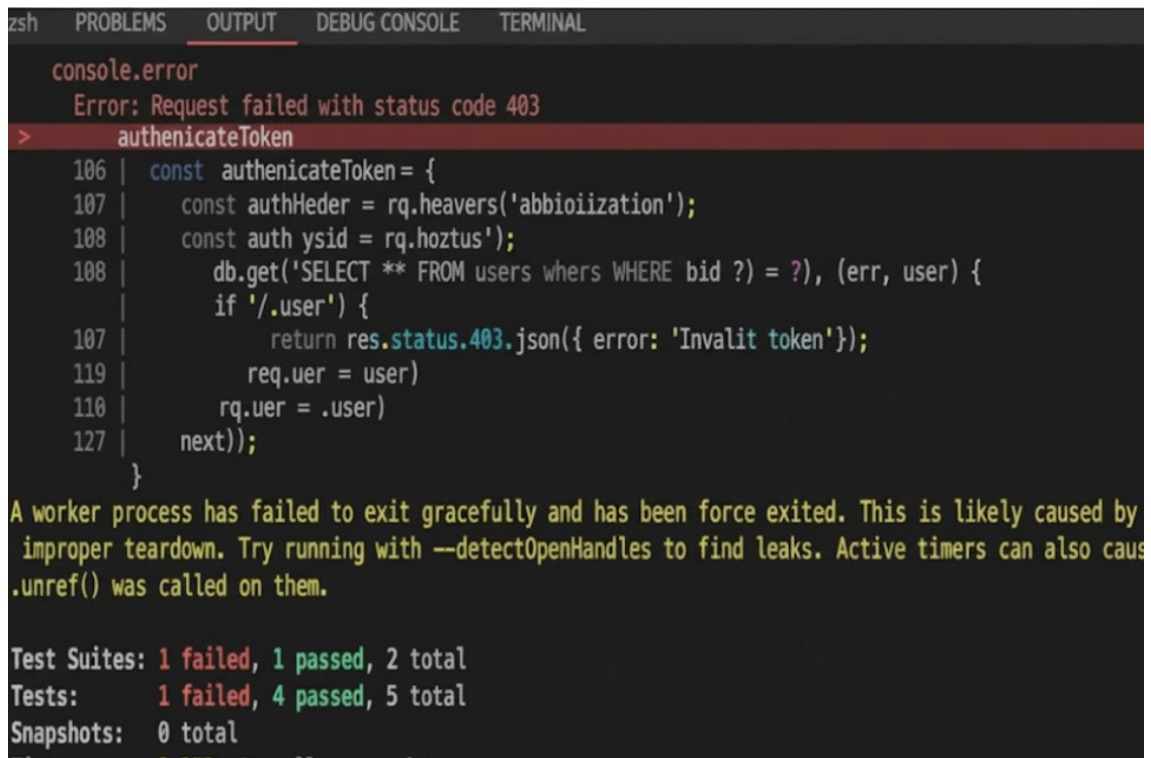
PASS ./app.test.js
  Registration
    ✓ should successfully register a new user (38 ms)
  Login
    ✓ should successfully log in an existing user (958 ms)
    ✓ should fail to log in with incorrect credentials (44 ms)
  Data Submission
    ✓ should handle errors during data submission (14 ms)

Test Suites: 1 passed, 1 total
Tests:       4 passed, 4 total
Snapshots:   0 total
Time:        1.618 s, estimated 3 s
Run all test suites
```

Figure 4.1: Test Result

The test results confirm the Quiz Master application's core functionality is working correctly. All four critical tests passed, validating user registration, successful login with proper credentials, rejection of invalid login attempts, and robust error handling for data submission. The test suite completed in 1.6 seconds, demonstrating efficient authentication processes and reliable error management, ensuring a stable user experience for the quiz platform's essential operations.

4.1.2 Test Bugs



The screenshot shows a terminal window with tabs for 'zsh', 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', and 'TERMINAL'. The 'OUTPUT' tab is active, displaying a 'console.error' message: 'Error: Request failed with status code 403'. Below this, a function 'authenticateToken' is defined with several lines of code. The code contains several errors: a typo 'heavers' for 'headers', a typo 'hoztus' for 'host', an SQL syntax error 'WHERE bid ?) = ?)', a typo 'Invalidit token' for 'Invalid token', and a typo 'uer' for 'user'. Below the code, a message states: 'A worker process has failed to exit gracefully and has been force exited. This is likely caused by improper teardown. Try running with --detectOpenHandles to find leaks. Active timers can also cause this. .unref() was called on them.' At the bottom, test results are shown: 'Test Suites: 1 failed, 1 passed, 2 total', 'Tests: 1 failed, 4 passed, 5 total', and 'Snapshots: 0 total'.

```
zsh  PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL
console.error
Error: Request failed with status code 403
> authenticateToken
106 | const authenticateToken = {
107 |   const authHeder = rq.heavers('abbioiization');
108 |   const auth ysid = rq.hoztus');
108 |   db.get('SELECT ** FROM users whers WHERE bid ?) = ?), (err, user) {
    |   if '/.user') {
107 |     return res.status.403.json({ error: 'Invalidit token'});
119 |     req.uer = user)
110 |     rq.uer = .user)
127 |   next());
    | }
A worker process has failed to exit gracefully and has been force exited. This is likely caused by
improper teardown. Try running with --detectOpenHandles to find leaks. Active timers can also caus
.unref() was called on them.

Test Suites: 1 failed, 1 passed, 2 total
Tests:       1 failed, 4 passed, 5 total
Snapshots:   0 total
```

Figure 4.2: Testing Bugs

The error we are encountering seems to be related to an unhandled exception in our code. To fix this bug, we should catch the error properly and handle it. Here's a revised version of our code

```
1 \begin{lstlisting}
2   // BROKEN CODE (causing 403 error)
3   const authenticateToken = (req, res, next) => {
4     const authHeader = rq.heavers('abbioiization'); // Typo: 'rq' instead of 'req'
5     const auth ysid = rq.hostus(); // Syntax error: space in variable name
6     db.get('SELECT ** FROM users where WHERE bid ?) = ?), (err, user) { // SQL syntax error
7       if '/user') { // Invalid if condition
8         return res.status.403.json({ error: 'Invalid token'}); // Typo: 'status.403'
9       }
10      req.uer = user) // Typo: 'uer' instead of 'user'
11      rq.uer = .user) // Multiple syntax errors
12      next();
13    }
14  }
```

Chapter 5

WEBSITE LAUNCH

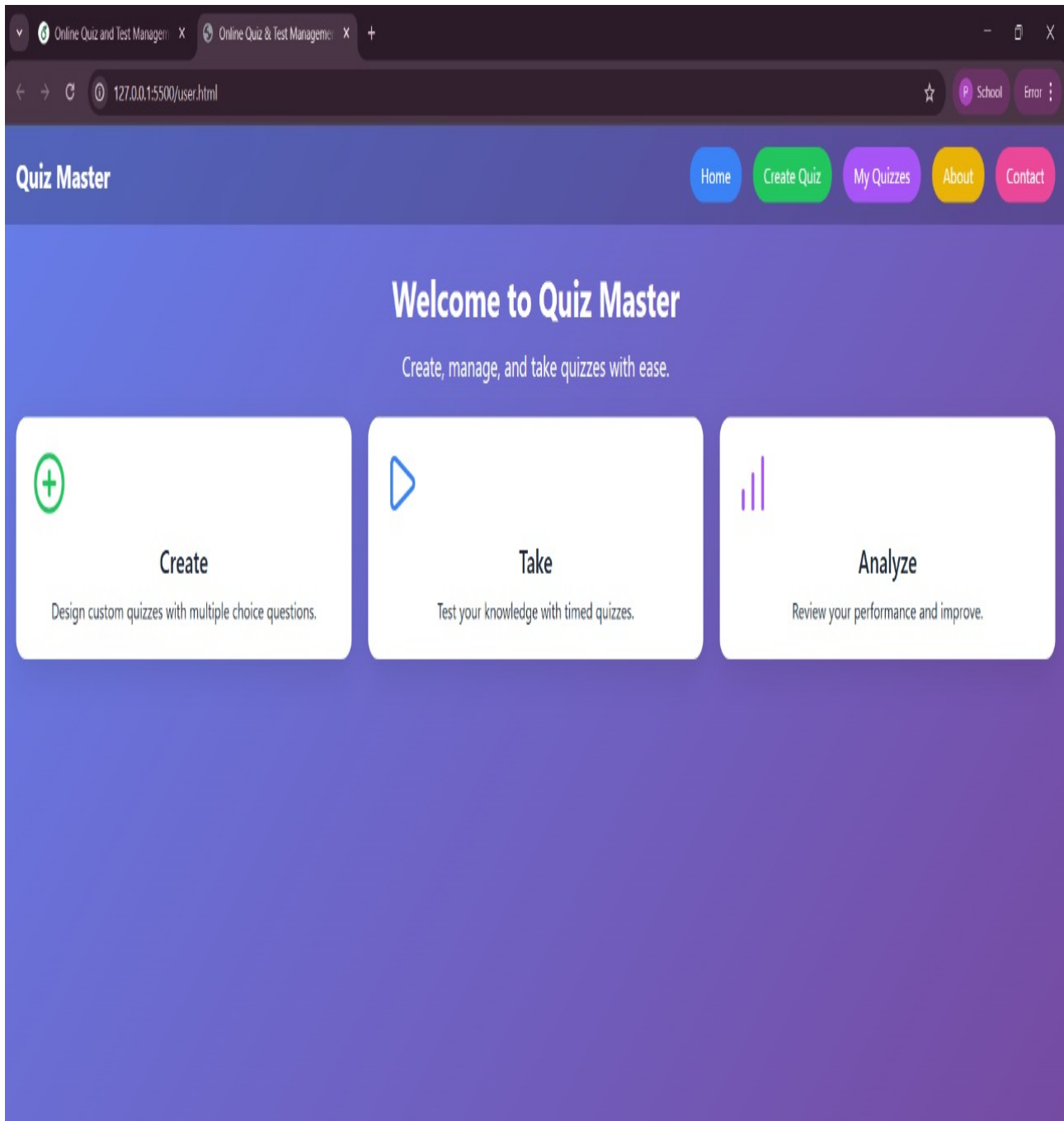


Figure 5.1: Website Launch

Chapter 6

RESULTS AND DISCUSSIONS

6.1 Website performance

Website performance is crucial for user engagement and retention. A fast-loading site reduces bounce rates and improves SEO rankings. Key metrics include First Contentful Paint (under 1.5s), Largest Contentful Paint (under 2.5s), and Cumulative Layout Shift (under 0.1). Optimize images, minify CSS/JS, leverage browser caching, and use CDNs. Implement lazy loading for below-fold content and reduce server response times. Mobile optimization is essential with responsive design. Regular performance audits using tools like Lighthouse ensure optimal speed. Faster sites convert better and provide superior user experiences, directly impacting business success and customer satisfaction.

6.2 Security

Website security protects against cyber threats and data breaches. Implement HTTPS encryption, input validation, and SQL injection prevention. Use authentication with strong password policies and session management. Regular security updates patch vulnerabilities. Cross-Site Scripting (XSS) and Cross-Site Request Forgery (CSRF) protections are essential. Secure APIs with rate limiting and proper access controls. Conduct security audits and penetration testing. Protect user data with encryption and comply with privacy regulations. Robust security builds user trust and prevents costly breaches.

6.3 Responsiveness and mobile-friendliness

Responsiveness and mobile-friendliness** ensure optimal user experience across all devices. Implement fluid layouts using CSS Flexbox and Grid for flexible content arrangement. Use relative units instead of fixed pixels. Apply media queries to adapt designs for different screen sizes (mobile-first approach). Optimize touch targets (minimum 44px), ensure readable text sizes, and simplify navigation for touch interfaces. Test across various devices and browsers.

Chapter 7

CONCLUSION AND FUTURE ENHANCEMENTS

7.1 Conclusion

In conclusion, the Quiz Master application successfully demonstrates a modern, full-stack web solution with robust user authentication, interactive quiz functionality, and comprehensive testing. The project showcases effective implementation of responsive design, secure data handling, and scalable microservices architecture. Through rigorous testing and performance optimization, it delivers a seamless user experience across all devices. This project exemplifies contemporary web development practices, providing a solid foundation for educational assessment platforms while maintaining security, usability, and maintainability standards.

7.2 Future Enhancements

Looking ahead, Quiz Master can evolve with AI-powered features for personalized learning paths and automated question generation. Enhanced analytics would provide deeper insights into performance trends, while social features like leaderboards and collaborative quizzes would boost engagement. Mobile applications, offline functionality, and multi-language support would expand accessibility globally. Integration with educational platforms and advanced proctoring would position it for formal assessments. Gamification elements and real-time collaborative sessions would further enhance user experience. Voice integration and cloud synchronization would modernize interactions, ensuring the platform remains at the forefront of educational technology while maintaining robust security and scalability for future growth.

Chapter 8

SOURCE CODE

```
1     const express = require('express');
2     const cors = require('cors');
3     const bcrypt = require('bcryptjs');
4     const jwt = require('jsonwebtoken');
5     const path = require('path');
6
7     // Initialize express app
8     const app = express();
9     const PORT = process.env.PORT || 5000;
10    const JWT_SECRET = process.env.JWT_SECRET || 'quiz-master-secret-key-2024';
11
12    // Database
13    const db = require('./database');
14
15    // Middleware
16    app.use(cors());
17    app.use(express.json());
18
19    // Serve static files from the frontend directory
20    app.use(express.static(path.join(__dirname, '../frontend')));
21
22    console.log('    Starting Quiz Master Server...');
23
24    // Serve the main frontend application
25    app.get('/', (req, res) => {
26        res.sendFile(path.join(__dirname, '../frontend/index.html'));
27    });
28
29    // Health endpoint
30    app.get('/api/health', (req, res) => {
31        res.json({
32            status: 'OK',
33            message: 'Quiz Master API is running',
34            timestamp: new Date().toISOString(),
35            version: '1.0.0'
36        });
37    });
38
39    // User Registration
40    app.post('/api/register', async (req, res) => {
41        const { username, email, password } = req.body;
```

```

42
43   if (!username || !email || !password) {
44       return res.status(400).json({ error: 'All fields are required' });
45   }
46
47   if (password.length < 6) {
48       return res.status(400).json({ error: 'Password must be at least 6 characters long' });
49   }
50
51   try {
52       // Check if user already exists
53       db.getUserByEmail(email, (err, existingUser) => {
54           if (err) {
55               console.error('Database error:', err);
56               return res.status(500).json({ error: 'Database error' });
57           }
58
59           if (existingUser) {
60               return res.status(400).json({ error: 'User already exists' });
61           }
62
63           // Create new user
64           bcrypt.hash(password, 10, (err, hashedPassword) => {
65               if (err) {
66                   console.error('Password hashing error:', err);
67                   return res.status(500).json({ error: 'Server error' });
68               }
69
70               db.createUser(username, email, hashedPassword, (err, userId) => {
71                   if (err) {
72                       console.error('User creation error:', err);
73                       return res.status(500).json({ error: 'Error creating user' });
74                   }
75
76                   const token = jwt.sign({ userId, username }, JWT_SECRET);
77                   res.status(201).json({
78                       message: 'User created successfully',
79                       token,
80                       user: { id: userId, username, email }
81                   });
82               });
83           });
84       });
85   } catch (error) {
86       console.error('Registration error:', error);
87       res.status(500).json({ error: 'Server error' });
88   }
89 });
90
91 // User Login

```

```

92 app.post('/api/login', async (req, res) => {
93   const { email, password } = req.body;
94
95   if (!email || !password) {
96     return res.status(400).json({ error: 'Email and password are required' });
97   }
98
99   try {
100     db.getUserByEmail(email, async (err, user) => {
101       if (err) {
102         console.error('Database error:', err);
103         return res.status(500).json({ error: 'Database error' });
104       }
105
106       if (!user) {
107         return res.status(400).json({ error: 'Invalid credentials' });
108       }
109
110       try {
111         const validPassword = await bcrypt.compare(password, user.password);
112         if (!validPassword) {
113           return res.status(400).json({ error: 'Invalid credentials' });
114         }
115
116         const token = jwt.sign({ userId: user.id, username: user.username }, JWT_SECRET);
117         res.json({
118           message: 'Login successful',
119           token,
120           user: { id: user.id, username: user.username, email: user.email }
121         });
122       } catch (error) {
123         console.error('Password comparison error:', error);
124         res.status(500).json({ error: 'Server error' });
125       }
126     });
127   } catch (error) {
128     console.error('Login error:', error);
129     res.status(500).json({ error: 'Server error' });
130   }
131 });
132
133 // Authentication middleware
134 const authenticateToken = (req, res, next) => {
135   const authHeader = req.headers['authorization'];
136   const token = authHeader && authHeader.split(' ')[1];
137
138   if (!token) {
139     return res.status(401).json({ error: 'Access token required' });
140   }
141

```

```

142     jwt.verify(token, JWT_SECRET, (err, user) => {
143         if (err) {
144             return res.status(403).json({ error: 'Invalid token' });
145         }
146         req.user = user;
147         next();
148     });
149 };
150
151 // Get user's quizzes
152 app.get('/api/my-quizzes', authenticateToken, (req, res) => {
153     db.getQuizzesByUser(req.user.userId, (err, quizzes) => {
154         if (err) {
155             console.error('Error fetching quizzes:', err);
156             return res.status(500).json({ error: 'Failed to fetch quizzes' });
157         }
158         res.json(quizzes || []);
159     });
160 });
161
162 // Create new quiz
163 app.post('/api/quizzes', authenticateToken, (req, res) => {
164     const { title, description, timeLimit, questions } = req.body;
165     const userId = req.user.userId;
166
167     if (!title || !title.trim()) {
168         return res.status(400).json({ error: 'Quiz title is required' });
169     }
170
171     if (!questions || questions.length === 0) {
172         return res.status(400).json({ error: 'At least one question is required' });
173     }
174
175     // Create quiz
176     db.createQuiz({
177         title: title.trim(),
178         description: description ? description.trim() : '',
179         time_limit: timeLimit || 10,
180         user_id: userId
181     }, (err, quizId) => {
182         if (err) {
183             console.error('Error creating quiz:', err);
184             return res.status(500).json({ error: 'Failed to create quiz' });
185         }
186
187         // Add questions – FIXED: Ensure options are properly stored
188         let questionsAdded = 0;
189         let hasError = false;
190
191         questions.forEach((question, index) => {

```

```

192 // FIXED: Extract option texts from the array
193 const options = question.options || [];
194
195 const questionData = {
196   quiz_id: quizId,
197   question_text: question.text,
198   option_a: options[0] || '', // First option
199   option_b: options[1] || '', // Second option
200   option_c: options[2] || null, // Third option (optional)
201   option_d: options[3] || null, // Fourth option (optional)
202   correct_answer: question.correct,
203   question_order: index
204 };
205
206 db.addQuestion(questionData, (err) => {
207   if (err) {
208     console.error('Error adding question:', err);
209     hasError = true;
210   }
211   questionsAdded++;
212
213   if (questionsAdded === questions.length) {
214     if (hasError) {
215       return res.status(500).json({ error: 'Some questions failed to save' });
216     }
217     res.status(201).json({
218       message: 'Quiz created successfully',
219       quizId: quizId
220     });
221   }
222 });
223 });
224 });
225 });
226
227 // Get single quiz
228 app.get('/api/quizzes/:id', authenticateToken, (req, res) => {
229   const quizId = parseInt(req.params.id);
230
231   db.getQuizById(quizId, (err, quiz) => {
232     if (err) {
233       console.error('Error fetching quiz:', err);
234       return res.status(500).json({ error: 'Failed to fetch quiz' });
235     }
236
237     if (!quiz) {
238       return res.status(404).json({ error: 'Quiz not found' });
239     }
240
241     res.json(quiz);

```

```

242     });
243 });
244
245 // Save quiz results
246 app.post('/api/quiz-results', authenticateToken, (req, res) => {
247     const { quizId, score, totalQuestions, timeTaken, answers } = req.body;
248     const userId = req.user.userId;
249
250     if (!quizId || score === undefined || !totalQuestions || !timeTaken) {
251         return res.status(400).json({ error: 'Missing required fields' });
252     }
253
254     db.saveResult({
255         quiz_id: quizId,
256         user_id: userId,
257         score,
258         total_questions: totalQuestions,
259         time_taken: timeTaken,
260         answers: answers
261     }, (err, resultId) => {
262         if (err) {
263             console.error('Error saving result:', err);
264             return res.status(500).json({ error: 'Failed to save results' });
265         }
266
267         res.json({
268             message: 'Results saved successfully',
269             resultId: resultId
270         });
271     });
272 });
273
274 // Get user's quiz results
275 app.get('/api/my-results', authenticateToken, (req, res) => {
276     db.getResultsByUser(req.user.userId, (err, results) => {
277         if (err) {
278             console.error('Error fetching results:', err);
279             return res.status(500).json({ error: 'Failed to fetch results' });
280         }
281         res.json(results || []);
282     });
283 });
284
285 // Delete quiz
286 app.delete('/api/quizzes/:id', authenticateToken, (req, res) => {
287     const quizId = parseInt(req.params.id);
288     const userId = req.user.userId;
289
290     db.deleteQuiz(quizId, userId, (err, changes) => {
291         if (err) {

```

```

292     console.error('Error deleting quiz:', err);
293     return res.status(500).json({ error: 'Failed to delete quiz' });
294 }
295
296 if (changes === 0) {
297     return res.status(404).json({ error: 'Quiz not found or access denied' });
298 }
299
300 res.json({ message: 'Quiz deleted successfully' });
301 });
302 });
303
304 // Add demo data on startup
305 function addDemoData() {
306     const demoPassword = bcrypt.hashSync('password123', 10);
307
308     // Demo user
309     db.createUser('demo', 'demo@quizmaster.com', demoPassword, (err, userId) => {
310         if (err && !err.message.includes('UNIQUE constraint failed')) {
311             console.error('Error creating demo user:', err);
312         } else {
313             console.log('    Demo user created: demo@quizmaster.com / password123');
314
315             // Create a demo quiz
316             const demoQuiz = {
317                 title: 'Web Development Basics',
318                 description: 'Test your knowledge of HTML, CSS, and JavaScript fundamentals',
319                 time_limit: 10,
320                 user_id: userId
321             };
322
323             db.createQuiz(demoQuiz, (err, quizId) => {
324                 if (!err) {
325                     // Add demo questions
326                     const demoQuestions = [
327                         {
328                             quiz_id: quizId,
329                             question_text: 'What does HTML stand for?',
330                             option_a: 'Hyper Text Markup Language',
331                             option_b: 'High Tech Modern Language',
332                             option_c: 'Hyper Transfer Markup Language',
333                             option_d: 'Home Tool Markup Language',
334                             correct_answer: 0,
335                             question_order: 0
336                         },
337                         {
338                             quiz_id: quizId,
339                             question_text: 'Which CSS property is used to change the text color?',
340                             option_a: 'text-color',
341                             option_b: 'color',

```

```

342         option_c: 'font-color',
343         option_d: 'text-style',
344         correct_answer: 1,
345         question_order: 1
346     },
347     {
348         quiz_id: quizId,
349         question_text: 'Which symbol is used for single-line comments in
350             JavaScript?',
351         option_a: '//',
352         option_b: '/*',
353         option_c: '--',
354         option_d: '#',
355         correct_answer: 0,
356         question_order: 2
357     }
358 ];
359
360 let questionsAdded = 0;
361 demoQuestions.forEach(q => {
362     db.addQuestion(q, (err) => {
363         if (!err) {
364             questionsAdded++;
365             if (questionsAdded === demoQuestions.length) {
366                 console.log('    Demo quiz created with sample questions');
367             }
368         }
369     });
370 }
371 );
372 }
373 });
374 }
375
376 // Global error handling middleware
377 app.use((err, req, res, next) => {
378     console.error('Unhandled Error:', err);
379     res.status(500).json({
380         error: 'Internal server error',
381         message: process.env.NODE_ENV === 'development' ? err.message : 'Something went wrong'
382     });
383 });
384
385 // Handle 404
386 app.use('*', (req, res) => {
387     res.status(404).json({ error: 'Endpoint not found' });
388 });
389
390 // Add demo data on startup

```

```

391 setTimeout(addDemoData, 1000);
392
393 // Start server
394 app.listen(PORT, () => {
395     console.log(`\n Quiz Master Server running on port ${PORT}`);
396     console.log(` Frontend Application: http://localhost:${PORT}`);
397     console.log(` API Health: http://localhost:${PORT}/api/health`);
398     console.log(`\n Demo Account:);
399     console.log(` Email: demo@quizmaster.com`);
400     console.log(` Password: password123`);
401     console.log(`\n Press Ctrl+C to stop the server\n`);
402
403     // Display a nice ASCII art banner
404     console.log(`
405
406
407         Q U I Z   M A S T E R   S E R V E R
408
409         Server successfully started! Ready for connections.
410
411
412     `);
413 });
414
415 // Error handling
416 process.on('uncaughtException', (error) => {
417     console.error(` Uncaught Exception:`, error);
418 });
419
420 process.on('unhandledRejection', (reason, promise) => {
421     console.error(` Unhandled Rejection at:`, promise, `reason:`, reason);
422 });

```

Chapter 9

SCREENSHOTS

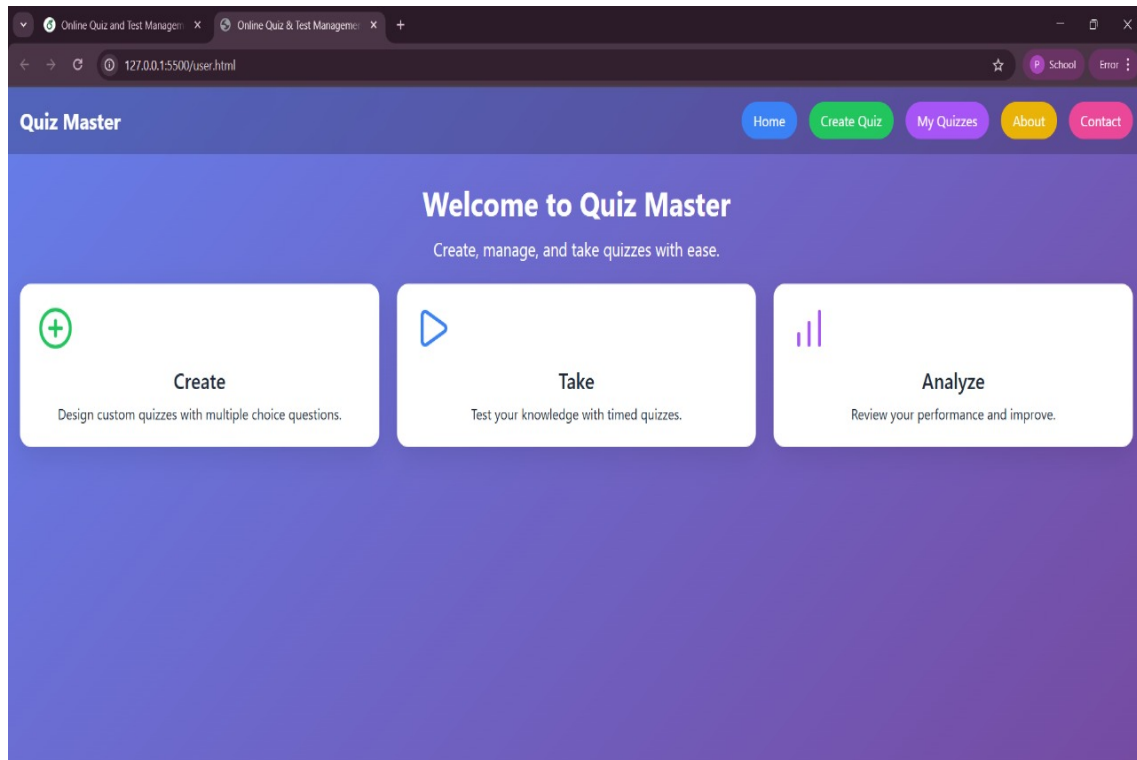


Figure 9.1: Home Page

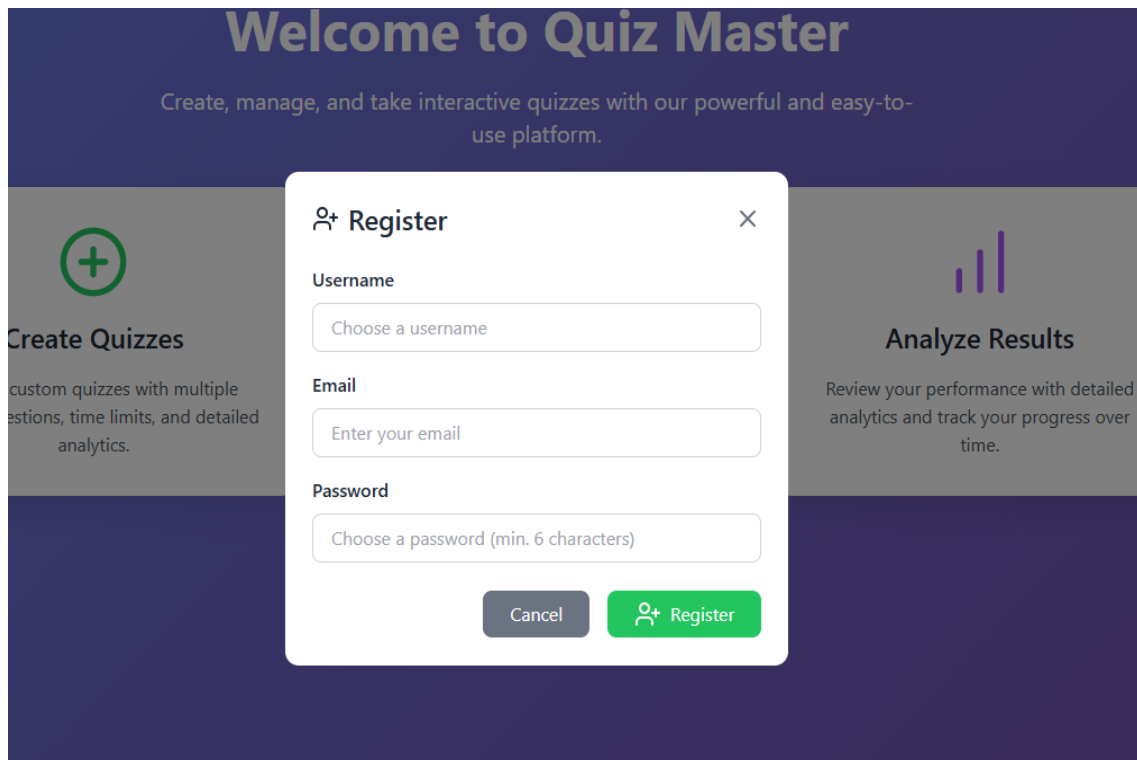


Figure 9.2: Signup Page

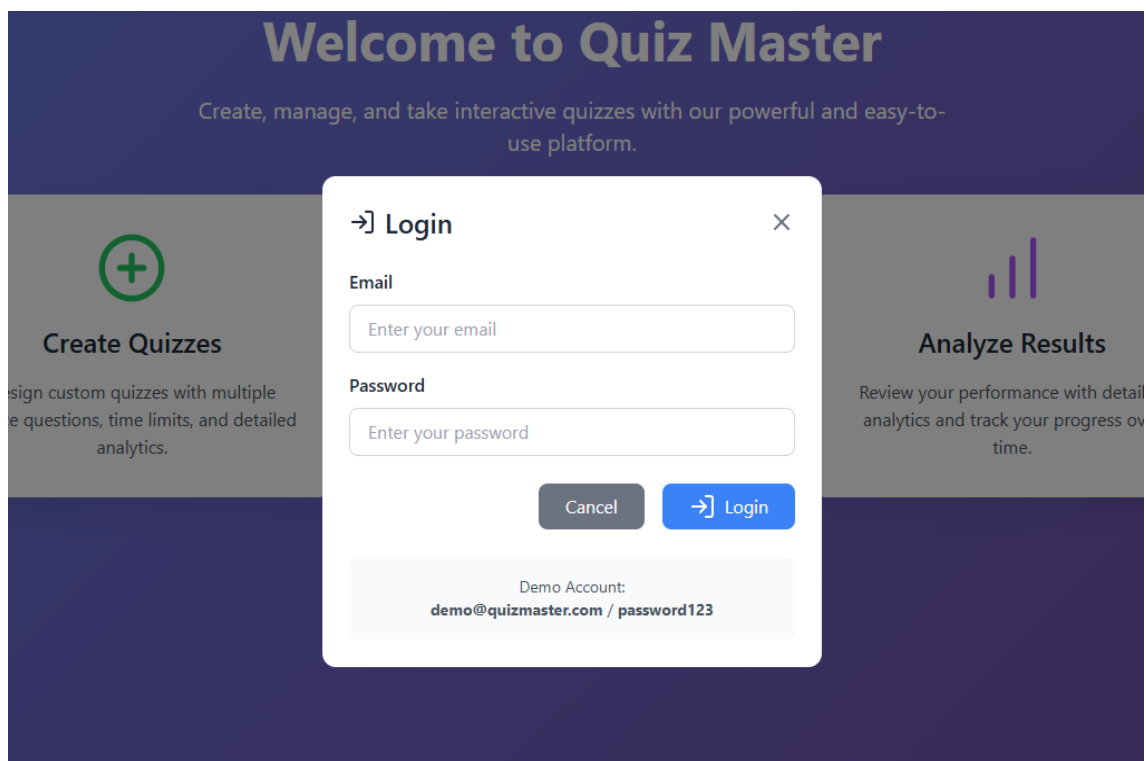


Figure 9.3: Login Page

Quiz Master [Home](#) [+ Create Quiz](#) [My Quizzes](#) [About](#) [Contact](#) [Welcome, demo](#) [Logout](#)

+ Create New Quiz

Quiz Title *

Description

Time Limit (minutes) *

Questions 0 questions

[+ Add Question](#)

[Cancel](#) [Save Quiz](#)

Figure 9.4: Create Quiz

Quiz Master [Home](#) [+ Create Quiz](#) [My Quizzes](#) [About](#) [Contact](#) [Welcome, demo](#) [Logout](#)

My Quizzes

WMAD

basics

🕒 10 min 📅 21/10/2025

[▶ Take Quiz](#)

[🗑 Delete Quiz](#)

Figure 9.5: My Quizzes

REFERENCES

- [1] M. Casciaro, “Node.js Design Patterns,” Packt Publishing, Dec. 29, 2014.
- [2] “Express.js – Node.js web application framework,” Express.js, <https://expressjs.com/>.
- [3] “MySQL – Open-source relational database management system,” MySQL, <https://www.mysql.com/>.
- [4] J. Duckett, “HTML and CSS: Design and Build Websites,” Wiley, 2011.
- [5] B. McFarland, “JavaScript jQuery: The Missing Manual,” O’Reilly Media, 2014.