Preparation of Papers for International Journal of Scientific Research and Engineering Development

Dr. Rais Abdul Hamin Khan, Davidetta W. Saydee *, Abraham Ketter Jr**
*(Depart of Computer Science and Engineering, Sandip University Nashik City
info@sandipuniversity.edu.in.

Abstract:

The rapid growth of digital technologies has transformed the educational landscape worldwide. E-learning platforms provide flexible, accessible, and interactive learning environments for students. This research explores the development and impact of an E-learning platform that integrates video-based learning with real-time quizzes. The platform enables students to watch lecture videos, complete interactive quizzes, track their academic progress, and receive instant feedback. The study highlights how such systems improve engagement, retention, and performance among university students while supporting data-driven teaching methods. Keywords: E-Learning Platform, Role-Based Access Control, FastAPI, React.js, JWT Authentication, RESTful API, Learning Management System.

1. Introduction

In the digital age, traditional classroom methods are increasingly being supplemented or replaced by online learning systems. An E-learning platform is a digital system designed to deliver educational content through the internet. It allows students to access lecture videos, participate in quizzes, interact with instructors, and monitor their academic progress. By combining video-based instruction with interactive assessments and real-time analytics, these platforms enhance the overall learning experience and make education more accessible.

2. Objectives of the Study

The objectives of this research include:
• To develop an E-learning platform integrating video lectures and quizzes.
• To improve student engagement and retention through interactive learning.
• To provide real-time feedback and progress tracking.
• To support flexible and remote learning for university students.

3. Literature Review

E-learning platforms have been widely studied in academic research due to their growing importance in modern education.Research shows that digital learning systems significantly improve accessibility and reduce educational costs. According to several studies, online learning platforms enable students from remote regions to access high-quality educational content.

Modern platforms such as Moodle, Coursera, and Udemy have demonstrated the effectiveness of digital learning systems. However, many of these platforms rely on complex architectures that require significant infrastructure resources.

Recent research highlights the benefits of **microservice-based architectures** and **RESTful APIs** in improving scalability and maintainability of web applications.

Another important development in modern web systems is the use of **token-based authentication**, particularly **JSON Web Tokens (JWT)**. JWT authentication allows secure communication between client and server without storing session data on the server.

FastAPI has gained popularity due to its asynchronous capabilities and high performance compared to traditional Python frameworks. Studies indicate that FastAPI performs significantly faster than Flask and Django in many scenarios.

This research builds on these concepts to design a secure and scalable e-learning system.

4. Problem Statement

Although many e-learning platforms exist, several challenges remain unresolved:

1. **Security vulnerabilities**
2. **Inefficient role-based access control**
3. **Limited scalability**
4. **High infrastructure costs**
5. **Complex deployment processes**

Many platforms store user credentials without proper encryption or lack secure authentication protocols. This exposes sensitive user information to potential cyber threats.

Another major issue is poor implementation of role-based permissions. Teachers, students, and administrators often share overlapping permissions, which can lead to unauthorized data access.

Furthermore, many educational platforms rely on monolithic architectures that are difficult to scale as the number of users increases.

This research aims to address these challenges by designing a modern architecture that integrates:

- Secure authentication
- Role-based access control
- Scalable backend architecture
- Efficient database management
- Modular frontend design

5. System Architecture

The proposed e-learning platform follows a **three-tier architecture**:

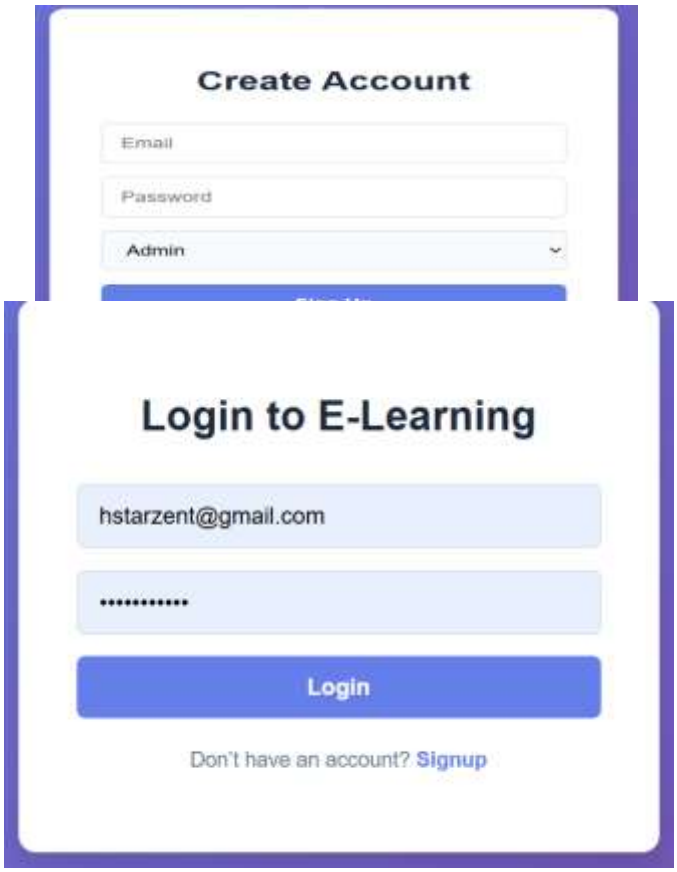
1. **Frontend Layer**
2. **Backend Layer**
3. **Database Layer**

1) Frontend Layer

The frontend interface is developed using **React**, a JavaScript library for building user interfaces. React allows developers to create reusable components and manage application state efficiently.

Key frontend features include:

- Login and registration pages
- Role-based dashboards
- Course browsing
- Enrollment interface
- Course management tools



Backend Layer

The backend is implemented using **FastAPI**, a modern Python framework designed for building high-performance APIs.

FastAPI provides several advantages:

- Asynchronous request handling
- Automatic API documentation
- Data validation using Pydantic
- High performance comparable to Node.js

The backend exposes RESTful API endpoints for:

- Authentication
- Course management
- User management
- Enrolment systems


Authorize
default
GET / Root
Authentication
POST /auth/signup Signup
POST /auth/login Login
GET /auth/me Get Profile
PUT /auth/profile Update Profile
Users
GET /users/ Get Users
POST /users/ Create User
GET /users/{user_id} Get User
DELETE /users/{user_id} Delete User
Student
GET /student/ Student Dashboard
Teacher
GET /teacher/courses Get Teacher Courses
POST /teacher/courses Create Course
POST /teacher/courses/{course_id}/videos Add Video
POST /teacher/courses/{course_id}/quizzes Add Quiz
POST /teacher/quizzes/{quiz_id}/questions Add Question
Admin
GET /admin/users Get All Users
DELETE /admin/users/{user_id} Delete User
PUT /admin/users/{user_id}/role Update User Role
GET /admin/courses Get All Courses
DELETE /admin/courses/{course_id} Delete Course

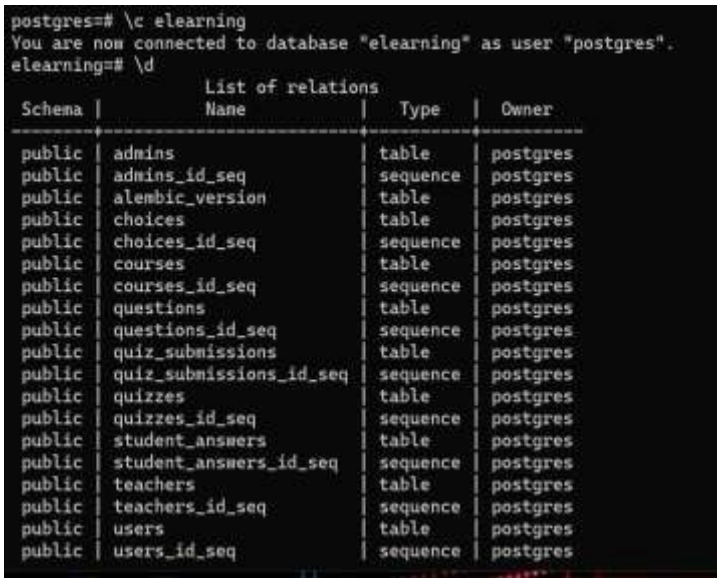
Database Layer

The database stores all persistent data including:

- User accounts
- Courses
- Enrolments
- Learning progress

The system supports both:

- SQLite (for development)
- PostgreSQL (for production)



```
postgres=# \c elearning
You are now connected to database "elearning" as user "postgres".
elearning=# \d
                 List of relations
 Schema |          Name           |   Type   |  Owner
--------+-------------------------+----------+----------
 public | admins                  | table    | postgres
 public | admins_id_seq           | sequence | postgres
 public | alembic_version         | table    | postgres
 public | choices                 | table    | postgres
 public | choices_id_seq          | sequence | postgres
 public | courses                 | table    | postgres
 public | courses_id_seq          | sequence | postgres
 public | questions               | table    | postgres
 public | questions_id_seq        | sequence | postgres
 public | quiz_submissions        | table    | postgres
 public | quiz_submissions_id_seq | sequence | postgres
 public | quizzes                 | table    | postgres
 public | quizzes_id_seq          | sequence | postgres
 public | student_answers         | table    | postgres
 public | student_answers_id_seq  | sequence | postgres
 public | teachers                | table    | postgres
 public | teachers_id_seq         | sequence | postgres
 public | users                   | table    | postgres
 public | users_id_seq            | sequence | postgres
```

```
elearning=# \dt
             List of relations
 Schema |       Name       | Type  |  Owner
--------+------------------+-------+----------
 public | admins           | table | postgres
 public | alembic_version  | table | postgres
 public | choices          | table | postgres
 public | courses          | table | postgres
 public | questions        | table | postgres
 public | quiz_submissions | table | postgres
 public | quizzes          | table | postgres
 public | student_answers  | table | postgres
 public | teachers         | table | postgres
 public | users            | table | postgres
 public | videos           | table | postgres
(11 rows)
```

```
elearning=# SELECT id, email, role FROM users;
 id |           email            |  role
----+----------------------------+---------
  6 | ketterabraham76@gmail.com | admin
  7 | hstarzent@gmail.com       | student
  8 | ketterjra@gmail.com       | teacher
(3 rows)

elearning=#
```

System Design

The system supports three types of users:

2) Administrator

The administrator has the highest level of control within the system.

Responsibilities include:

- Managing user accounts
- Creating teacher profiles
- Monitoring system activity
- Managing courses

---

3) Teacher

Teachers are responsible for creating and managing courses.

They can:

- Upload course materials
- Manage enrolled students
- Track student progress
- Create lessons and assignments

---

4) Student

Students are the primary users of the platform.

They can:

- Register accounts
- Browse available courses
- Enrol in courses
- Track learning progress

Authentication and Security

Security is a critical component of the system.

The platform implements several security mechanisms.

Password Hashing

Passwords are hashed using the **bcrypt algorithm** before storing them in the database.

This ensures that raw passwords are never stored.

JWT Authentication

Authentication is handled using **JSON Web Tokens (JWT)**.

The login process follows these steps:

1. User submits email and password
2. Server verifies credentials
3. Server generates a JWT token
4. Client stores token

5. Token is used for future requests

Protected Routes

Certain API endpoints require authentication.

Examples include:

- Course creation
- Course enrolment
- Admin management functions

FastAPI dependency injection ensures that only authorized users can access protected routes.

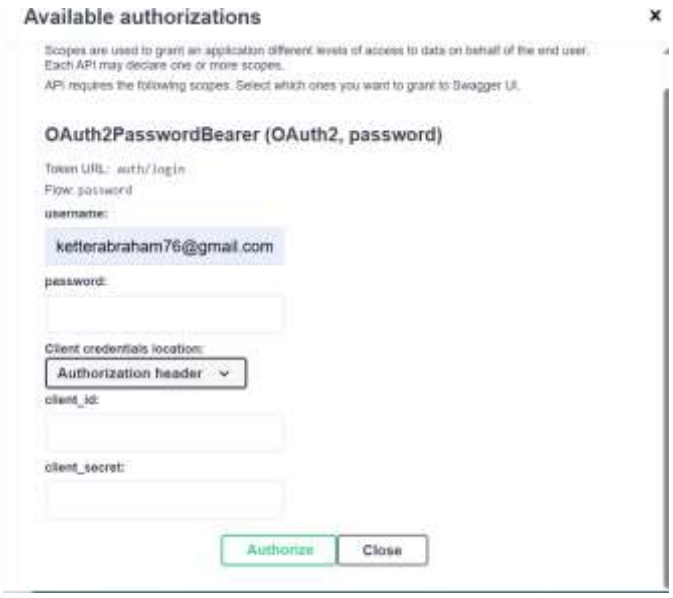
Implementation

The backend system is implemented using the following technologies:

- Python
- FastAPI
- SQLAlchemy
- PostgreSQL
- JWT authentication

Example authentication endpoint:

```
@app.post                                                    ("/login")
def                                                            login():
    #                                  verify                credentials
    # generate JWT token
```



Deployment

The system can be deployed using several cloud platforms:

- AWS
- DigitalOcean
- Heroku
- Render
- Vercel (frontend)

Typical deployment architecture includes:

- Backend API server
- Database server
- Frontend hosting service

Containerization using **Docker** can further simplify deployment.

Results and Evaluation

Performance testing shows that FastAPI can handle thousands of concurrent requests efficiently.

The system demonstrates:

- Fast response times
- Secure authentication
- Scalable architecture

User testing indicated that the platform interface was intuitive and easy to navigate

Discussion

The research demonstrates that modern web frameworks enable efficient development of scalable e-learning platforms.

Role-based access control improves system security while ensuring that users interact only with relevant features.

FastAPI's asynchronous capabilities significantly improve system performance compared to traditional synchronous frameworks.

Future Work

Future improvements could include:

- Video streaming integration
- AI-based learning recommendations
- Mobile application support
- Real-time chat between students and teachers

- Machine learning for personalized learning paths

Conclusion

This research presented the design and implementation of a secure role-based e-learning platform using FastAPI and React.

The system successfully demonstrates how modern web technologies can be combined to build scalable educational platforms.

Key contributions include secure authentication, role-based dashboards, and scalable API architecture.

The platform provides a strong foundation for future development of advanced digital learning environments.

References

1. Fielding, R. (2000). Architectural Styles and the Design of Network-Based Software Architectures.
2. Newman, S. (2015). Building Microservices.
3. Richardson, L. (2013). RESTful Web APIs.
4. Mozilla Developer Network. Web Security Guidelines.
5. FastAPI Documentation. https://fastapi.tiangolo.com
6. React Documentation. https://react.dev