Webapp Deployment and Security for Research Project Management System

This document outlines the deployment and security practices for a web application to manage the registration and review process of student research projects. The system involves three types of users: students, lecturers, and administrators.

# 1. System Design and Requirement Analysis

a. User Roles

- Students:  
 - Register and log in.  
 - Create and edit research project submissions.  
 - View the review status of their projects.  
  
- Lecturers:  
 - Review the assigned research projects.  
 - Provide feedback and approve or reject projects.  
  
- Administrators:  
 - Manage student and lecturer accounts.  
 - Assign lecturers to review specific projects.  
 - Monitor reports and project review status.

b. Core Functions

- User Registration and Authentication: Functions for account creation, login, logout, email verification, and optional 2FA.  
- Project Management: Students can create projects, lecturers can review them, and administrators can assign reviewers and monitor progress.  
- Reporting and Statistics: Administrators can view reports on projects, including their status, review outcomes, and generate downloadable reports.

c. Technology Stack

- Frontend: HTML, CSS, JavaScript (React.js, Vue.js, or Angular).  
- Backend: Node.js, Django, or Flask.  
- Database: MySQL, PostgreSQL, or MongoDB.  
- Hosting: AWS, DigitalOcean, Azure, or similar platforms.

# 2. Implementation of Role-based Access Control (RBAC)

a. Define Roles

Roles include `student`, `lecturer`, and `admin`, each with specific access rights. These roles determine what actions users can perform.

b. User Registration and Login

- Use bcrypt to hash passwords before saving them to the database.  
- Implement JWT-based authentication to issue tokens for users after successful login.  
- Consider implementing 2FA using tools like `speakeasy` for time-based one-time passwords (TOTP).

c. Example Database Schema for User Roles

CREATE TABLE users (  
 id SERIAL PRIMARY KEY,  
 username VARCHAR(50) NOT NULL UNIQUE,  
 password\_hash VARCHAR(255) NOT NULL,  
 email VARCHAR(100) NOT NULL UNIQUE,  
 role ENUM('student', 'lecturer', 'admin') NOT NULL,  
 created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP  
);

# 3. Project Management Functionality

a. Student Project Submission

Students can submit their research projects through forms with input validation.

b. Lecturer Review Process

Lecturers can review and approve or reject the assigned projects, ensuring data access is restricted by roles.

c. Admin Management

Admins can assign lecturers, view all projects, and generate reports.

# 4. API and Database Security

a. Database Security

- Use prepared statements to prevent SQL injection.  
- Encrypt sensitive data like passwords using bcrypt before saving it in the database.

b. API Security

Implement JWT authentication and rate limiting to protect against brute force attacks.

# 5. Frontend Security

a. Minify and Obfuscate Source Code

Use tools like Terser or UglifyJS to minify JavaScript code and remove source maps before deployment.

# 6. Deployment and Monitoring

Deploy the system on a reliable hosting platform like AWS or DigitalOcean, set up automatic backups, and use monitoring tools like Grafana or Prometheus for system health tracking.

# 7. Folder Structure

Example folder structure for full-stack development (React/Vue + Node.js/Django):

project-root/  
├── client/  
│ ├── public/  
│ ├── src/  
│ ├── components/  
│ ├── pages/  
│ ├── services/  
│ ├── store/  
│ └── utils/  
├── server/  
│ ├── config/  
│ ├── controllers/  
│ ├── models/  
│ ├── routes/  
│ ├── services/  
│ └── middlewares/  
├── database/  
│ ├── migrations/  
│ ├── seeds/  
│ └── schema.sql  
├── tests/  
├── logs/  
├── scripts/  
├── .env  
└── README.md