TOPIC: Project Review 1 **DATE:** 23.07.2024

COLLEGE NAME: M. Kumarasamy College of Engineering

DEPARTMENT: AIML, ECE

TITLE: Role Based Access Control (RBAC)System

TEAM NAME: New Risers

TEAM MEMBERS:

1. Vaishnavi N
2. Puvitha Sri N
3. Pruthiga M S
4. Prithiga V

AIM:

The aim of this project is to implement a Role-Based Access Control (RBAC) system to improve security by ensuring users only access what they need for their roles. This will simplify how access is managed, making it easier to add or remove users. The system will also help meet regulatory requirements and increase operational efficiency, reducing the time and effort needed to manage access permissions.

OBJECTIVE:

The primary objective of a Role-Based Access Control (RBAC) system is to enhance security and streamline the management of user permissions within an organization. By assigning access rights based on roles rather than individual users, RBAC minimizes the risk of unauthorized access and potential security breaches. It simplifies user management, allowing administrators to efficiently assign and reassign roles without adjusting individual permissions. This system ensures compliance with regulatory requirements by restricting access to sensitive information based on defined roles and responsibilities. RBAC also increases administrative efficiency, reducing complexity and enabling quick onboarding of new employees or role changes. Centralized role management reduces errors in permission assignments, ensuring appropriate access levels for all users. Furthermore, an RBAC system supports scalability, easily adapting to organizational growth and changes without compromising security. Finally, RBAC provides robust auditability, maintaining clear records of role-based access to resources and user assignments, which is crucial for auditing and reporting purposes.

METHODOLOGY:

Planning:

1. **Gather Requirements:**
   - Identify the specific needs of the organization.
   - Consult with stakeholders to understand their access control needs.
   - Define the scope and objectives of the RBAC system.
2. **Research Existing Tools:**
   - Evaluate current access control systems and RBAC tools.
   - Compare features, scalability, and integration capabilities.
   - Select the most suitable tools for your project.

Implementation:

1. **Set Up the Development Environment:**
   - Configure the necessary hardware and software.
   - Install required development tools and frameworks.
   - Establish a version control system for tracking changes.
2. **Code and Documentation:**
   - Develop the RBAC system according to the gathered requirements.

- Ensure thorough documentation of the codebase, including comments and usage instructions.
- Create user manuals and technical documentation for future reference.

HTML CODE:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <meta http-equiv="X-UA-Compatible" content="IE=edge" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <title>Login</title>
  <link rel="stylesheet" href="style.css" />
  <script src="https://www.gstatic.com/firebasejs/8.10.0/firebase-app.js"></script>
  <script src="https://www.gstatic.com/firebasejs/8.10.0/firebase-auth.js"></script>
  <script src="https://www.gstatic.com/firebasejs/8.10.0/firebase-storage.js"></script>
  <script src="https://www.gstatic.com/firebasejs/8.10.0/firebase-firestore.js"></script>
  <script src="https://www.gstatic.com/firebasejs/8.10.0/firebase-database.js"></script>
  <script src="firebase.js"></script>
  <script>
  function login() {
  const email = document.getElementById('loginEmail').value;
  const password = document.getElementById('loginPassword').value;
   if(email==="admin@gmail.com" && password === "admin@12")
```

```
    {
     window.location.href = 'admin.html';
    }
    firebase.auth().signInWithEmailAndPassword(email, password)
     .then((userCredential) => {
      const user = userCredential.user;
      window.location.href = index.html?email=${email};
     })
     .catch((error) => {
      const errorCode = error.code;
      const errorMessage = error.message;
      console.error('Login error:', errorMessage);
      alert(errorMessage);
     });
   }
 </script>
 <script>
   function signUpUser(name, email, password, idProofFile) {
    const usersRef = database.ref('users');
    // Check if the email is already registered
    usersRef.orderByChild('email').equalTo(email).once('value')
     .then(snapshot => {
      if (snapshot.exists()) {
```

```
  alert('This email is already registered. Please use a different email.');
} else {
  // Upload ID proof image to Firebase Storage
  const storageRef = firebase.storage().ref();
  const idProofImageRef = storageRef.child(idProofs/${email}_${idProofFile.name});
  return idProofImageRef.put(idProofFile)
    .then(snapshot => snapshot.ref.getDownloadURL())
    .then(idProofURL => {
      return database.ref('users').push({
        name: name,
        email: email,
        password: password,
        approved: 'no',
        idProofURL: idProofURL
      });
    })
    .then(() => {
      alert('Sign up successful! Wait for admin approval.');
      window.location.reload();
    })
    .catch(error => {
      console.error('Error uploading ID proof or saving user data:', error);
      alert('An error occurred during sign up. Please try again.');
```

```
          });
        }
      })
      .catch(error => {
        console.error('Error checking email existence:', error);
        alert('An error occurred. Please try again.');
      });
  }


  function signup() {
    event.preventDefault();
    const name = document.querySelector('input[name="name"]').value;
    const email = document.querySelector('input[name="email"]').value;
    const password = document.querySelector('input[name="password"]').value;
    const idProofFile = document.querySelector('input[name="idProof"]').files[0]; // Get uploaded
file
    if (!idProofFile) {
      alert('Please upload ID proof.');
      return;
    }
    signUpUser(name, email, password, idProofFile);
  }
```

```
 </script>
</head>
<body>
 <main>
  <center>
  <div class="box">
   <div class="inner-box">
    <div class="forms-wrap">
     <form class="sign-in-form">
      <div class="heading">
       <h2>Role Based Access Control</h2><br>
       <h6>Not registered yet?</h6>
       <a href="#" class="toggle">Create Your Account</a>
      </div>
      <div class="actual-form">
       <div class="input-wrap">
       <input type="text" id="loginEmail" class="input-field" placeholder="Email" required />
       </div>
       <div class="input-wrap">
      <input type="password" id="loginPassword" class="input-field" placeholder="Password"
required />
       </div>
```

```
<input type="button" onclick="login()" value="Get Started &nbsp;&nbsp;&nbsp;&nbsp;&rarr;"
class="sign-btn" />
      </div>
     </form>
     <form class="sign-up-form" onsubmit="signup();">
      <div class="heading">
       <h2>Role Based Access Control</h2><br>
       <h6>Already have an account?</h6>
       <a href="#" class="toggle">Sign in</a>
      </div>
      <div class="actual-form">
       <div class="input-wrap">
        <input type="text" name="name" class="input-field" placeholder="Name" required />
       </div>
       <div class="input-wrap">
        <input type="email" name="email" class="input-field" placeholder="Email" required />
       </div>
       <div class="input-wrap">
        <input type="password" name="password" class="input-field" placeholder="Password"
required />
       </div>
       <div class="input-wrap">
        <select name="" id="" class="input-field" required/>
```

```
                <option value="">Select Role</option>
                <option value="">Designer</option>
                <option value="">Developer</option>
                <option value="">Tester</option>
               </select>
              </div>
              <input type="submit" value="Sign Up" class="sign-btn" />
             </div>
            </form>
          </div>
          <div class="carousel">
            <div class="images-wrapper">
              <img src="./img/security.webp" class="image img-1 show" height="150%" alt="" />
            </div>
            </div>
          </div>
        </div>
      </div>
    </center>
      </main>
      <script src="app.js"></script>
    </body>
    </html>
```

Testing:

1. **Integration Testing:**
    - Test the integration of the RBAC system with existing IT infrastructure.
    - Verify that the system works seamlessly with other applications and services.
2. **Performance Testing:**
    - Assess the system's performance under various load conditions.
    - Ensure the system can handle the expected number of users and access requests without degradation.
3. **Security Testing:**
    - Conduct security assessments to identify vulnerabilities.
    - Perform penetration testing to ensure the system's robustness against attacks.
    - Implement necessary security measures to protect sensitive data.

Deployment:

1. **Package the Tool:**
    - Compile the RBAC system into a deployable format.
    - Ensure all dependencies and configurations are included.
2. **Deploy to Users:**
    - Roll out the RBAC system to the intended user base.
    - Provide training and support to ensure smooth adoption.
    - Monitor the deployment for any issues and address them promptly.

Existing Tools Disadvantages:

- Manual Permission Management: Traditional systems require manual updates and management of user permissions, which is time-consuming and prone to errors.
- Inflexibility: Many current access control tools lack the flexibility to easily adapt to organizational changes, making it difficult to add or modify roles.
- Inconsistent Security Enforcement: Existing systems often fail to consistently enforce security policies across different applications, leading to potential vulnerabilities.

TOOLS:

Programming Language:

- HTML
- CSS
- JavaScript

Development Environment:

Software: Visual Studio Code

Operating System: Windows

Integration testing:

- Integration with Email service
- Error handling and Recovery

Security and Vulnerability Databases:

- Google Firebase
- Secure Email communication

Reporting:

- It is configured to use SMTP for sending emails
- Generate activity logs
- Maintain history of role and permission changes
- Schedule and send automated reports

GITHUB REPOSITORY LINKS:

https://github.com/shaluvaishnavi/Role-Based-Access-Control-System

https://github.com/PUVITHA2611/Role-Based-Access-Control-RBAC-System

https://github.com/927621BEC157pruthiga/Role-Based-Access-Control-RBAC-System.

https://github.com/PrithigaVelmurugan/Role-based-Access-Control-System