**USER ACCESS MANAGEMENT SYSTEM**

**1. Introduction**

The **User Access Management System (UAMS)** provides an efficient and secure way of managing user access to resources and services within an organization. It ensures that only authorized users can access sensitive information, and provides an administrative interface for managing users, approving access requests, and maintaining security logs. The system supports multiple roles, such as regular users and administrators, and uses role-based access control (RBAC) to manage permissions.

**2. Purpose**

The **UAMS** is designed to:

* **Authenticate users** and control access to different resources.
* Provide **administrative control** over user permissions and access requests.
* **Secure user data** by encrypting sensitive information such as passwords.
* Maintain **audit trails** to track user actions and requests for transparency and security.

**3. Scope**

The system covers:

* **User Registration**: Enables users to sign up for the system.
* **Login & Authentication**: Secure login functionality with password hashing.
* **Role-Based Access Control**: Users are assigned roles determining their access levels.
* **Access Requests**: Users can request access to specific resources.
* **Approval System**: Admins approve or deny access requests.
* **Profile Management**: Users can update their own profiles.

**4. Overall Description**

UAMS is a web-based application consisting of two main interfaces:

* **User Interface**: For users to register, log in, request access, and view request statuses.
* **Admin Interface**: For admins to manage users, review and approve access requests, and configure system settings. The system uses Java for backend logic, JSP for dynamic web pages, and MySQL for database management.

**5. Product Perspective**

UAMS is built using:

* **Frontend**: HTML, CSS, and JavaScript.
* **Backend**: Java, Servlets, JSP pages.
* **Database**: MySQL for storing user data, requests, and logs.
* **Web Server**: Apache Tomcat for running the application.

The system is designed to scale with the needs of growing organizations and can be integrated with other enterprise systems via APIs.

**6. Product Functions**

* **User Registration**: Users can create an account by submitting basic information (e.g., username, password, email).
* **Login**: Users authenticate via username and password.
* **Access Request**: Users can request access to specific resources, such as files or applications.
* **Access Approval**: Admins review and approve or deny requests based on the resource and user role.
* **User Profile Management**: Users can manage their personal details (e.g., email, password).

**7. User Classes and Characteristics**

* **Administrator**: Full access to the system, including user management and approval of access requests.
* **Regular User**: Can request access to resources and view their own request status.
* **Guest User**: Limited to viewing public information, must register for access requests.
* **Support Team**: Assists users but cannot approve requests or manage system configurations.

**8. Operating Environment**

* **Client-Side**: Supported on modern web browsers (Chrome, Firefox, Safari).
* **Server-Side**: Java-based backend with Apache Tomcat as the web server.
* **Database**: MySQL database for storing user, request, and access data.

**9. Design and Implementation Constraints**

* **Security**: User data must be securely stored and transmitted.
* **Scalability**: The system should be able to handle a growing number of users and requests.
* **Cross-Browser Compatibility**: The application must be compatible with multiple browsers.

**10. Specific Requirements**

* **Password Encryption**: Store user passwords using hashing algorithms (e.g., bcrypt).
* **Session Management**: Use secure session handling to ensure logged-in users are properly authenticated.
* **Audit Logging**: Record user activity for accountability.

**11. Signup System**

Users can sign up by filling in a registration form. Upon submission, their details are validated and stored in the database. They will receive a confirmation email upon successful registration.

// Sample Signup Servlet

@WebServlet("/signup")

public class SignupServlet extends HttpServlet {

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

String username = request.getParameter("username");

String password = request.getParameter("password");

String email = request.getParameter("email");

// Encrypt the password before storing

String encryptedPassword = BCrypt.hashpw(password, BCrypt.gensalt());

// Save user in database

UserDAO.saveUser(username, encryptedPassword, email);

// Send confirmation email (simplified)

EmailService.sendConfirmationEmail(email);

response.sendRedirect("login.jsp");

}

}

**12. Login System**

Users log in by entering their username and password. The system verifies their credentials against the database.

// Sample Login Servlet

@WebServlet("/login")

public class LoginServlet extends HttpServlet {

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

String username = request.getParameter("username");

String password = request.getParameter("password");

// Fetch user from database

User user = UserDAO.getUserByUsername(username);

// Check if password matches

if (user != null && BCrypt.checkpw(password, user.getPassword())) {

// Create session

request.getSession().setAttribute("user", user);

response.sendRedirect("dashboard.jsp");

} else {

response.sendRedirect("login.jsp?error=true");

}

}

}

**13. Software Management**

The system is built using:

* **Java** for backend logic.
* **JSP** for dynamic web pages.
* **MySQL** for data storage.
* **Apache Tomcat** as the server.

**14. Access Request System**

Users can submit access requests for resources via a simple form.

// Sample Access Request Servlet

@WebServlet("/access-request")

public class AccessRequestServlet extends HttpServlet {

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

String resource = request.getParameter("resource");

String reason = request.getParameter("reason");

User user = (User) request.getSession().getAttribute("user");

// Save the request in the database

AccessRequestDAO.submitRequest(user.getId(), resource, reason);

response.sendRedirect("requests.jsp");

}

}

**15. Approval System**

Administrators review access requests and approve or deny them.

// Sample Approval Servlet

@WebServlet("/approve-request")

public class ApprovalServlet extends HttpServlet {

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

int requestId = Integer.parseInt(request.getParameter("requestId"));

boolean approved = Boolean.parseBoolean(request.getParameter("approved"));

// Update request status in database

AccessRequestDAO.updateRequestStatus(requestId, approved);

response.sendRedirect("admin-dashboard.jsp");

}

}

**16. Data Requirements**

The system requires the following data:

* **User Data**: Username, password, email, role.
* **Access Requests**: Resource name, reason for access, status (approved/denied).
* **Audit Logs**: User actions, login attempts, access request history.

**17. Database Design**

* **Users** Table: Stores user information.
* **AccessRequests** Table: Stores access request data.
* **Logs** Table: Stores system logs.

-- User Table

CREATE TABLE Users (

id INT PRIMARY KEY AUTO\_INCREMENT,

username VARCHAR(255) NOT NULL,

password VARCHAR(255) NOT NULL,

email VARCHAR(255),

role VARCHAR(50)

);

-- Access Requests Table

CREATE TABLE AccessRequests (

id INT PRIMARY KEY AUTO\_INCREMENT,

userId INT,

resource VARCHAR(255),

reason TEXT,

status VARCHAR(50)

);

**18. Deliverables**

* Source Code: Java files, JSP files, and servlets.
* Database Schema: SQL scripts for creating tables.
* Documentation: Detailed project documentation, installation, and user guide.
* Testing: Unit tests and integration tests for the system.

**19. Evaluation Criteria**

The system will be evaluated based on:

* **Functionality**: Ensuring all features work correctly.
* **Security**: Proper handling of sensitive data and secure access.
* **Usability**: User-friendly interface and ease of navigation.
* **Performance**: The system should handle a growing number of users and requests.
* **Reliability**: The system should be stable and reliable.

**20. Roles Summary**

* **Administrator**: Full system access, including user management and approval of requests.
* **Regular User**: Can request access and manage their own profile.
* **Guest User**: Can view public content and must register for access.
* **Support Team**: Assists with user issues but cannot approve requests.

**Conclusion**

The **User Access Management System (UAMS)** is an essential tool for managing user access securely and efficiently. It ensures that users can access only the resources they are authorized for, while administrators have the necessary control to manage users and approve or deny access requests. The system's modular design and secure access mechanisms make it scalable for growing organizations.