

Role-Based Access Control (RBAC) Web App

Documentation

Overview

This project implements a full-stack RBAC system with organization and department hierarchies, fine-grained permission control, and guest access via secure tokenized URLs. Built using **Flask**, **HTML/Bootstrap/JavaScript**, and **SQLAlchemy**, this system allows administrators to control user access based on roles and manage organizational structure securely.

Features

- JWT-based Authentication
- Role assignment: Admin, Manager, Contributor, Viewer
- Public Guest Access (View/Edit with Expiry)
- Organization and Department Management
- Modular Backend Routes

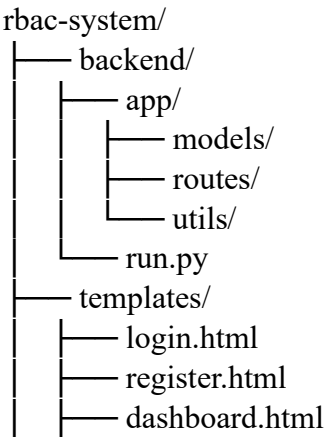
Roles and Permissions

| Role | Create Org | Create Dept | Create Resource | Share Resource | View Content |
|-------------|------------|-------------|-----------------|----------------|---------------|
| Admin | ✓ | ✓ | ✓ | ✓ | ✓ |
| Manager | ✗ | ✓ | ✓ | ✓ | ✓ |
| Contributor | ✗ | ✗ | ✓ | ✗ | ✓ |
| Viewer | ✗ | ✗ | ✗ | ✗ | ✓ |
| Guest | ✗ | ✗ | ✗ | ✗ | ✓ (via token) |

Technologies Used

- **Backend:**
 - Flask
 - Flask-JWT-Extended
 - SQLAlchemy ORM
 - SQLite (can upgrade to PostgreSQL/MySQL)
- **Frontend:**
 - HTML5
 - Bootstrap 5
 - Vanilla JavaScript

File Structure (Simplified)



```
| └─ guest-view.html
| └─ requirements.txt
| └─ README.md
```

Setup Instructions (Localhost)

Backend

```
git clone https://github.com/your-username/rbac-system.git
cd rbac-system
python -m venv venv
venv/Scripts/activate
pip install -r requirements.txt
```

Initialize Database

```
flask shell
>>> from app import db
>>> db.create_all()
>>> exit()
```

Run Server

```
python backend/run.py
Server runs at: http://127.0.0.1:5000
```

Frontend

Open directly from templates/ folder in browser:

- register.html
- login.html
- dashboard.html
- guest-view.html

Postman API Endpoints

Signup

POST /api/auth/signup

```
{
  "name": "devi",
  "email": "devi@gmail.com",
  "password": "devi@123",
  "role": "Admin"
}
```

Login

POST /api/auth/login

```
{
  "email": "devi@gmail.com",
  "password": "devi@123"
}
```

Assign Role

POST /api/auth/assign-role

Authorization: Bearer <token>

```
{
  "email": "devi@gmail.com",
```

```
"role": "Admin"
}
```

Create Organization

POST */api/orgs/create*

Authorization: Bearer <token>

```
{
  "name": "HopCorps"
}
```

Get Organizations

GET */api/orgs/*

Authorization: Bearer <token>

Create Department

POST */api/orgs/<org_id>/departments*

Authorization: Bearer <token>

```
{
  "name": "Finance"
}
```

Get Departments

GET */api/orgs/<org_id>/departments*

Authorization: Bearer <token>

Create Resource

POST */api/resources/<dept_id>/create*

Authorization: Bearer <token>

```
{
  "title": "Employee Salary Management",
  "content": "Initial draft for Employee Salary Management Strategy"
}
```

Get Resources in a Department

GET */api/resources/department/<dept_id>*

Authorization: Bearer <token>

Share Resource

POST */api/resources/<resource_id>/share*

Authorization: Bearer <token>

```
{
  "permission": "edit",
  "expires_in": 5
}
```

Guest Access

- Link: *guest-view.html?token=<your-token>*
- Can view/edit based on token permissions and expiry

Architecture Decisions

| Area | Decision |
|------------------|--|
| Authentication | JWT-based tokens for secure sessions |
| RBAC Enforcement | Role-based decorators for endpoints |
| Guest Access | Scoped, time-limited JWT guest tokens |
| Storage | SQLite used for demo; easily upgradable to PostgreSQL |
| Scalability | Modular folder structure with distinct models/routes/utils |

Challenges Faced

- Handling nested role logic for org/department
- Creating secure and temporary guest tokens
- Keeping frontend minimal but functional without JS frameworks
- Managing complex permissions using decorators

Scope for Improvement

- Integrate OpenFGA or Casbin
- Add a visual frontend with React/Vue
- Add audit logs and token revocation
- Support for OAuth (Google login)
- Real-time permission updates via WebSockets

Conclusion

This RBAC web app mimics modern collaboration tools like Google Docs by implementing a scalable access control system that includes role-based permissions, guest sharing, and a secure backend. Built to be modular and extendable for real-world enterprise needs.