BLOGGER'S STOP
Flask Application

A CS814 Course Project Report

Submitted By:

Abhashri Deshmukh (202IS002)

Department of Computer Science and Engineering
National Institute of Technology Karnataka
P.O. Srinivasnagar, Surathkal, Mangalore-575025
Karnataka, India
January 2021

Table of Contents

INTRODUCTION

A. Use of Application

"Blog" is an abbreviated version of "weblog," which is a term used to describe websites that maintain an ongoing chronicle of information. A blogger is a person who owns or runs a blog or a person who maintains the blog. That is, posting articles or new posts, information, sharing the most up-to-date news, opinions and case studies to name but a few. Such entries are known as blog posts. Blogs range from the personal to the political, and can focus on one narrow subject or a whole range of subjects.

The application Blogger's Stop provides a blueprint as to how a blogging application works at a microscopic level. This application comes up with a platform for bloggers to post blogs.

The usage of the platform requires the registration of the blogger into the application. The posts creation requires the blogger's login to the application. And Hence, the content of the application will not be disclosed to anyone outside the organisation. The complete application is administered by one of the members within the organisation. The purpose of this document is to build a blog application to manage blogs to ease the blogger's.

B. Types of users and their functionalities

There are two types of user in the application mentioned below -

Admin - It is a member of the organisation which has the role of administering the data of the members in the organisation.
It has the following functionalities:

- View Posts - The admin can view all the blogs posted by the normal user into the system.
- Delete Posts - The admin has the authority to delete any of the blogs posted by any user.
- Access Account - The admin can access the account details.
- Update Account - The admin has the authority to update his account details.
- Make Announcements - The admin has the ability to make the announcements.

Client/Normal User - It is a member of the organisation which can post a blog in the application. Every client requires registration to be the part of the members of the organisation. The client will be able to perform operation only after login to the

application. Only a verified user will be able to login to the system. If the user is not authorized, in a way having wrong credentials will not be allowed to enter into the system.

It has the following functionalities:

- View Posts - The user can view all the blogs posted by the normal user into the system.
- Delete Posts - The user has the authority to delete any of the blogs posted by himself/herself.
- Update Posts - The user has the authority to update any of the blogs posted by himself/herself.
- Access Account - The user can access his account details.
- Update Account - The user has the authority to update his account details.
- Create Blogs- The user has the ability to create new posts.

C. Flask Framework

Blogger's stop application is made in Flask Framework. **Flask** is a micro web framework written inPython. It is classified as a micro framework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions.

However, Flask supports extensions that can add application features as if they were implemented in Flask itself. Extensions exist for object-relational mappers, form validation, upload handling, various open authentication technologies and several common framework related tools.

D. Database Used

Flask-**SQLAlchemy** is an extension for Flask that adds support for SQLAlchemy to your application. It aims to simplify using SQLAlchemy with Flask by providing useful defaults and extra helpers that make it easier to accomplish common tasks. After configuring the application, create the SQLAlchemy object by passing it the application.

The code snippet of configuring SQLAlchemy is attached as:

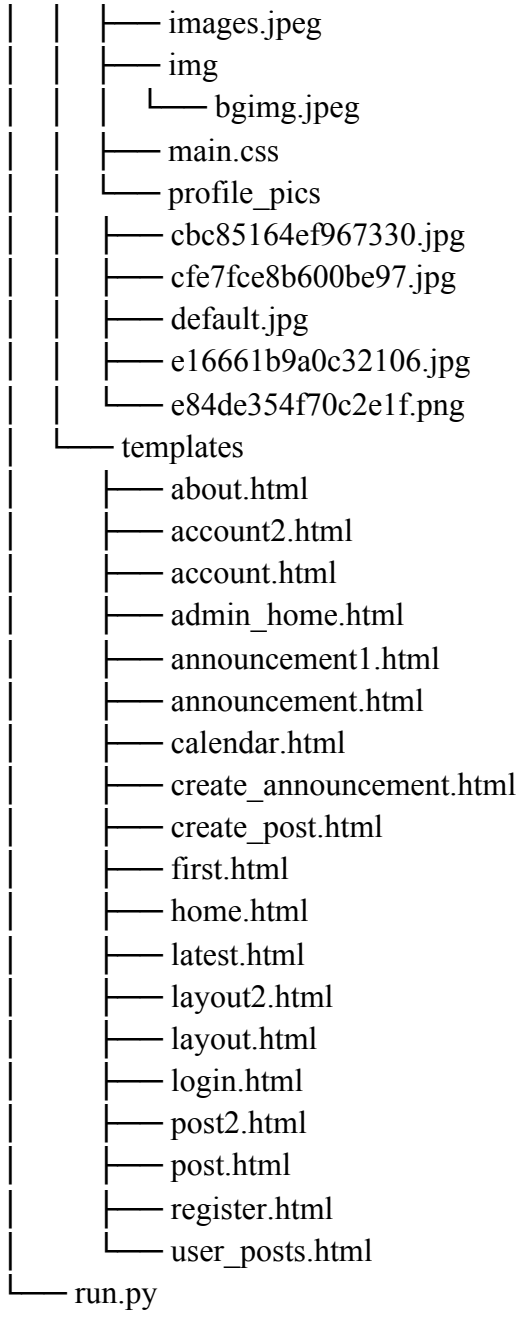
```
from flask import Flask
from flask_sqlalchemy import SQLAlchemy
from flask_bcrypt import Bcrypt
from flask_login import LoginManager

app = Flask(__name__)
app.config['SECRET_KEY'] = '5791628bb0b13ce0c676dfde280ba245'
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///site.db'
db = SQLAlchemy(app)
```

The snippet of how the database looks like is as:

```
abha@Abha-hp:~/Desktop/Flask_Board$ python
Python 3.8.5 (default, Sep  4 2020, 07:30:14)
[GCC 7.3.0] :: Anaconda, Inc. on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> from flaskboard.models import User, Post
/home/abha/.local/lib/python3.8/site-packages/flask_sqlalchemy/__init__.py:833: FSADeprecationWarning: SQLALCHEMY_TRACK_MODIFICATIONS adds sig
nificant overhead and will be disabled by default in the future.  Set it to True or False to suppress this warning.
  warnings.warn(FSADeprecationWarning(
>>> User.query.all()
[User('CoreyMS', 'CoreyMSchafer@gmail.com', 'cbc85164ef967330.jpg'), User('TestUser', 'TestUser@demo.com', 'default.jpg'), User('Abhashri', 'a
bhasd16@gmail.com', 'c95f21e7466af1c1.png'), User('Admin', 'admin@blog.com', 'default.jpg'), User('Ruchi D.', 'ruchi@demo.com', 'e84de354f70c2
e1f.png'), User('Vibha SD.', 'vibha@demo.com', '918384c71939f75b.png'), User('Annu N.', 'anupama@demo.com', '722eed98ccee36fa.png'), User('Sne
ha M.', 'sneha@demo.com', 'default.jpg')]
>>> Post.query.all()
[Post('C Programming Language', '2020-12-27 05:17:28.264311'), Post('Discrete Mathematics', '2020-12-27 05:19:47.102213'), Post('Data Structur
es', '2020-12-27 05:22:12.517163'), Post('Object Oriented Programming', '2020-12-27 05:25:01.515785'), Post('Algorithms - Design & Analysis',
'2020-12-27 05:26:24.266261'), Post('Computer Architecture', '2020-12-27 05:28:11.116284'), Post('Operating System', '2020-12-27 05:30:36.7754
35'), Post('Database Management System', '2020-12-27 05:33:47.380374'), Post('Computer Network', '2020-12-27 05:35:10.365003'), Post('Compiler
', '2020-12-27 05:37:21.912512'), Post('Artificial Intelligence', '2020-12-27 05:39:20.641995'), Post('Cloud Computing', '2020-12-27 05:41:27.
808458'), Post('Data Science', '2020-12-27 05:43:05.106020'), Post('Information, Network and Cyber security', '2020-12-27 05:44:54.374122'), P
ost('Computer Graphics ', '2020-12-27 05:46:09.708261'), Post('Machine Learning', '2020-12-27 05:48:09.739031'), Post('Deep Learning', '2020-1
2-27 05:49:22.638845'), Post('Graph theory', '2021-01-13 07:27:58.244724')]
```

E. Tree Structure of Application Directory

```
Flask_Board/
├── flaskboard
│   ├── forms.py
│   ├── __init__.py
│   ├── models.py
│   ├── posts.json
│   ├── __pycache__
│   │   ├── forms.cpython-38.pyc
│   │   ├── forms.cpython-39.pyc
│   │   ├── __init__.cpython-38.pyc
│   │   ├── __init__.cpython-39.pyc
│   │   ├── models.cpython-38.pyc
│   │   ├── models.cpython-39.pyc
│   │   ├── routes.cpython-38.pyc
│   │   └── routes.cpython-39.pyc
│   ├── routes.py
│   ├── site.db
│   ├── static
│   │   ├── bgimg.jpeg
│   │   ├── image1.jpeg
```



AUTHORIZATION

A. Need of RBAC based authorization

Role-based access control (RBAC) is a method which regulates access to computer or network resources based on the roles of individual users within an organization. In the RBAC model an authorization to access components in the system is provided to eligible users based on which roles that user has.

Role Based Access Control (RBAC) model has a capability to manage a wide range of important access control policies. The range includes mandatory access control policies and discretionary access control policies and also the Principle of Least Privilege and the well-known Separation of Duty (SOD) based policies. The components of RBAC are role-permissions, user-role and role-role relationships which make it simple to perform user assignments. When compared to other authorization models the main advantages of RBAC model are ease of system administration and convenience in authorization information management.

In an RBAC model, the functions define the role within a given organization, further based on the roles authorizations are permitted, rather than single users. The authorizations granted to a role are strictly related to the data objects and resources that are needed for executing the functions associated with.

Access control policy is embodied in RBAC components such as role-permission, user-role,and role-role relationships. These components collectively determine whether a particular user is allowed access to a certain piece of system data. RBAC components can be configured directly by the system administrator or indirectly by appropriate roles as delegated by the system administrator. The policy enforced in a given system results from the specific configuration of RBAC components as directed by the system administrator. Because the access control policy can, and usually does, change over the system life cycle, RBAC offers a key benefit through its ability to modify access control to meet changing organizational needs.

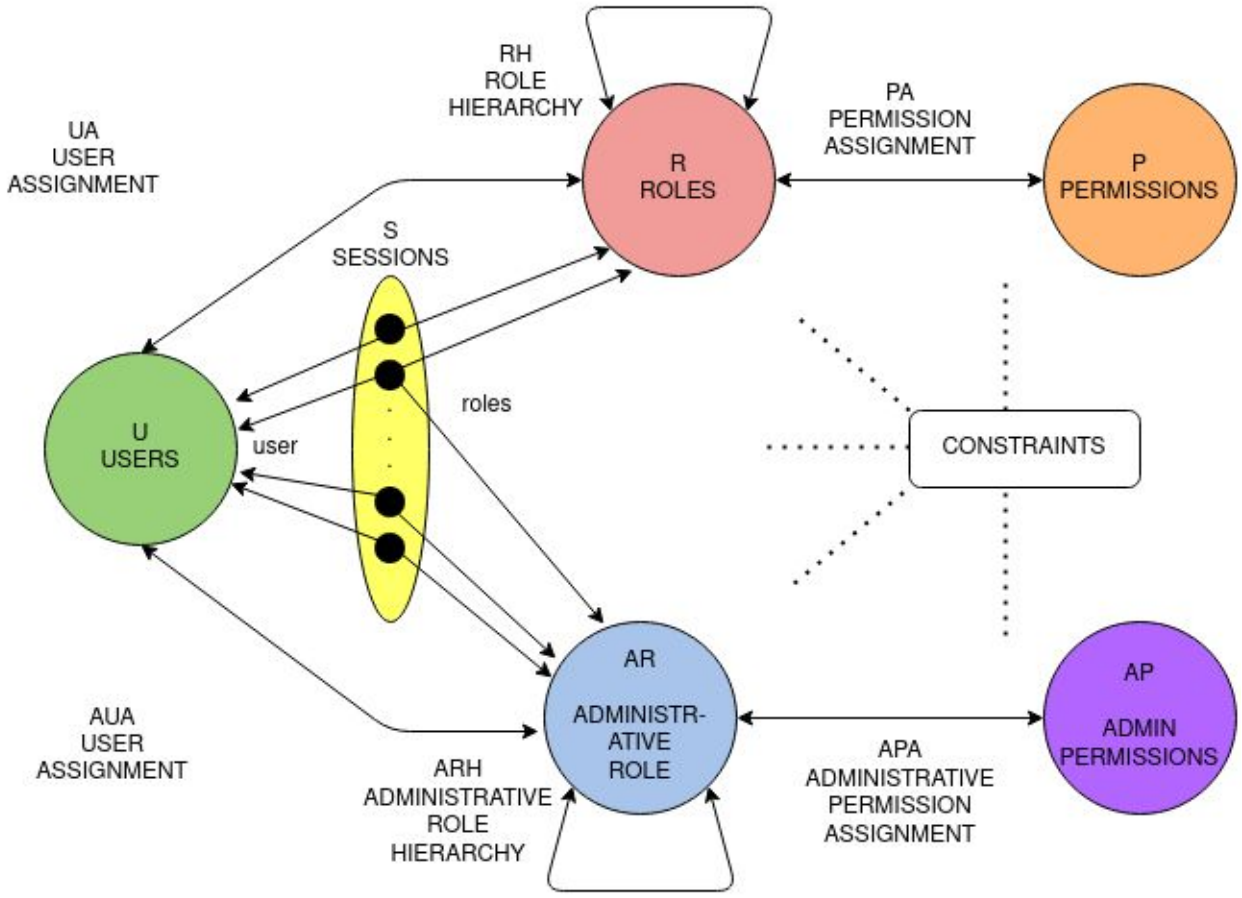
B. Component of RBAC present in the Application

In the application, there is one role i.e. Blogger. It has few permissions associated with it and few constraints related to them. Permissions to Role are assigned in the system. Each user in the system is assigned a role i.e the system does User to Role assignment.

C. Component of Administrative model present in the Application

It is worth emphasizing that RBAC distinguishes roles and permissions from administrator roles and permissions, respectively, where the latter are used to manage the former. RBAC has many components. Administration of RBAC involves control over each of these components, including creation and deletion of roles,creation and deletion of permissions, assignment of permissions to roles and their removal, creation and deletion of users, assignment of users to roles and their removal, definition and maintenance of the role hierarchy,definition and maintenance of constraints, and all of these in turn for administrative roles and permissions. In the application, there is an admin role with administrative permissions assigned to it.

Summarizing the existence of RBAC and Administrative RBAC in the application “ Blogger’s Stop ”.

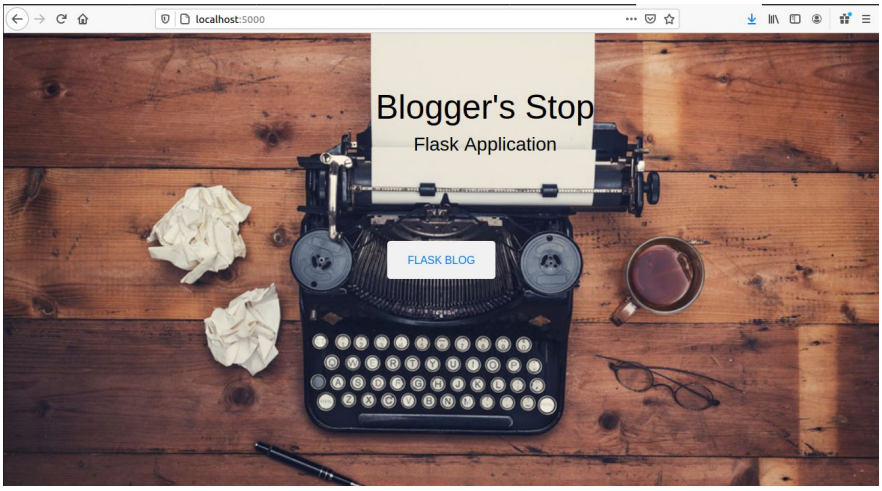


The different components of RBAC in the blog application is as listed below:

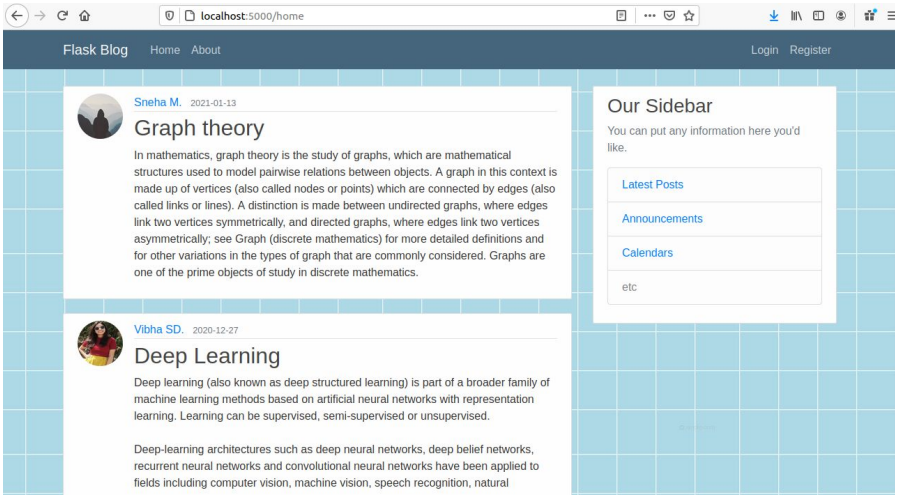
- USERS, U - Normal Client
- ROLES, R - Blogger
- PERMISSIONS, P - The blogger is allowed to register, login, create new blog, view all the blogs, update and delete its own blogs.
- SESSION, S - Each session maps to a single user.
- ADMINISTRATIVE ROLE, AR - Admin
- ADMIN PERMISSIONS, AP - The admin is allowed to login and has the authority to view all the blogs, delete the blog of all the users, access and update the account details, etc.

IMPLEMENTATION

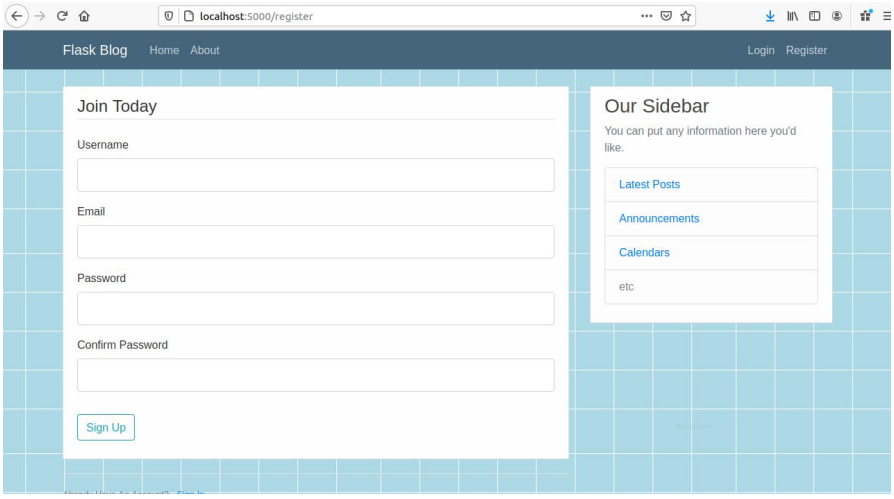
1. USER INTERFACE



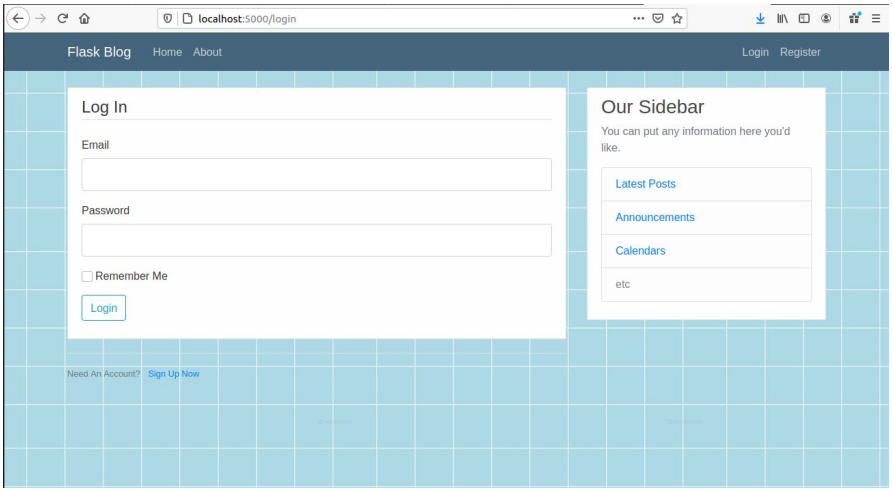
2. HOME PAGE



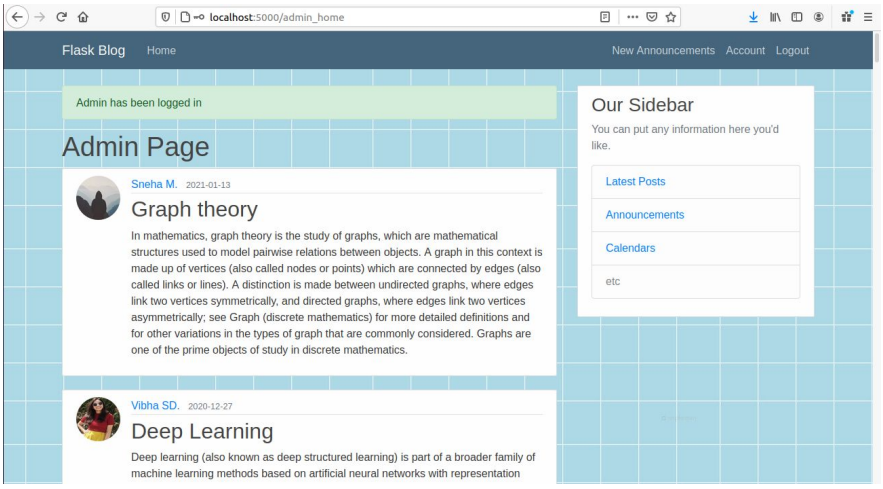
3. SIGN UP FORM



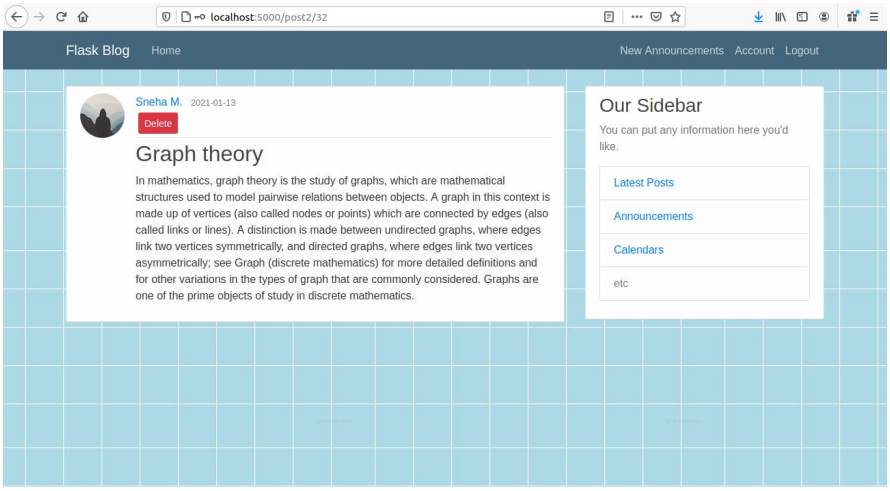
4. LOGIN INTERFACE



5. ADMIN'S AUTHORITY



Admin Home Page where he can view all the blogs posted.



Admin can delete the blog of any user.

CONCLUSION

The application has successfully utilized the components of the RBAC model to make it easier to assign the permissions to users by assigning them roles. These roles have permissions associated with them which can only be given to the user having the role to perform these tasks. The application can be further improved by adding more functionality to it if the need arises. More roles can be added if required and permissions are assigned to these roles. RBAC model makes it very convenient to keep track of all the permissions assigned to a user by assigning these permissions to the roles.

REFERENCES

1. https://palletsprojects.com/p/flask/
2. The ARBAC97 Model for Role-BasedAdministration of Roles , RAVI SANDHU, VENKATA BHAMIDIPATI, and QAMAR MUNAWERGeorge Mason University.
3. Ravi S. Sandhu, Edward J Coyne " Role-Based access control models".
4. https://profsandhu.com/articles/advcom/adv_comp_rbac.pdf
5. https://profsandhu.com/conference_papers.htm