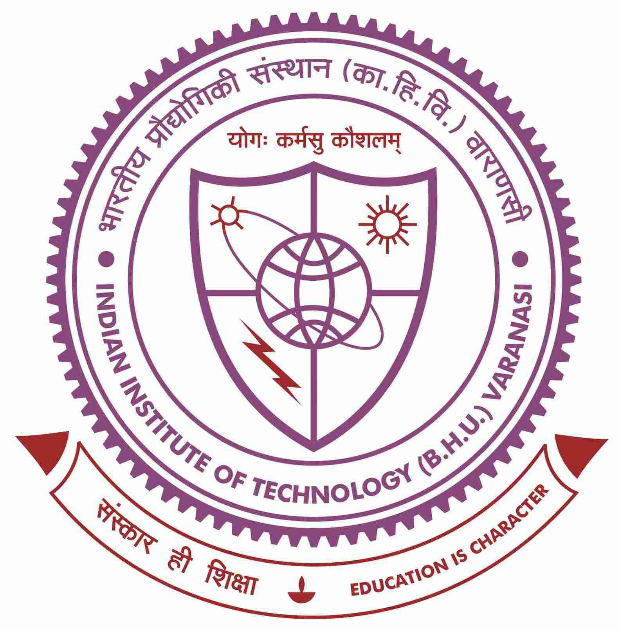
DESIGN AND INNOVATION HUB PROJECT

“Library Management System”



**Under Supervision of:**

Dr. Ashok Jaiswal

Department of Mining Engineering

IIT (BHU) VARANASI

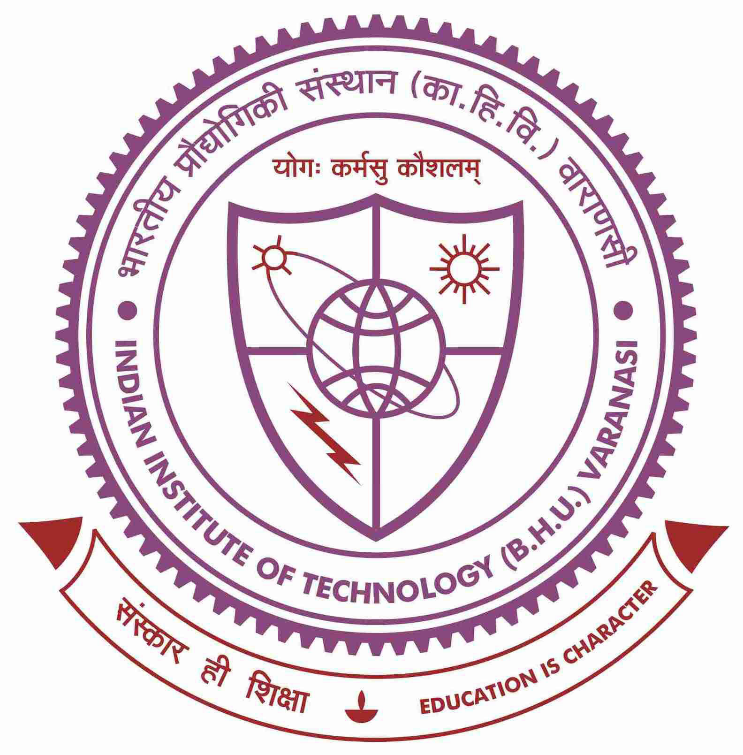
**SUBMITTED BY:**

Rishabh Acharya Arun Marothiya Vishvendra Singh Dagur

Roll No.15155063 Roll No.15155017 Roll no.15155091

B. Tech Part-III B. Tech Part-III B. Tech Part-III

“Library Management System”

****

**PROJECT APPROVED BY:**

Dr. A K Singh

Department of computer Science and Engineering

IIT (BHU), VARANASI

**FORWARDED BY:**

Dr. Ashok Jaiswal

Department of Mining Engineering

IIT (BHU) VARANASI

# CERTIFICATE

This is to certify that the project entitled “Library Management System” is a bonafide record of Design and innovation hub project work done by Rishabh Acharya (Roll No: 15155063), Arun Marothiya (Roll no. 15155017), Vishvendra Singh Dagur (Roll no. 15155091) under my supervision and submitted to Design and innovation hub, Indian Institute of Technology (BHU), Varanasi.

**Dr. Ashok Jaiswal**

**Department of Mining Engineering**

**IIT (BHU) VARANASI**

**ACKNOWLEDGEMENTS**

We wish to express our deep sense of gratitude to Dr. Ashok Jaiswal, Professor, Department of Mining Engineering IIT (BHU) Varanasi for introducing the present topic and for their inspiring guidance and valuable suggestions throughout the project work.

We are also thankful and express our gratitude towards our friends who at various stages had lent a helping hand.

Rishabh Acharya Arun Marothiya

Roll No. – 15155063 Roll No. – 15155017

B. Tech Part III B. Tech Part III

Mining Engineering Mining Engineering

IIT (BHU) Varanasi IIT (BHU) Varanasi

Vishvendra Singh Dagur

Roll No.—15155091

B. Tech Part III

Mining Engineering

IIT (BHU) Varanasi

**CONTENT**

1. Introduction
   1. Scope Of Library
   2. Requirements
   3. Project Aim And Objective
2. Required Languages for Project
   1. Front-End
   2. Back-End
3. Tasks Of Librarian
   1. Log in
   2. Check Book and Student Status
   3. Add New Book
   4. Add New Student
   5. Book List
   6. Issue Book
   7. Return Book
   8. View Book Detail
   9. List of Books and Students
   10. . Update Book
4. Database Structure
5. Url Structure
6. Conclusion
7. **INTRODUCTION**

A library management system, also known as an automated library system is software that has been developed to handle basic housekeeping functions of a library. It’s a well organized software solution for a library. It helps to provide information on any book present in library to the user as well as staff member. It keeps a track of book issued, return and added to library.

* 1. **Scope Of Library :-**

To make the existing system more efficient and provide a user friendly environment where user can be serviced better. Make functioning of library faster and provide a system where the library staff can catch defaulters and not let them escape. And also minimize the loss done to books.

* 1. **Requirements :-**

The library management system shall be required to maintain information about its users and books. It shall store databases for students, teachers and books. The student database stores information about a student’s roll number, name, address, course and year. The book database store information about a book title, author, publisher, cost, bill number, year of publishing and pages. The teacher database stores information about a teacher’s id, name, department, designation, address and telephone number.

* 1. **Project Aim and Objective :-**

To eliminate the paper work in library and record every transaction in computerized system so that problem such as record file missing won’t be happen. Also design a user friendly graphical user interface which suit the user and save the cost and time.

1. **Required Languages For Project :-**
   1. **Front-End**

**Front**-**end web development** is the practice of producing HTML, CSS and JavaScript for a website or **Web** Application so that a user can see and interact with them directly. The challenge associated with front-end development is that the tools and techniques used to create the front end of a website change constantly.

The objective of designing a site is to ensure that when the users open up the site they see the information in a format that is easy to read and relevant. This is further complicated by the fact that users now use a large variety of devices with varying screen sizes and resolutions thus forcing the designer to take into consideration these aspects when designing the site. They need to ensure that their site comes up correctly in different browsers (cross-browser), different operating systems (cross-platform) and different devices (cross-device), which requires careful planning on the side of the developer.

There are several tools available that can be used to develop the front end of a website, and understanding which tools are best fit for specific tasks marks the difference between developing a hacked site and a well designed, scalable site.

* + 1. HTML

Hyper Text Mark-up Language (HTML) is the backbone of any website development process, without which a web page doesn't exist. Hypertext means that text has links, termed hyperlinks, embedded in it. When a user clicks on a word or a phrase that has a hyperlink, it will bring another webpage. A markup language indicates text can be turned into images, tables, links, and other representations. It is the HTML code that provides an overall framework of how the site will look. HTML was developed by [Tim Berners-Lee](https://en.wikipedia.org/wiki/Tim_Berners-Lee). The latest version of HTML is called [HTML5](https://en.wikipedia.org/wiki/HTML5) and was published on October 28, 2014 by the W3 recommendation. This version contains new and efficient ways of handling elements such as video and audio files.

* + 1. CSS

[Cascading Style Sheets](https://en.wikipedia.org/wiki/Cascading_Style_Sheets) (CSS) controls the presentation aspect of the site and allows your site to have its own unique look. It does this by maintaining style sheets which sit on top of other style rules and are triggered based on other inputs, such as device screen size and resolution.

* + 1. BOOTSTRAP

Bootstrap is a [free and open-source](https://en.wikipedia.org/wiki/Free_and_open-source_software) front-end [web framework](https://en.wikipedia.org/wiki/Web_framework) for designing [websites](https://en.wikipedia.org/wiki/Website) and [web applications](https://en.wikipedia.org/wiki/Web_application). It contains [HTML](https://en.wikipedia.org/wiki/HTML)- and [CSS](https://en.wikipedia.org/wiki/CSS)-based design templates for [typography](https://en.wikipedia.org/wiki/Typography), forms, buttons, navigation and other interface components, as well as optional [JavaScript](https://en.wikipedia.org/wiki/JavaScript) extensions. Unlike many web frameworks, it concerns itself with [front-end development](https://en.wikipedia.org/wiki/Front-end_web_development) only.

* + 1. JavaScript

If you could only learn one language in your lifetime, you’d be well-advised to choose JavaScript. Though it’s not exclusively a frontend language, that’s where it’s most commonly used. JavaScript is a language that is run on a client machine, i.e. a user’s computer. This means that JavaScript can be used to program fast, intuitive and fun user experiences, without forcing a user to refresh their web page. Drag-and-drop, infinite-scroll and videos that come to life on a web page can all be programmed with JavaScript. JavaScript is so popular that entire frameworks have been built just to make building application frontends easier. Frameworks like Angular, Ember, React and Backbone are all very widely used for JavaScript-heavy frontends.

* + 1. JQUERY

**jQuery** is a [cross-platform](https://en.wikipedia.org/wiki/Cross-platform) [JavaScript library](https://en.wikipedia.org/wiki/JavaScript_library) designed to simplify the [client-side scripting](https://en.wikipedia.org/wiki/Client-side_scripting) of [HTML](https://en.wikipedia.org/wiki/HTML). It is [free, open-source software](https://en.wikipedia.org/wiki/Free_and_open_source_software) using the permissive [MIT License](https://en.wikipedia.org/wiki/MIT_License).[[2]](https://en.wikipedia.org/wiki/JQuery#cite_note-jqorg-license-2) [Web](https://en.wikipedia.org/wiki/World_Wide_Web) analysis indicates that it is the most widely deployed JavaScript library by a large margin.

* 1. **Back-End**

The backend of a web application is an enabler for a frontend experience. An application’s frontend may be the most beautifully crafted web page, but if the application itself doesn’t work, the application will be a failure. The backend of an application is responsible for things like calculations, business logic, database interactions, and performance. Most of the code that is required to make an application work will be done on the backend. Backend code is run on the server, as opposed to the client. This means that backend developers not only need to understand programming languages and databases, but they must have an understanding of server architecture as well. If an application is slow, crashes often, or constantly throws errors at users, it’s likely because of backend problems.

* + 1. Python

Much like Ruby, [Python](https://www.upwork.com/hiring/development/python-programming-language/) is known in part for its readability. A popular web development framework, Django, also makes Python a popular choice for building web applications. Dropbox was built with – and still uses – Python to run much of its server-side code. Python can be used for everything Ruby can. Whether you use Python or Ruby depends on your skill set, the communities you’re exposed to, and your development team.

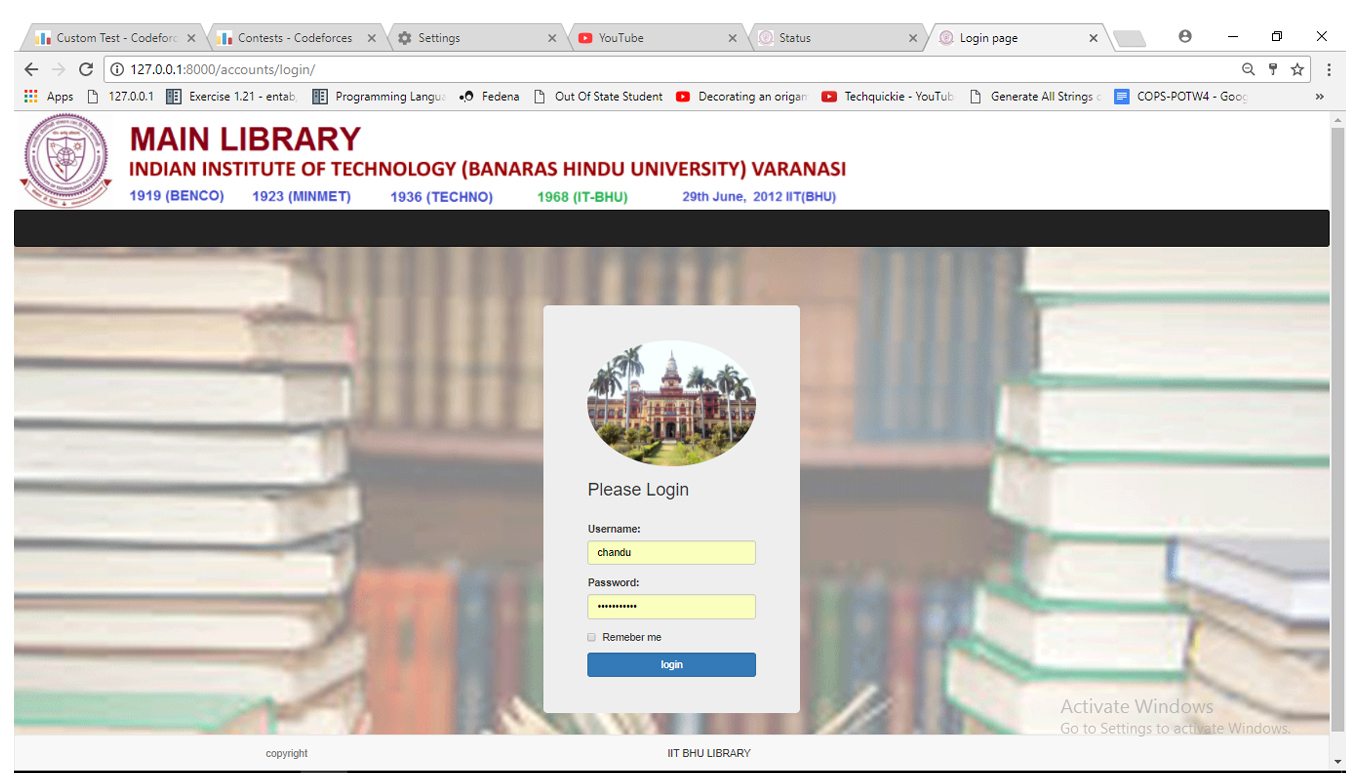
Django:- Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of Web development, so you can focus on writing your app without needing to reinvent the wheel.

* + 1. SQL

SQL is a universal database query language. SQL is used to interact with databases, which are a part of every backend web application. No matter what language or framework you choose to build your web application, you will likely use SQL, or some abstraction of it, to interact with the database.

1. **Functions :-**

3.1 Log In:



Here we made same interface for Staff user and student user category. It will automatically detect about login category and according to that it will show interface. To student category it will show functions-  
\* Book Status

\* Student status

\* Book List

\* Student List

Other functions like Issue Book, Return Book, add book, add student, Update Book, Delete Book are just for staff User.

def login(request):

c ={}

c.update(csrf(request))

return render\_to\_response('library/login.html',c)

def auth\_view(request):

username = request.POST.get('username','')

password = request.POST.get('password','')

user = auth.authenticate(username=username,password=password)

if user is not None:

auth.login(request,user)

return HttpResponseRedirect('/library/home/')

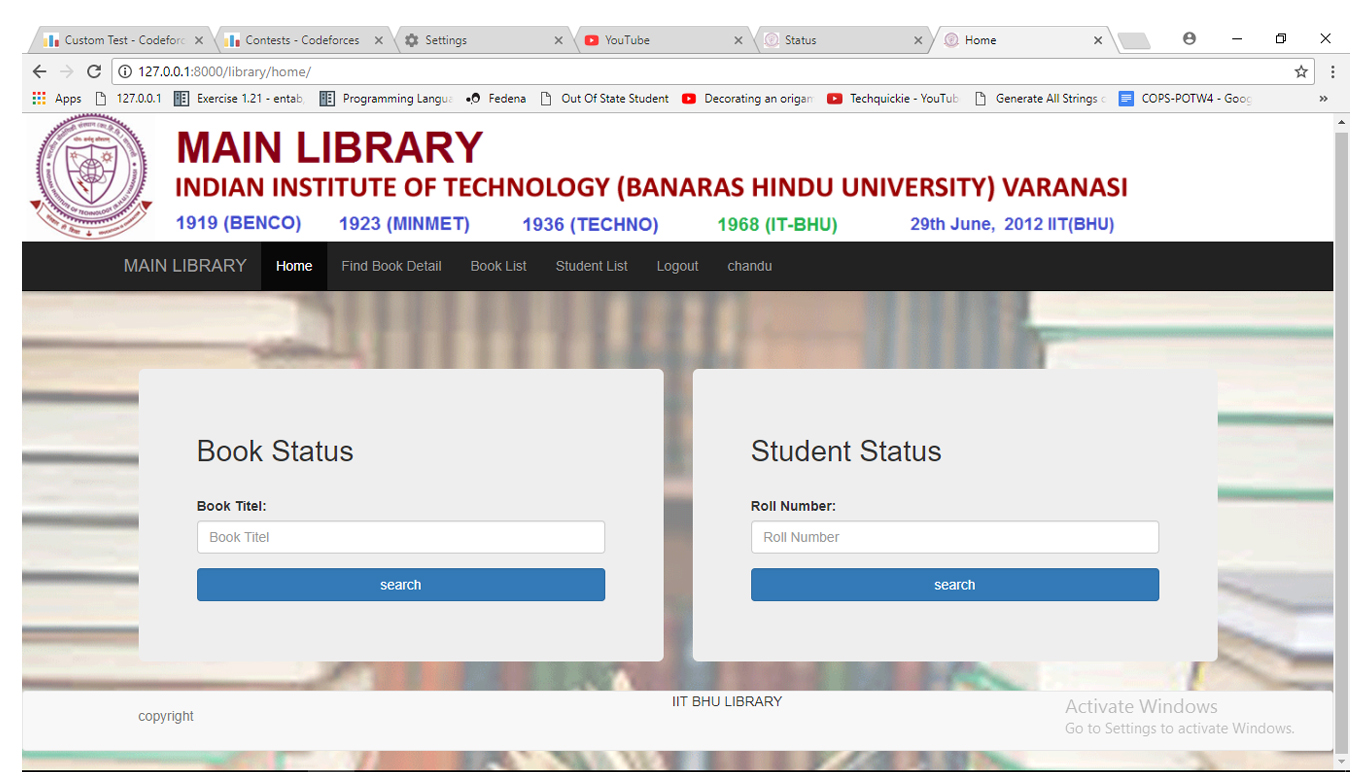
else:

c ={'error':"invalid"}

c.update(csrf(request))

return render\_to\_response('library/login.html',c)

3.2 Check Book and Student Status:



This function is available for both type of users. Student itself can check their status and also book status. Student can check their status at library by staff user also.

Book status function take Book title as input and show whole list of that books with student name who issued that book;

Student status function take Student roll number as input and show list of books which were issued by them. There is an authentication process which take care of all obstacle like wrong input or none data type input. If we enter wrong input or any roll number don’t match with our data base , it show related error.Hook status means checking about any book that aalso.itself can check their status and also book status. student

@login\_required(login\_url='/accounts/login/')

def bookstatus(request):

template=loader.get\_template('library/bookstatus.html')

book = request.POST.get('Book')

book\_objects=Book.objects.filter(title=book)

context={}

if book\_objects:

context ={'book\_object':book\_objects}

return HttpResponse(template.render(context, request))

else:

template = loader.get\_template('library/index.html')

context = {'error\_message': "book does not exist, please enter a valid detail"}

return HttpResponse(template.render(context,request))

@login\_required(login\_url='/accounts/login/')

def studentstatus(request):

template = loader.get\_template('library/studentstatus.html')

roll\_number = request.POST.get('roll\_number')

try:

student=Student.objects.get(roll\_number=roll\_number)

except:

template = loader.get\_template('library/index.html')

context = {

'error\_message': "entered student details is invalid, please enter a valid detail"}

return HttpResponse(template.render(context, request))

book\_objects = Book.objects.filter(student\_\_roll\_number=roll\_number)

context = {}

if book\_objects:

context = {'book\_objects': book\_objects}

return HttpResponse(template.render(context,request))

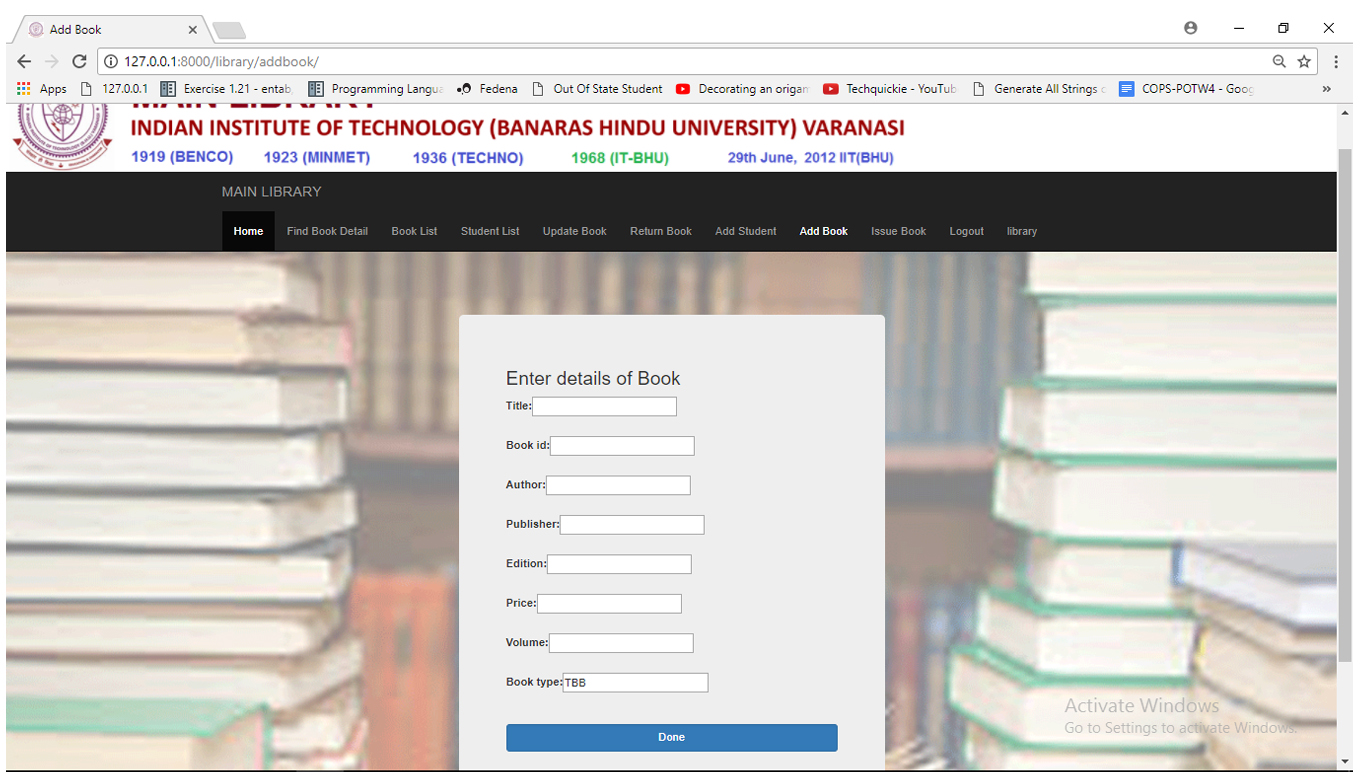
else:

template = loader.get\_template('library/index.html')

context = {'error\_message': "No book found"}

return HttpResponse(template.render(context, request))

3.3 Add New Book:



This function is developed to add new book in data base. As we may purchase more book for library so we have to add new book.

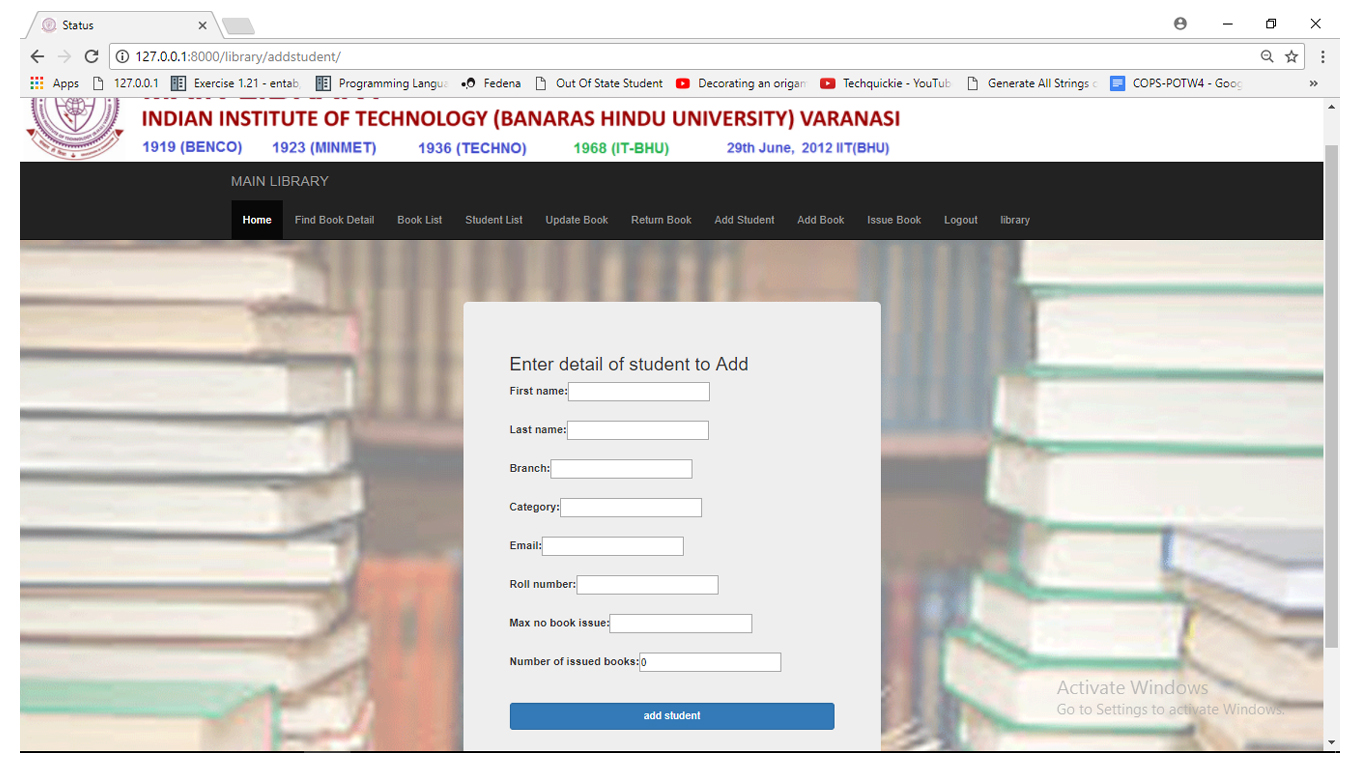
Here also an authentication process is developed. It checks about empty fields or wrong type of input. It generates error after submission, if there is any problem.

class BookCreate(LoginRequiredMixin,generic.CreateView):

model = Book

fields = ['title','book\_id','author','publisher','edition','price','volume','book\_type']

3.4 Add Student:



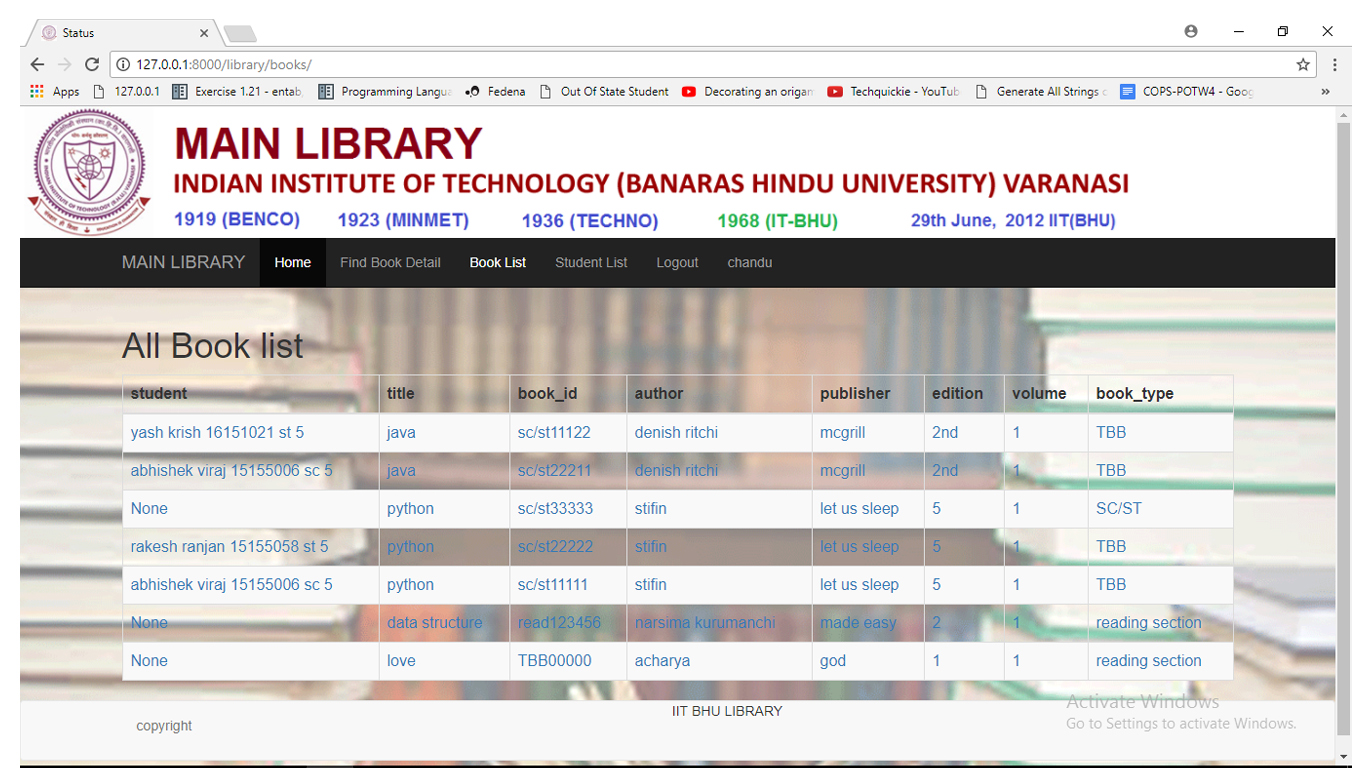
Same as book add this function is also developed.

class StudentCreate(LoginRequiredMixin,generic.CreateView):

model = Student

fields = ['first\_name','last\_name','branch','category','email','roll\_number','max\_no\_book\_issue','number\_of\_issued\_books']

3.5 Book List:



This function show whole books list with all title, book Id Author, Publisher, Edition , Volume, Book Type.

class Books(LoginRequiredMixin,generic.ListView):

template\_name = 'library/bookList.html'

def get\_queryset(self):

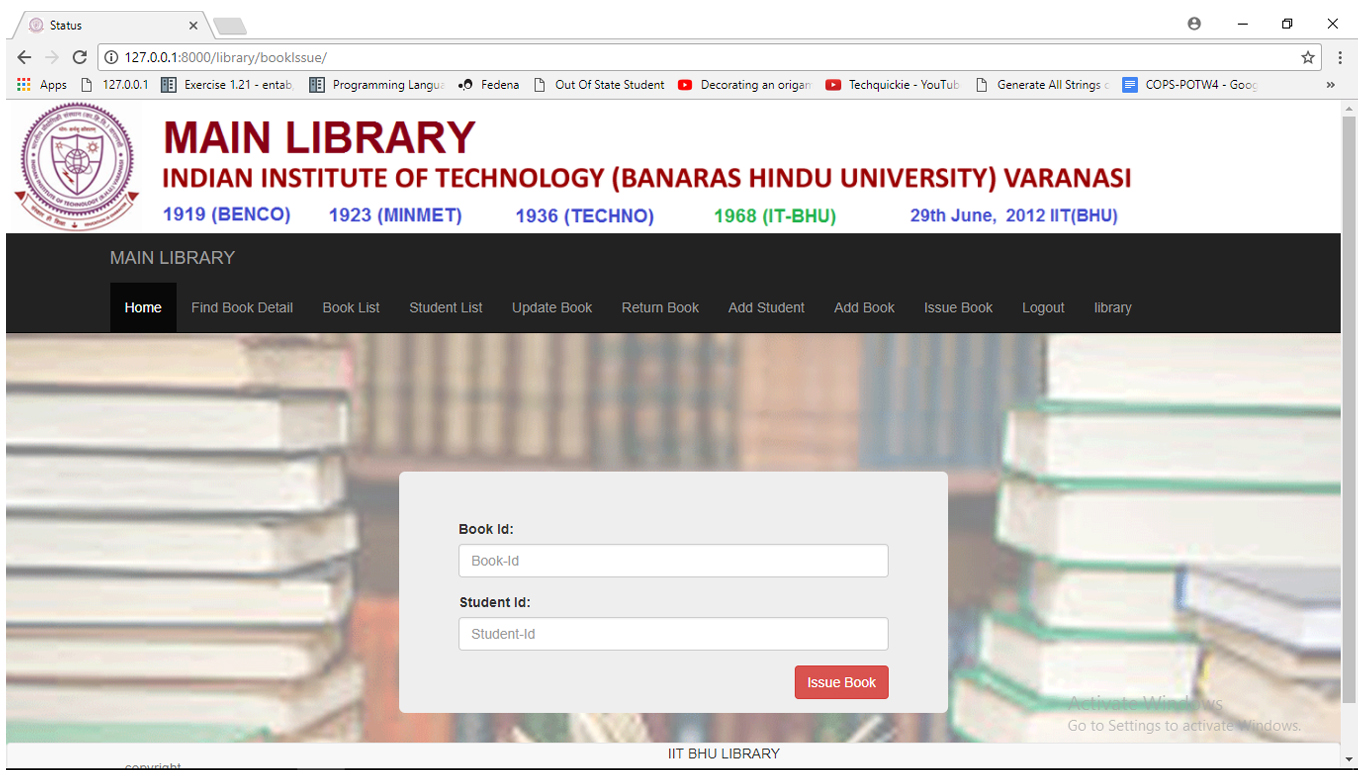
return Book.objects.all()

class BookDetail(LoginRequiredMixin,generic.DetailView):

model = Book

template\_name = 'library/bookDetail.html'

3.6 Issue Book :



This function is developed to issue book. It take two input , first book id and second student roll number and it issue book to that roll number. There is a authentication process which take care of all vulnerabilities.

@staff\_member\_required

@login\_required(login\_url='/accounts/login/')

def bookIssue(request):

template = loader.get\_template('library/bookIssue.html')

return HttpResponse(template.render(None, request))

@staff\_member\_required

@login\_required(login\_url='/accounts/login/')

def bookIssueAuth(request):

c={}

bookId = request.POST.get('bookId','')

studentRoll = request.POST.get('studentRoll','')

try:

book = Book.objects.get(book\_id=bookId)

student = Student.objects.get(roll\_number=studentRoll)

except :

template = loader.get\_template('library/bookIssue.html')

c = {

'error\_message': "you entered wrong value"

}

return HttpResponse(template.render(c, request))

if book.student is not None:

template = loader.get\_template('library/bookIssue.html')

c = {

'error\_message': "book is already issued to someone"

}

return HttpResponse(template.render(c, request))

if student.category=="SC" or student.category=="ST":

if book.book\_type=="SC/ST":

if student.number\_of\_issued\_books<5:

template = loader.get\_template('library/bookDetail.html')

book.student=student

student.number\_of\_issued\_books +=1

book.save()

student.save()

c = {

'book': book,

}

return HttpResponse(template.render(c, request))

else:

template = loader.get\_template('library/bookIssue.html')

c = {

'error\_message': "ypu have issued 5 books already"

}

return HttpResponse(template.render(c, request))

elif book.book\_type == "reading section":

template = loader.get\_template('library/bookIssue.html')

c = {

'error\_message': "sorry! this book belong to reading section. you cant issue it"

}

return HttpResponse(template.render(c, request))

else:

template = loader.get\_template('library/bookIssue.html')

c = {

'error\_message': "book don't belong to your catagory"

}

return HttpResponse(template.render(c, request))

elif student.category=="GENRAL" or student.category=="OBC":

if book.book\_type == "TBB":

if student.number\_of\_issued\_books < 3:

template = loader.get\_template('library/bookDetail.html')

book.student = student

student.number\_of\_issued\_books += 1

book.save()

student.save()

c = {

'book': book,

}

return HttpResponse(template.render(c, request))

else:

template = loader.get\_template('library/bookIssue.html')

c = {

'error\_message': "ypu have issued 3 books already"

}

return HttpResponse(template.render(c, request))

elif book.book\_type == "reading section":

template = loader.get\_template('library/bookIssue.html')

c = {

'error\_message': "sorry! this book belong to reading section. you cant issue it"

}

return HttpResponse(template.render(c, request))

else:

template = loader.get\_template('library/bookIssue.html')

c = {

'error\_message': "book don't belong to your catagory"

}

return HttpResponse(template.render(c, request))

else:

template = loader.get\_template('library/bookIssue.html')

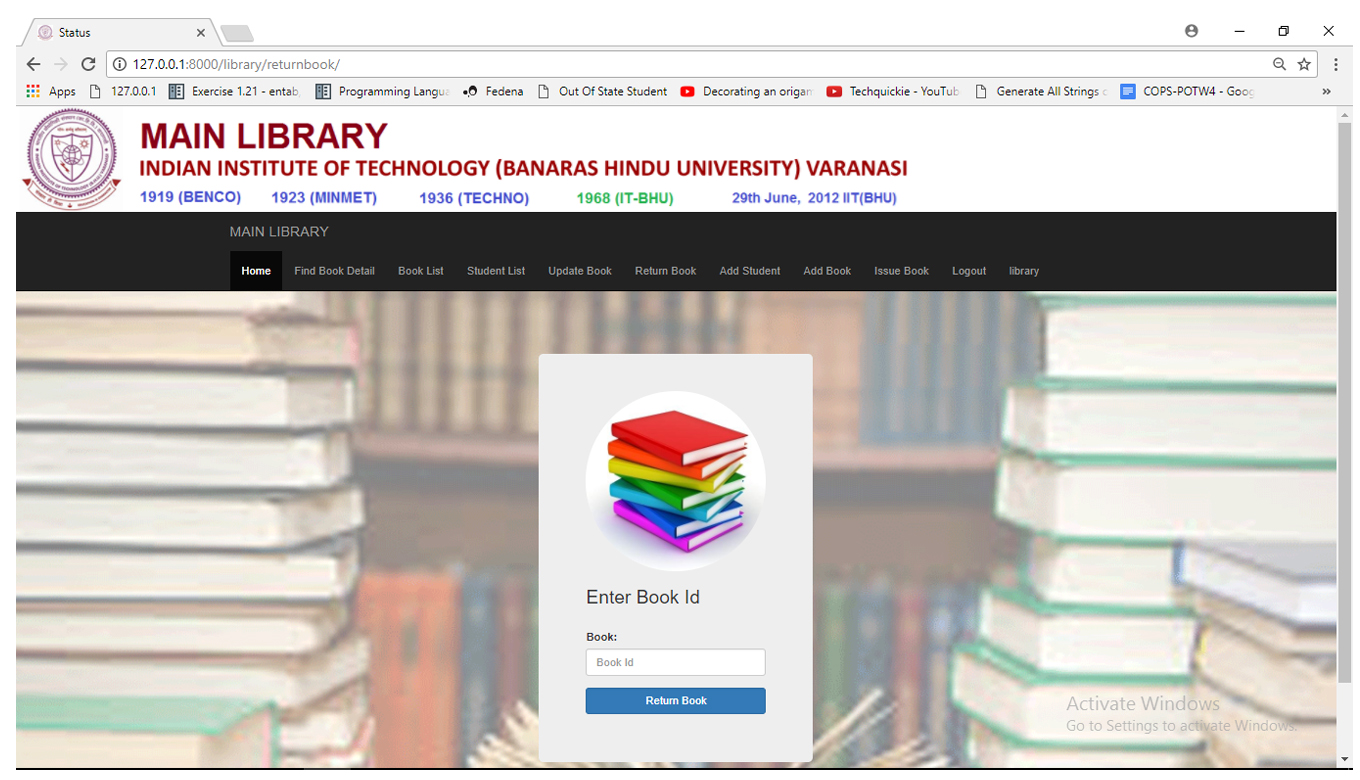
c = {

'error\_message': "there is some problem with this book please try again"

}

return HttpResponse(template.render(c, request))

3.7 Return Book:



This function just take book id and return book.

@staff\_member\_required

@login\_required(login\_url='/accounts/login/')

def bookReturn(request):

template = loader.get\_template('library/bookreturn.html')

return HttpResponse(template.render(None, request))

@staff\_member\_required

@login\_required(login\_url='/accounts/login/')

def bookReturnAuth(request):

c = {}

bookId = request.POST.get('bookId', '')

try:

book = Book.objects.get(book\_id=bookId)

except :

template = loader.get\_template('library/bookreturn.html')

c = {

'error\_message': "you entered wrong value"

}

return HttpResponse(template.render(c, request))

if book.student :

student=book.student

book.student=None

book.save()

student.number\_of\_issued\_books-=1

student.save()

template = loader.get\_template('library/studentDetail.html' )

c={

'student':student

}

return HttpResponse(template.render(c,request))

else:

template = loader.get\_template('library/bookreturn.html')

c = {

'error\_message': "book is not issued"

}

return HttpResponse(template.render(c, request))

3.8 View Book Detail:



@login\_required(login\_url='/accounts/login/')

def findBookdetail(request):

template = loader.get\_template('library/findBookDetail.html')

return HttpResponse(template.render(None, request))

@login\_required(login\_url='/accounts/login/')

def findBookdetailAuth(request):

c = {}

bookId = request.POST.get('bookId', '')

try:

book\_object = Book.objects.filter(book\_id=bookId)

except :

template = loader.get\_template('library/findBookDetail.html')

c = {

'error\_message': "you entered wrong value"

}

return HttpResponse(template.render(c, request))

if book\_object:

for book in book\_object:

template = loader.get\_template('library/bookDetail.html')

c = {

'book': book,

}

return HttpResponse(template.render(c, request))

else:

template = loader.get\_template('library/findBookDetail.html')

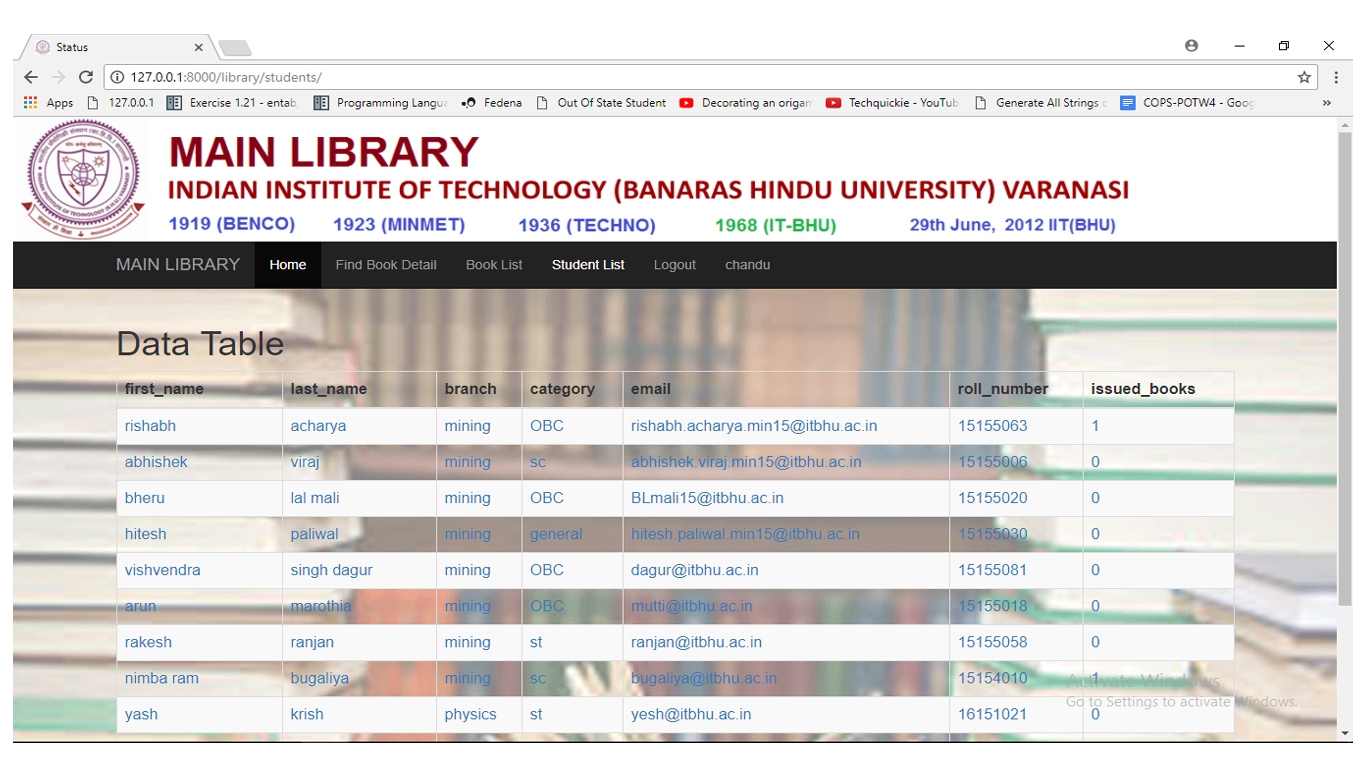
c = {

'error\_message': "you entered wrond book detail"

}

return HttpResponse(template.render(c, request))

3.9 Student List:



class StudentList(LoginRequiredMixin,generic.ListView):

template\_name = 'library/studentList.html'

def get\_queryset(self):

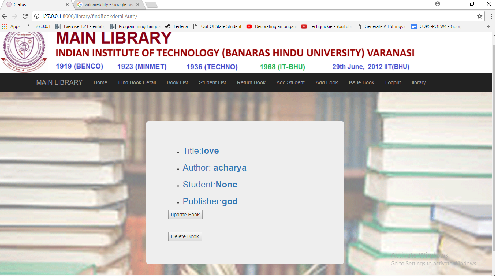
return Student.objects.all()

class StudentDetail(LoginRequiredMixin,generic.DetailView):

model = Student

template\_name = 'library/studentDetail.html'

3.10 Update Book :



class BookUpdate(LoginRequiredMixin,UpdateView):

model=Book

fields=['title','book\_id','author','publisher','edition','volume','book\_type','price']

class BookDelete(LoginRequiredMixin,DeleteView):

model=Book

success\_url=reverse\_lazy('library:books')

1. Database Structure:

from django.db import models

from django.core.urlresolvers import reverse

class Student(models.Model):

first\_name=models.CharField(max\_length=50)

last\_name = models.CharField(max\_length=50)

branch= models.CharField(max\_length=50)

category = models.CharField(max\_length=50)

email = models.EmailField()

roll\_number=models.IntegerField()

max\_no\_book\_issue=models.IntegerField()

number\_of\_issued\_books=models.IntegerField(default=int(0))

def get\_absolute\_url(self):

return reverse('library:studentDetail',kwargs={'pk' :self.pk})

def \_\_str\_\_(self):

return (self.first\_name+" "+self.last\_name +" "+ str(self.roll\_number)+" "+self.category+" "+str(self.max\_no\_book\_issue))

class Book(models.Model):

student = models.ForeignKey(Student,blank=True,null=True,default=None)

title=models.CharField(max\_length=50)

book\_id=models.CharField(max\_length=10)

author = models.CharField(max\_length=50)

publisher=models.CharField(max\_length=50)

edition=models.CharField(max\_length=10)

volume = models.IntegerField()

book\_type = models.CharField(max\_length=20,default="TBB")

price=models.IntegerField()

def get\_absolute\_url(self):

return reverse('library:bookDetail',kwargs={'pk' :self.pk})

def \_\_str\_\_(self):

return (self.title+" "+self.book\_id)

1. Url Structure:-

Library- portal app Url:-

from django.conf.urls import url

from . import views

from django.contrib.admin.views.decorators import staff\_member\_required

app\_name='library'

urlpatterns=[

# url(r'^$',views.index,name='index'),

url(r'^register/$' ,staff\_member\_required(views.UserFormView.as\_view()) ,name='register'),

url(r'^students/$' ,views.StudentList.as\_view() ,name='students'),

url(r'^students/(?P<pk>[0-9]+)/$' ,views.StudentDetail.as\_view() ,name='studentDetail'),

url(r'^books/$' , views.Books.as\_view() ,name='books'),

url(r'^books/(?P<pk>[0-9]+)/$' , views.BookDetail.as\_view() , name='bookDetail'),

url(r'^addbook/$' , staff\_member\_required(views.BookCreate.as\_view()) , name='AddBook'),

url(r'^addstudent/$' , staff\_member\_required(views.StudentCreate.as\_view()) , name='Addstudent'),

url(r'^home/$' , views.home,name='index'),

url(r'^bookstatus/$' , views.bookstatus),

url(r'^studentstatus/$' , views.studentstatus) ,

url(r'^bookIssue/$' , views.bookIssue, name='BookIssue') ,

url(r'^BookIssueAuth/$' , views.bookIssueAuth , name= 'BookIssueAuth') ,

url(r'^returnbook/$' , views.bookReturn , name='ReturnBook'),

url(r'^bookReturnAuth/$' , views.bookReturnAuth , name='bookReturnAuth'),

url(r'^findBookdetail/$' , views.findBookdetail , name='findBookdetail'),

url(r'^findBookdetailAuth/$', views.findBookdetailAuth, name='findBookdetailAuth'),

url(r'^update/(?P<pk>[0-9]+)/$', staff\_member\_required(views.BookUpdate.as\_view()), name='BookUpdate'),

url(r'^books/(?P<pk>[0-9]+)/delete/$' , staff\_member\_required(views.BookDelete.as\_view()) , name='BookDelete'),

]

Url structure at library app:-

from django.conf.urls import url,include

from django.contrib import admin

from django.contrib.auth import views as auth\_views

from . import views

app\_name='library\_portal'

urlpatterns = [

url(r'^admin/', admin.site.urls),

url(r'^library/',include('library.urls')),

#user auth urls

url(r'^accounts/login/$', views.login),

url(r'^accounts/auth/$', views.auth\_view),

url(r'^accounts/logout/$', views.logout),

]

1. **CONCLUSION**

* Checking book status and student status is easy.
* Managing library functions and make them easy.
* Students also check particular book status,availability of books.
* Non-paper database for library use.
* Book issue and return procedure is easy.