Automated Library

Aurors

Introduction

Automated Library is a software to ease the entire library-book issuing process contributing to overall standards of the library and ensuring transparent and smooth conduction of the library processes.

Problem Statement

An idea for an automated library is not easy to implement considering various parameters in mind which are essential for the overall functioning of the library and should be better and helping in terms of services and facilities.
Following pointers were kept in mind while designing the software.

- Book issue process should be simple. Currently, in many public libraries in India books are issued manually by making an entry in the register.
- Library Dues should be automatically updated and field must display fines corresponding a particular member.
- Since, the software is a web app, contributing to digital India, its search engine will be aligned with the **Microsoft bing search** and **Goodreads** to display information like book review, rating and cover picture.
- Web app will be able to generate automatic mail to its members whose books are due and are currently incurring fine.
- Since, a web app, the software would be made aesthetically appealing by exploiting HTML, CSS and JavaScript.
- Differential account facility, as a library is managed by multiple employs who own an individual account to issue books and limited excess as compared to the admin account.
- Detailed information of members and books issued by them which are linked both ways.
- Listing of members and books with search facility filtered in various categories.
- Member photo ID is displayed in member account from IIT Kanpur student database to ensure authenticity.

Implementation

In order to implement such a problem statement, it was agreed to upon to work on Django as it enables the use of forms while provides a front end in HTML with a back end in **Django framework** encoded in python.
In order to enhance the basic infrastructure of the app, various APIs were used including bing search in order to display relevant and updated text for the concerned book including its book review and rating.
Database is implemented via **MySql**.
Web app is deployed on Microsoft Azure Cloud Facility.

Models

Models were created to segregate data belonging to post i.e. books present in the library and the library members. Namely, there are two models:

- Member
- Post

Created applications

Separate apps were created to organize the entire functioning of the app. Namely there are following app:

- Member
- Catalogue
- django.contrib.admin

Member

It is responsible for the entire database of the registered library members, displaying, search, query and filtering including updating of member related variables with time.

Model.py

```
class Member(models.Model):
   Name=models.CharField(max_length=140)
   RollNo=models.CharField(max_length=140)
   EmailID=models.CharField(max_length=140)
   Slots=models.CharField(max_length=140)
   IssuedBy=models.CharField(max_length=140)
   date=models.DateTimeField()
   books = models.ManyToManyField(Post, null=True)
   Fine = models.IntegerField(default=0)
```

- **Linking issued book:**

  ```
  def save(self, *args, **kwargs):
        books = self.books.all()
        for book in books:
           if (book.memberid !=self.id):
              book.memberid = self.id
              book.member_Name =self.Name
              book.date = datetime.now(pytz.timezone('Asia/Kolkata'))
              book.hidden_date = datetime.now(pytz.timezone('Asia/Kolkata'))
              book.save()
        super(Member, self).save(*args, **kwargs)
  ```

- **Fine Calculation:** A book is due after fourteen days of its issue and after that it incurs a fine of one rupee per day. In order to implement this an extra variable hidden variable is declared which keep a track of difference in days between current date and date of issue.

```
for book in books:
          if (datetime.now(pytz.timezone('Asia/Kolkata')) - book.date).days >14:
            self.Fine += int((datetime.now(pytz.timezone('Asia/Kolkata')) -
               book.hidden_date).days)
            book.hidden_date = datetime.now(pytz.timezone('Asia/Kolkata'))
          book.save()
```

- **Reminder Mail:** An automatic reminder mail facility is included which dispatches a mail after due date to the concerned member.

```
for book in books:
          if (datetime.now(pytz.timezone('Asia/Kolkata')) - book.date).days >14:
            if (book.duestatus == 0):
              email = EmailMessage('Due', book.Title+'is due', to=[self.EmailID])
              email.send()
              book.duestatus = 1
          book.save()
```

Views.py

It consists of index view and detailed view which is displayed on the front end.

```
if query:
        if filter_q=="Name":
          queryset_list = queryset_list.filter(Name__icontains =query)
        elif filter_q=="RollNo":
          queryset_list = queryset_list.filter(RollNo__icontains =query)
```

Following code takes user input query to filter desired member details.

Catalogue

It is responsible for the entire database of the registered library bibliographies, displaying, search, query and filtering including updating of member related variables with time.

Model.py

Consists of all the user and default variables.

```
class Post(models.Model):
  CallNum=models.CharField(max_length=140)
  Barcode=models.CharField(max_length=140)
  Genre=models.CharField(max_length=140)
  Title=models.CharField(max_length=140)
  Author=models.CharField(max_length=140)
  date=models.DateTimeField(editable=False)
  hidden_date=models.DateTimeField(null=True , editable=False)
  member_Name =models.CharField(max_length=140, editable=False, null=True)
  memberid = models.IntegerField(editable=False, null=True)
  choices = ((0 , 'OK'), (1, 'DUE'))
  duestatus = models.IntegerField(choices = choices, default=0)
```

Views.py

It is responsible for linking books and members apart from parsing information from bing search and Goodreads API.

- **Search and Query**

```
if query:
        if filter_q=="Title":
           queryset_list = queryset_list.filter(Title__icontains =query)
        elif filter_q=="Author":
           queryset_list = queryset_list.filter(Author__icontains =query)
        elif filter_q=="Genre":
           queryset_list = queryset_list.filter(Genre__icontains =query)
        elif filter_q=="Call Num":
           queryset_list = queryset_list.filter(CallNum__icontains =query)
```

- **Parsing APIs** Book description, review, rating and book cover are parsed from Goodreads and bing.

```
url = 'https://api.cognitive.microsoft.com/bing/v5.0/images/search?q='
     urlend = '&count=2&offset=0&mkt=en-us&safesearch=Off'
     headers = {'Ocp-Apim-Subscription-Key': '9fd91d75528344cb9b983e7ca664adfd'}
     try:
          r = requests.get(url+(self.object).Title+' book'+urlend,headers=headers)
          results=(r.json())["value"][0]
          goodreads_url =
          'https://www.goodreads.com/book/title.xml?key=r1kcfqJjWCaypoMLJzPGw&title='+bTitle
          gd = ET.parse(urllib.urlopen(goodreads_url)).getroot()
          context['description'] = gd[1][16].text
          context['rating'] = gd[1][18].text
          context['image_url']= results
     except requests.exceptions.ConnectionError:
          status_code = "Connection refused"
```

dango.contrib.admin

It is responsible for providing differential login facilities for admin and various staff accounts who have limited access rights.