Project Documentation

OnBook.

Online Book Reading Platform

TYCS SEM 5
Minor Project

Made by **Anuj Rathore**
Under Guidance of **Dr.Charmy Patel**

A Software Engineering Project
On

OnBook | Online Book Reading Platform

as the partial fulfillment of

Semester -V study

for the degree of

Bachelor of Science (Computer Science)
(Affiliated to Veer Narmad South Gujarat University)

during the academic year 2022-2023.

Developed By:

Anuj Rathore

Guided By:

Dr. Charmy Patel

Sarvajanik Education Society

Shree Ramkrishna Institute of Computer Education and Applied Sciences

Behind P.T Science College, M.T.B. College Campus, Athwalines, Surat-395 001.

Sarvajanik Education Society

Shree Ramkrishna Institute of Computer Education and Applied Sciences

Behind P.T. Science College, M.T.B College Campus, Athwalines, Surat-395 001.

CERTIFICATE

DEPARTMENT OF COMPUTER SCIENCE

This is to certify that

Anuj Rathore *Exam no*

has / have successfully completed his / her / their software engineering project work entitled

OnBook | Online Book Reading Platform

as the partial fulfillment of

Semester -V study

for the degree of

Bachelor of Science (Computer Science)

(Affiliated to Veer Narmad South Gujarat University)

during the academic year 2022-2023.

_____ _____

Internal Project Guide **HOD**

Dept. of Computer Science

Date : _____ **Place : Surat**

University Examination:

Examination No. : _____

Signature of External Examiners : _____

_____

_____

INDEX for Project Documentation

1 | Introduction

System Profile

Project Title		OnBook | Online Book Reading Platform
Organization	|	Shree Ramakrishana Institute of Education and Applied Sciences
Application Type	|	Web Application
Front-End tool	|	HTML , CSS , JavaScript
Back-End tool	|	Python , Django Framework
Database	|	Sqllite 3
Operation System	|	Windows
Project Duration	|	2 Months
Project Guided By	|	Dr. Charmy Patel
Developed By	|	Anuj Rathore

2 | System Introduction

2.1 | System Definition, Objective and Scope

2.1.1 | System Information

| This project OnBook - Online Books reading Platform is develop for those student as well as those people who likes the book reading. This project solve the all problems of readers. In this Website Readers can make their profile on the website and add their favourite book in Saved Book Section and After login they can also able to Make a book request which is currently is not available on website

| We have Two modules in this website. First User Module or Reader Module and Second is Admin Module

| User can Search a book and sort book by it's category

| User Module In this module User can make their account on the website and After Register User / Reader can Login.

| After Login User can Edit their Profile , Change their password also can reset their password using email, when User click on Reset Password button Our system send a email with the reset password link

| After Login User or Readers can access Some additional Functionality of this Website User can add their favourite book in Saved book and Make a book request and Admin can approve or reject book request. If the request is accepted than with in 24 hours book will be uploaded on website

| Admin module in this module Admin can add a user and admin can also delete the user. He also Manage Book request of user he can accept or reject User book request

| Admin can add and remove a book from website

2.1.2 | Project Objective

- Whole Application is Under Supervision of admin Who Manages the users on website and upload books with the full information with little description
- Admin will upload the book which is demanded by user/reader. He has full right to accept or reject the request of user's book request.
- Reader can search books & filter book by category.
- Reader can create their profile and edit it later.
- Reader can track their book request.
- Reader can save favorite book in saved book section.

2.1.3 | Project Scope

- Only Admin can upload books on website,Only admin can see Book requests of Readers and Only he can Reject or Accept the request

.

- Admin can delete any reader or user
- User can Only allowed to edit their Details & change password.
- User only see their book they saved to read later..
- Reader can track their book request.
- Reader can save favorite book in saved book section.

2.2 | Hardware & Software Requirements

2.2.1 | Software Requirements

- Visual Studio Code
- Python 3
- Django
- Sqlite 3
- Web Browser
- Windows 7
- Figma (for UI Design)

2.2.1 | Hardware Requirements

- Processor | i3
- Hard Disk | 5GB
- RAM | 4 GB

3 | Requirement Analysis & Modeling

3.1 | Expected Work Of System

3.1.1 | Role Of Admin

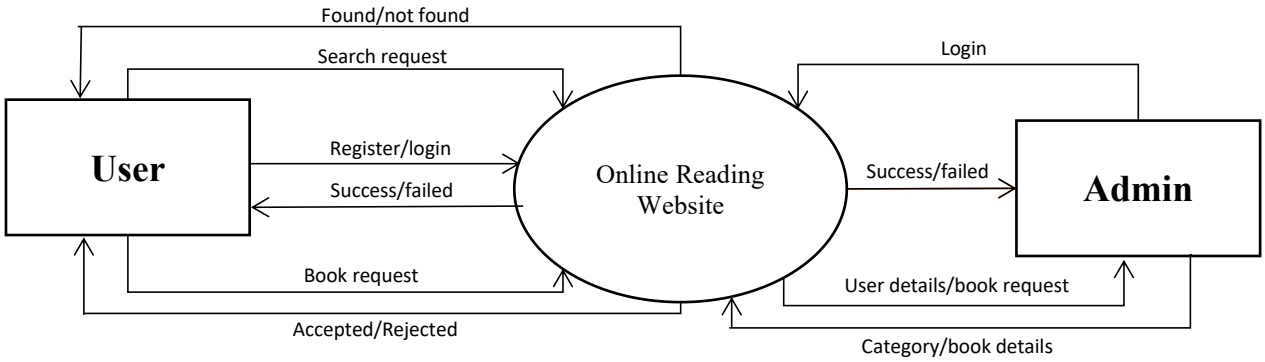
- Login/Logout
- Edit Profile
- Change Password & Reset Password
- Add Books On website
- View Users/Reader
- Delete Users/Reader
- Edit Users/Reader
- Manage User’s Book Request
- Accept or Reject User’s Book Request

3.1.1 | Role Of User Reader

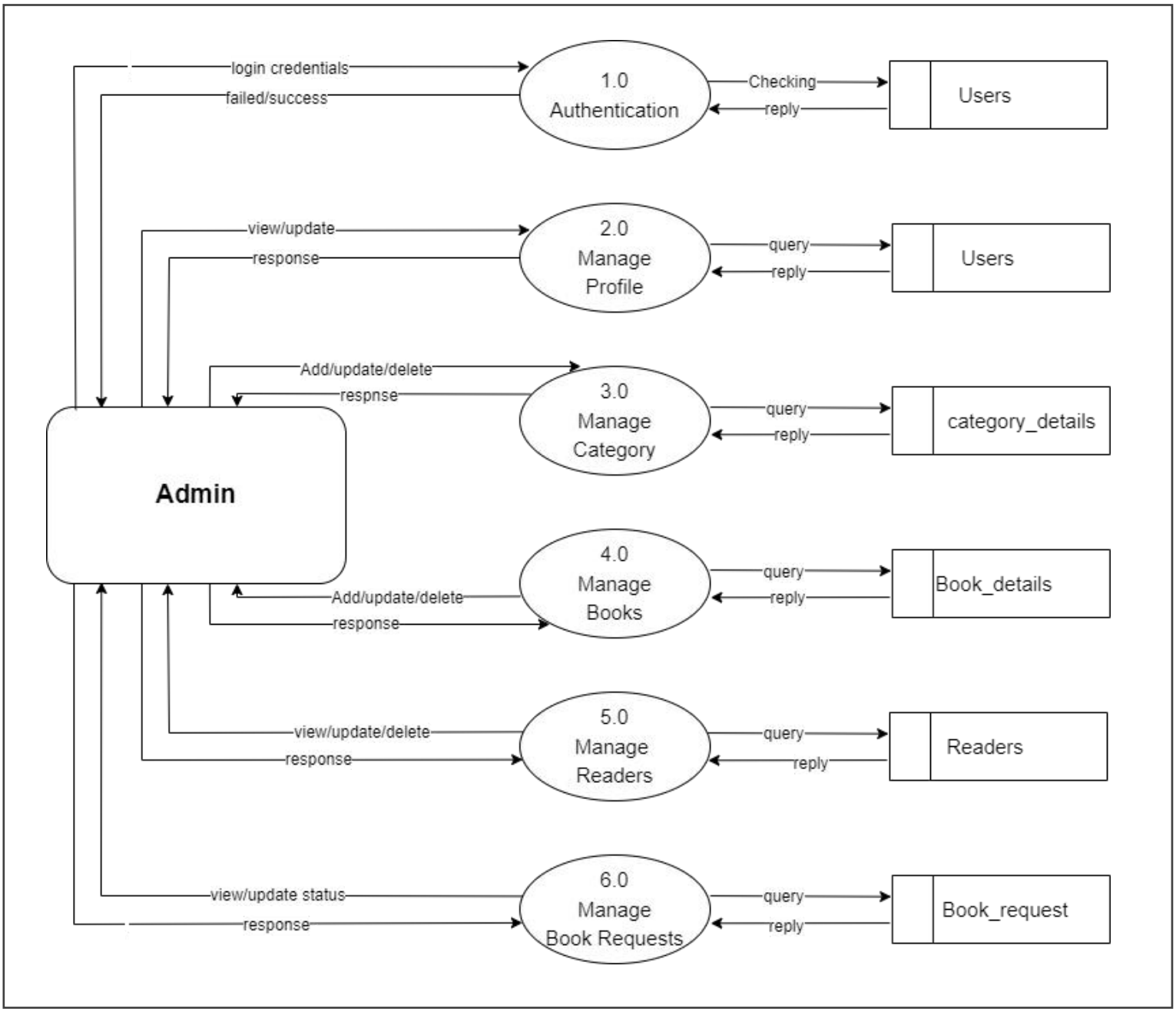
- Register
- Login/Logout
- Edit Profile
- Change Password
- Reset Password
- Save Books
- Request Books
- Read Books
- Search Books
- Filter Books by Category

3.2 | Data Flow Diagrams

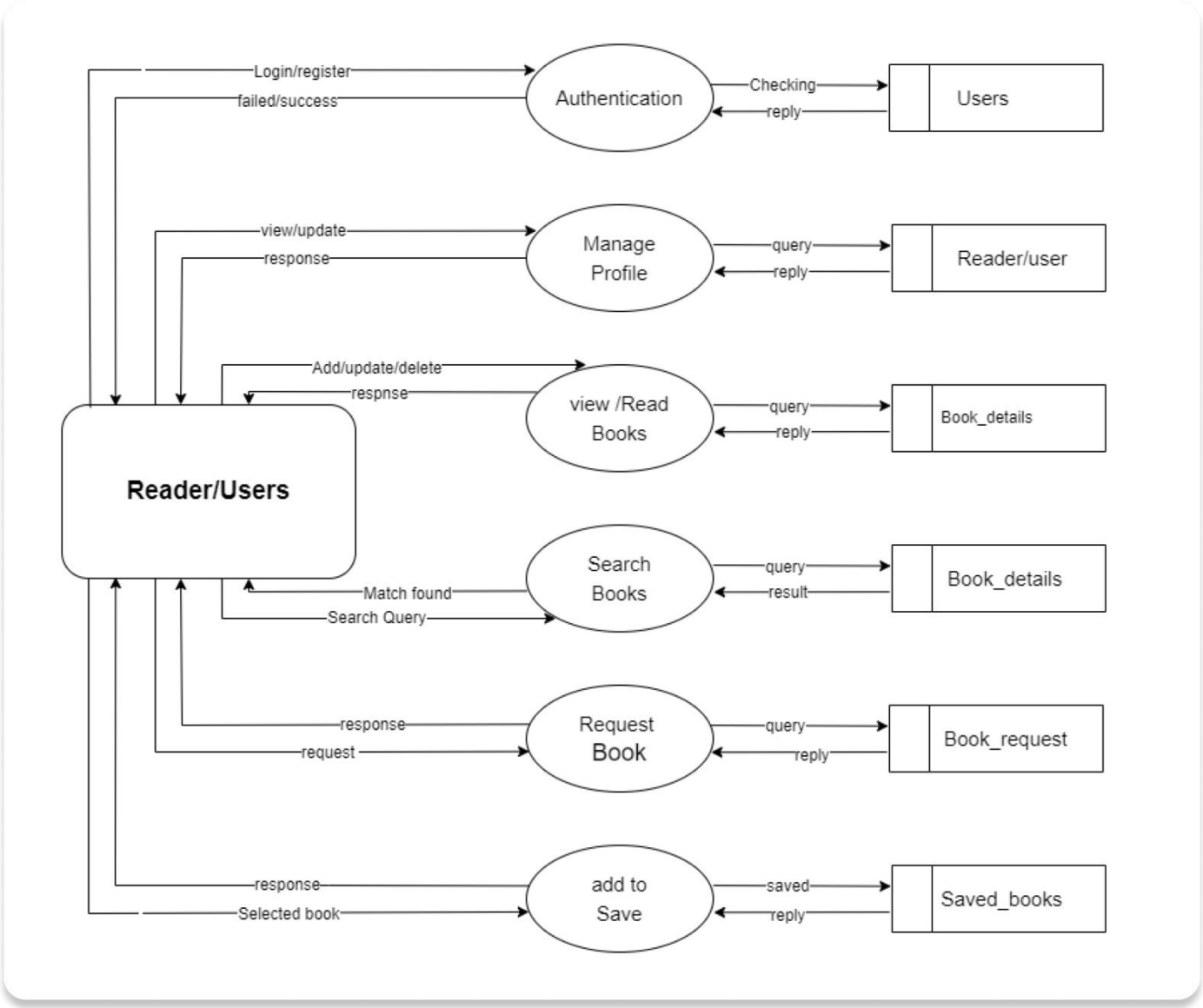
3.2.1 | 0 Level DFD



3.2.2 | 1st Level For Admin



3.2.2 | 1st Level For Users/Readers



3.3 | Process Specification

Process : 1.0	
Process Name	Login
Input	Username, password
Entity	Admin ,Readers
Database	Users, Readers
Description	In this process both Admin and User can login if the input correct username and password

Process : 2.0	
Process Name	Search book
Input	Query - book name , author name
Entity	Reader
Database	Book_detials
Description	In this process user can search a book and read

Process : 3.0	
Process Name	Edit Profile
Input	New User or Admin Details
Entity	Admin , Reader
Database	User, Reader
Description	In this process Admin and user can edit their profile

Process : 4.0	
Process Name	Add/update/delete Books
Input	Book details
Entity	Admin
Database	Book_detials
Description	Admin can add books, view books, update and delete from website

Process : 5.0	
Process Name	Add/update/delete Category
Input	Category_name ,category_img
Entity	Admin
Database	Category_details
Description	Admin can Add new category, update and delete

Process : 6.0	
Process Name	Book Request
Input	Book name , author name
Entity	Reader
Database	Book_request
Description	User can make any book request which is currently not on the website

Process : 7.0	
Process Name	Save book to saved book
Input	Select any book and click on save button
Entity	Reader
Database	Saved_book
Description	User can save any book they like

3.4 | Data Dictionary

| User Table

Name	User
Alias	-
Where used / How used	To store registration information of the user and admin also use For login for both admin and user.
Content Description	Username Password Email Is_superuser

| Reader Table

Name	Reader
Alias	-
Where used / How used	To Store additional information of user/reader such as gender ,city Profile picture
Content Description	reader_id(pk) , name , age, gender , city , user_img User_id

| Book Details Table

Name	Book_details
Alias	-
Where used / How used	Book Details table use for store book data and in books page show all books form database
Content Description	book_id(pk) , book_title, book_author, book_desc, book_page, book_lng, book_file, book_img book_upload_date, category_id(FK),

| Category Table

Name	Category
Alias	-
Where used / How used	Store all book category in book_details table category id is FK and data of this table used for shorting books by category.
Content Description	Category_id, category_name, category_img

| Saved Books Table

Name	Saved_books
Alias	-
Where used / How used	After user login user can save their favorite book to saved book and read and remove from saved book..
Content Description	category_id, category_name, category_img

| Book Request Table

Name	Book_request
Alias	-
Where used / How used	Store data of user request for any book. Which is not on the website user made request and he can check status of request
Content Description	Request_id, Book_name Author_name Req_status User_id

4 | Design

4.1 | ER Diagram & Database Design

4.1.1 | Database Design

User Table

Field Name	Datatype	Constraint
User_id	Int	Primary Key
Name	Char(200)	Not null
Username	Char(200)	unique
Email	Char(200)	Unique
Is_superuser	Tiny int	Not null

Reader Table

Field Name	Datatype	Constraint
Reader_id	Int	Primary Key
Name	Char(200)	-
Age	Int	-
Gender	Char(200)	-
City	Char(200)	-
User_img	Char(200)	-
User_id	Int	OneToOne Relation with User

Book Details Table

Field Name	Datatype	Constraint
Book_id	Int	Primary Key
Book_title	Char(200)	Not null
Book_author	Char(200)	Not null
Book_desc	Text	
Book_lng	Char(200)	
Book_file	Char(200)	
Book_img	Char(200)	
Book_upload_date	Date	
Category_id	Char(200)	Forgin Key

Category Table

Field Name	Datatype	Constraint
Category_id	Int	Primary Key
Category_name	Char(200)	-
Category_img	Char(200)	-

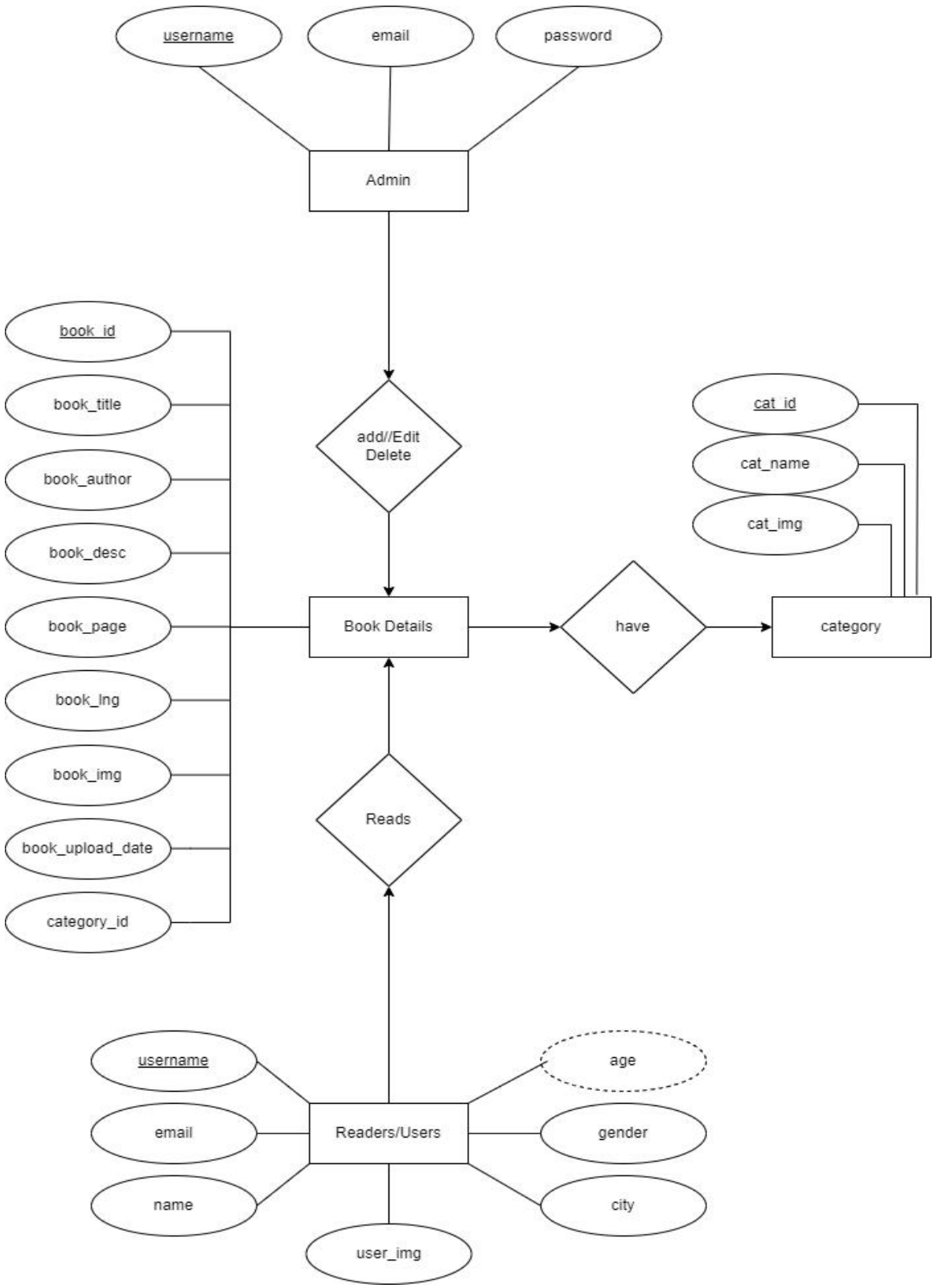
Saved Book Table

Field Name	Datatype	Constraint
Saved_book_id	Int	Primary Key
Book_id	Int	FK
User_id	Int	FK

Request book Table

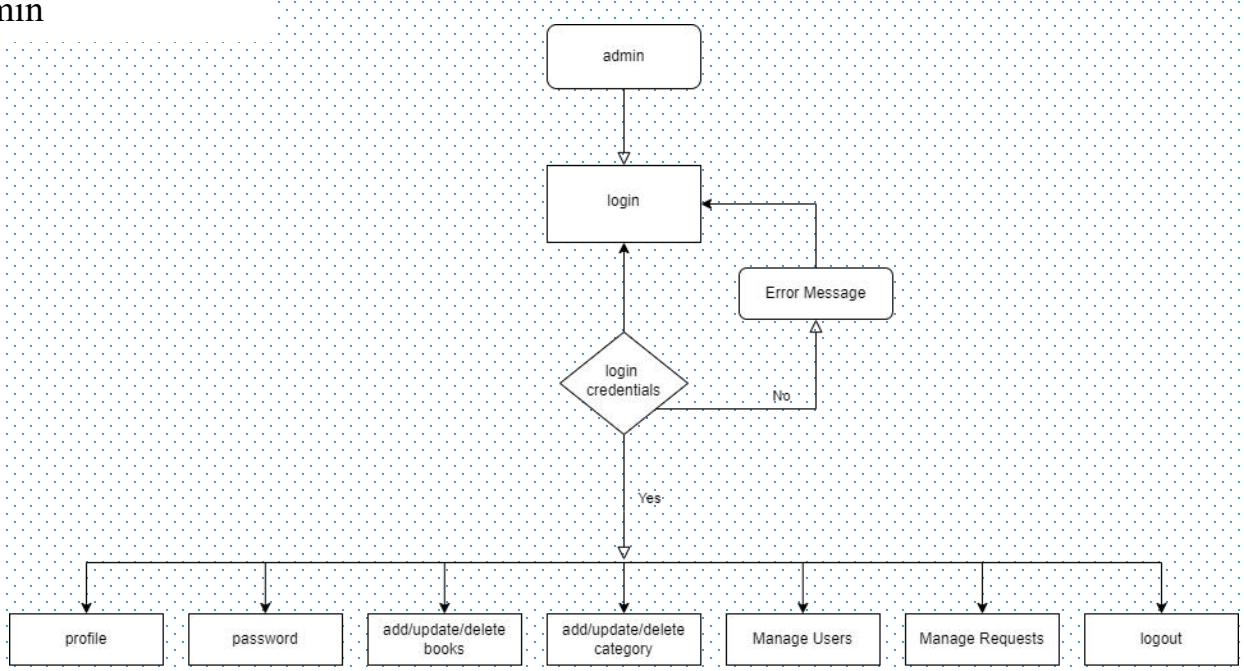
Field Name	Datatype	Constraint
Request_id	Int	Primary Key
Book_name	Char(200)	-
Author_name	Char(200)	-
Req_date	Datetime	Current_time
Req_status	Char(200)	-
User_id	Int	FK

4.1.1 | ER Diagram

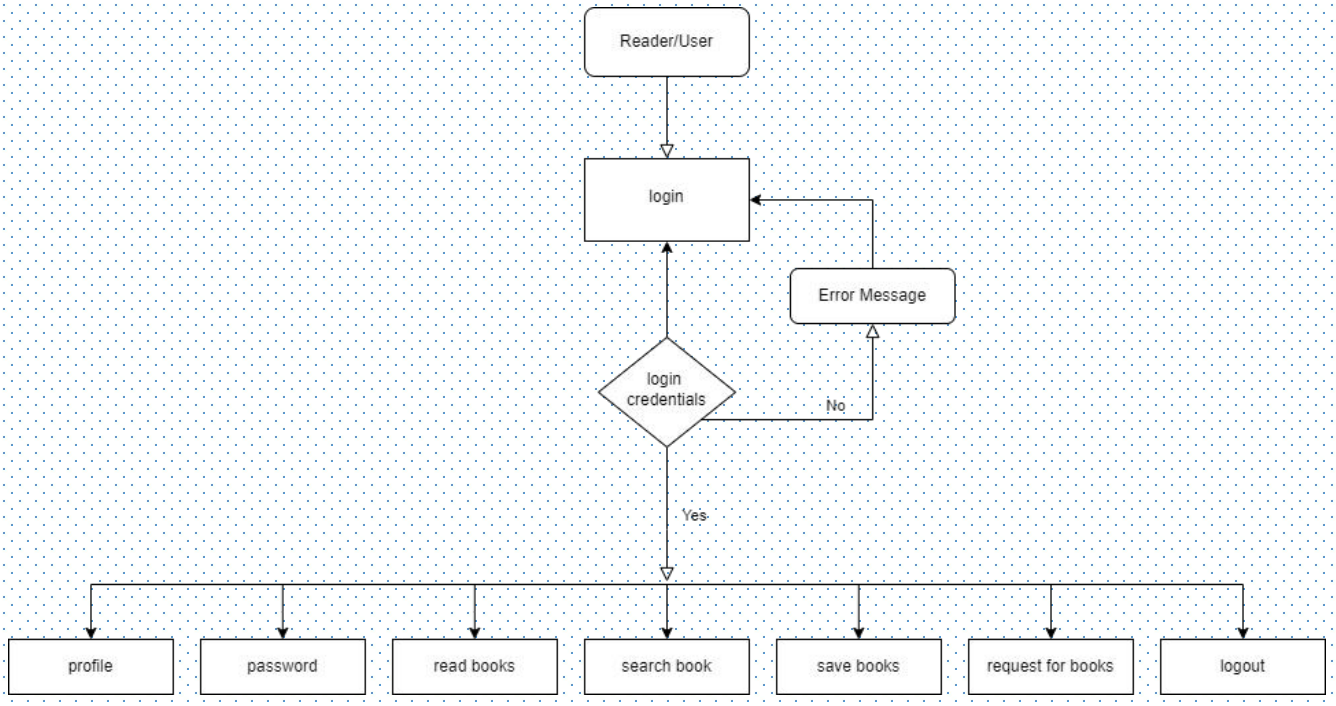


4.3 | System Flow Chart

Admin



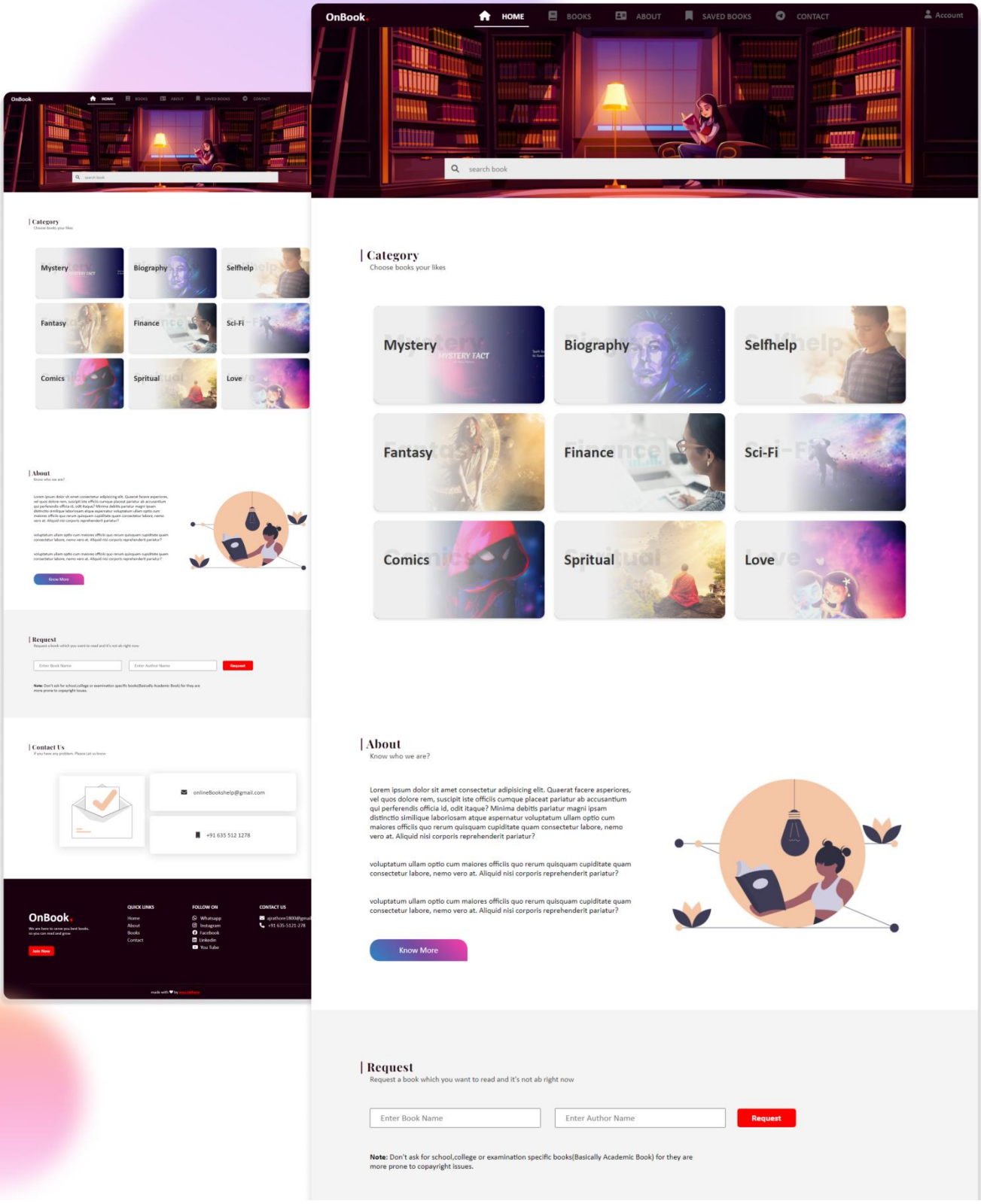
Reader/User



4.4 | Form & Report Design
| Layout Design

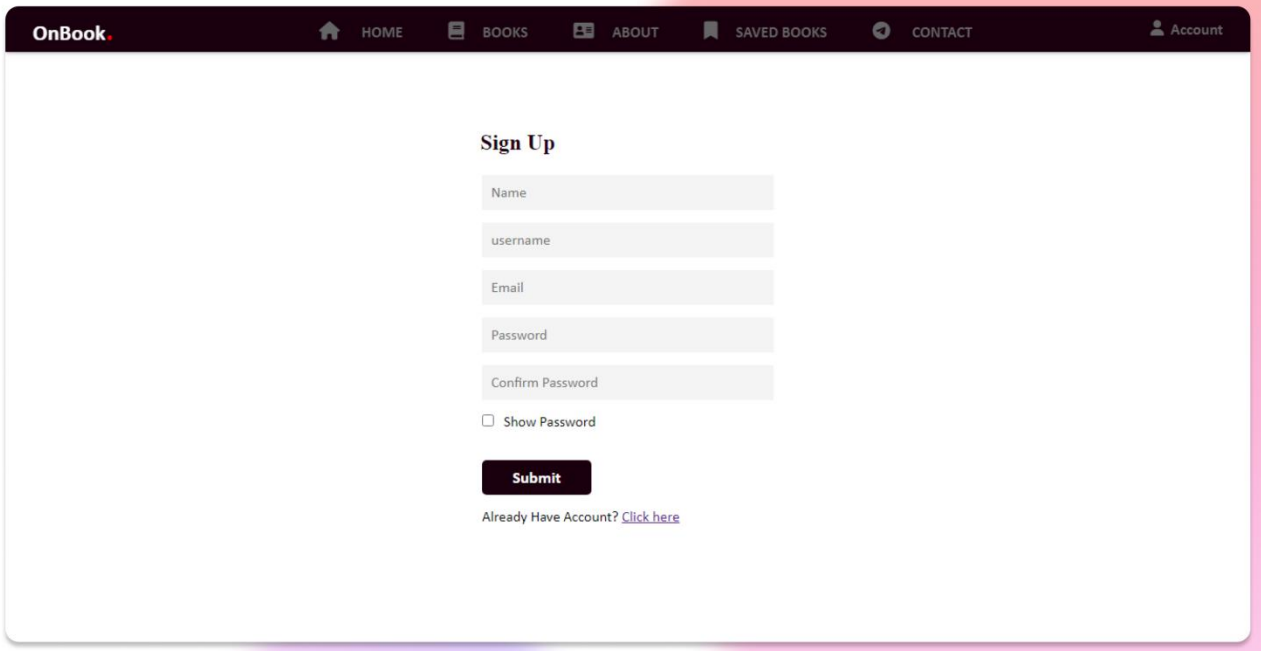
Home Page Design

Any one can view home page of website, and view books, Search and filter but after login Reader get some Additional features like request for a book add to save and maintain profile change password etc..



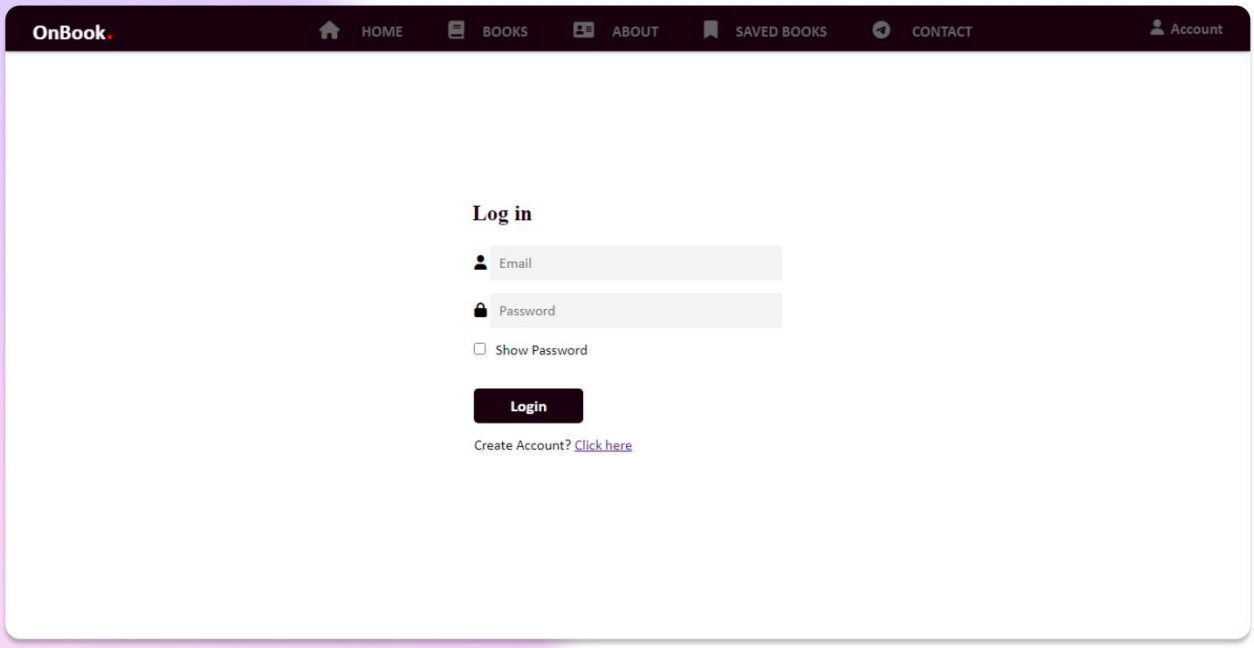
Form Design

Signup Form used by Readers or users



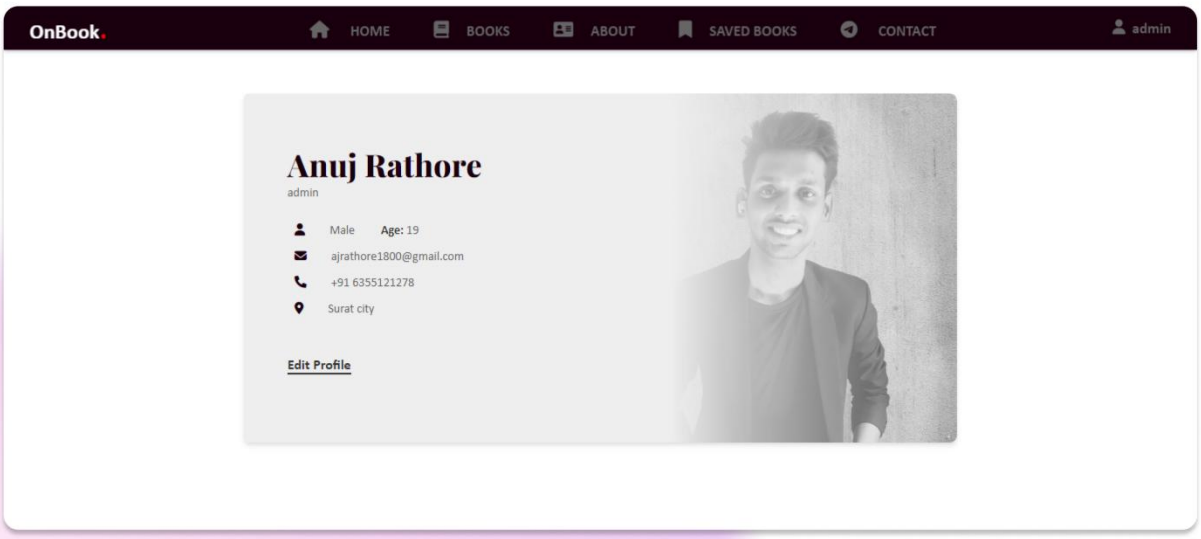
Login From For both Admin & User

If a use/reader Enter their correct details this login page redirect to the home page. if an Admin login he also redirect to the home page but he has some additional featureus



Profile Manage Pages

User can view/update their profile and Admin and User will see some diffrent features



Before Login

Account

Sign Up

Sign In

Admin

Hello, **admin**

Admin Pannel

Profile

Change Password

Saved & Requests

New Request

Log Out

Readers

Hello, **test1**

Profile

Change Password

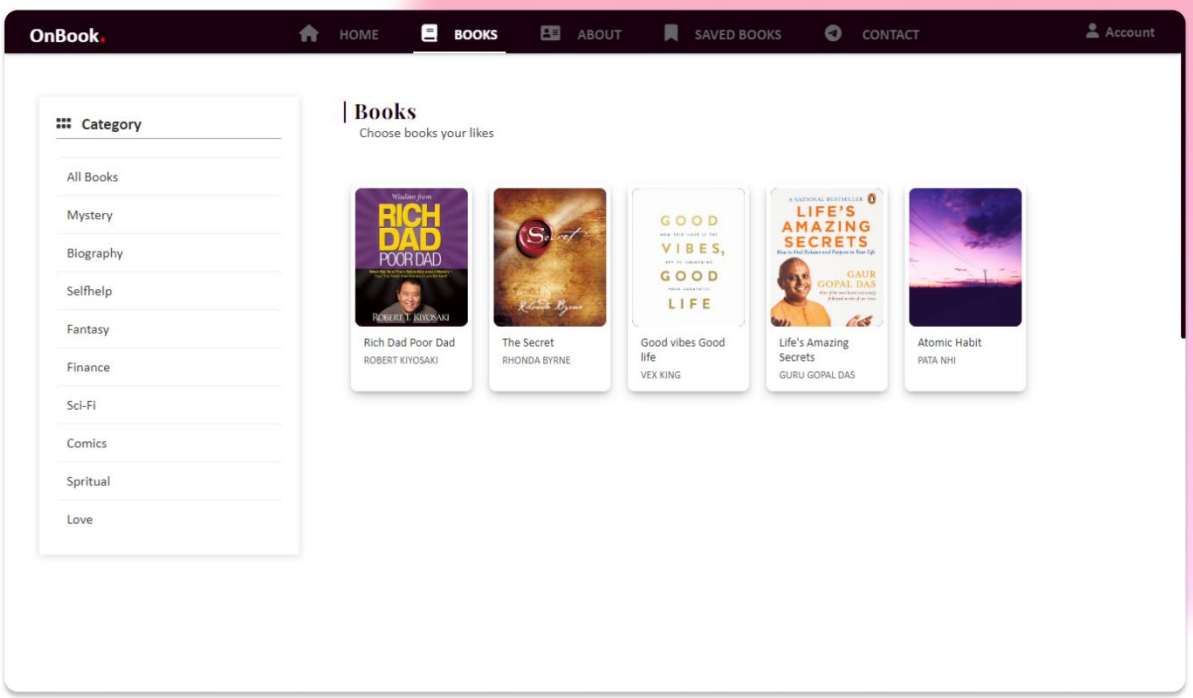
Saved & Requests

New Request

Log Out

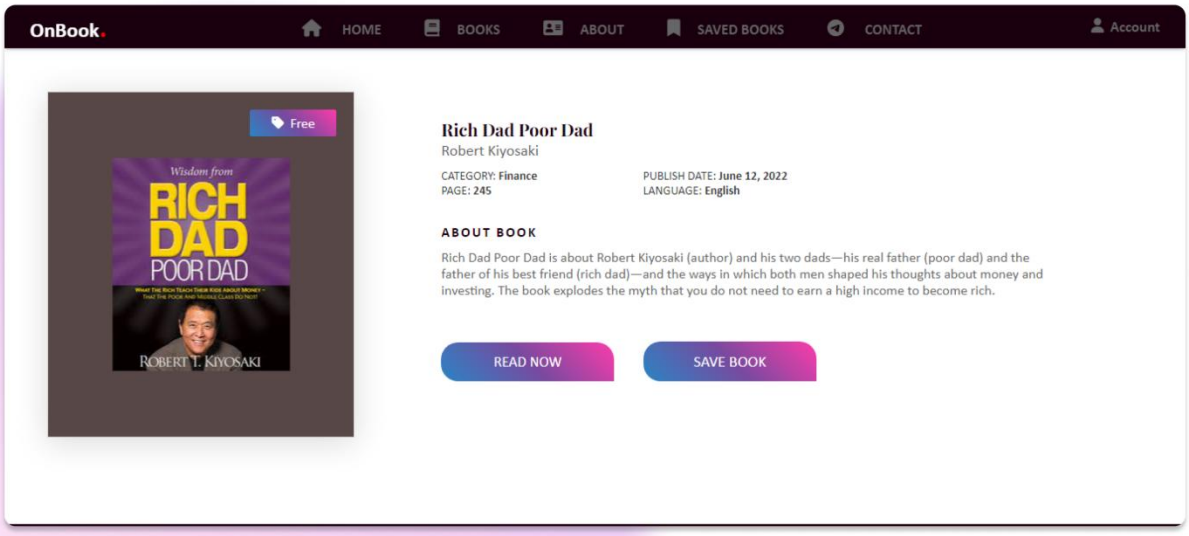
Book Page Design

This is book page of website here user can view books and filter by category and search for a book



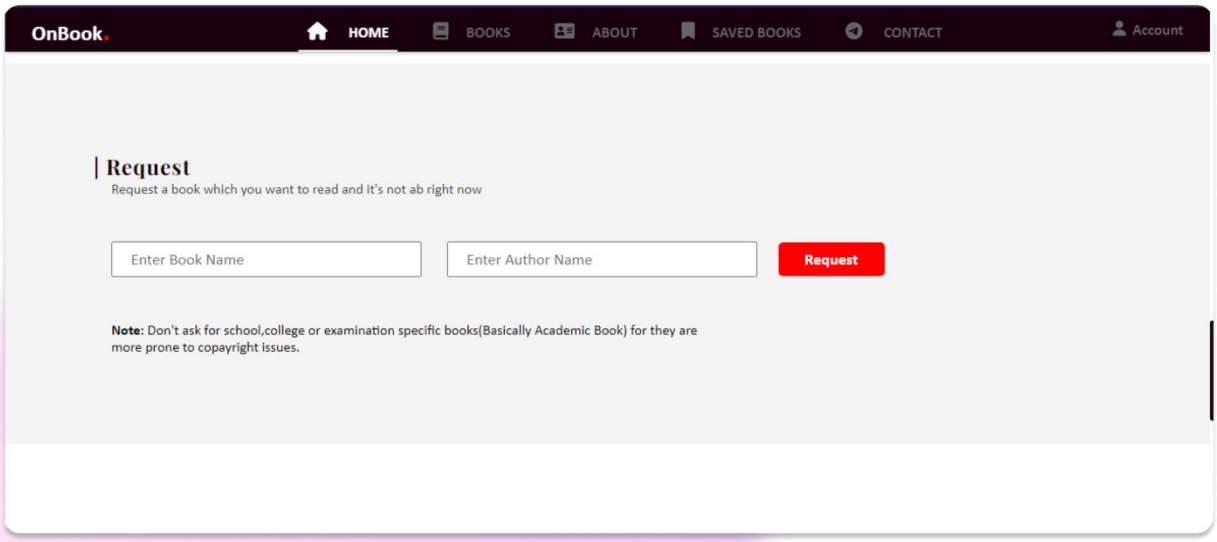
Book Details Page

After click any book user will redirect to the book Details page and user can read book form this page and save for read later but after login



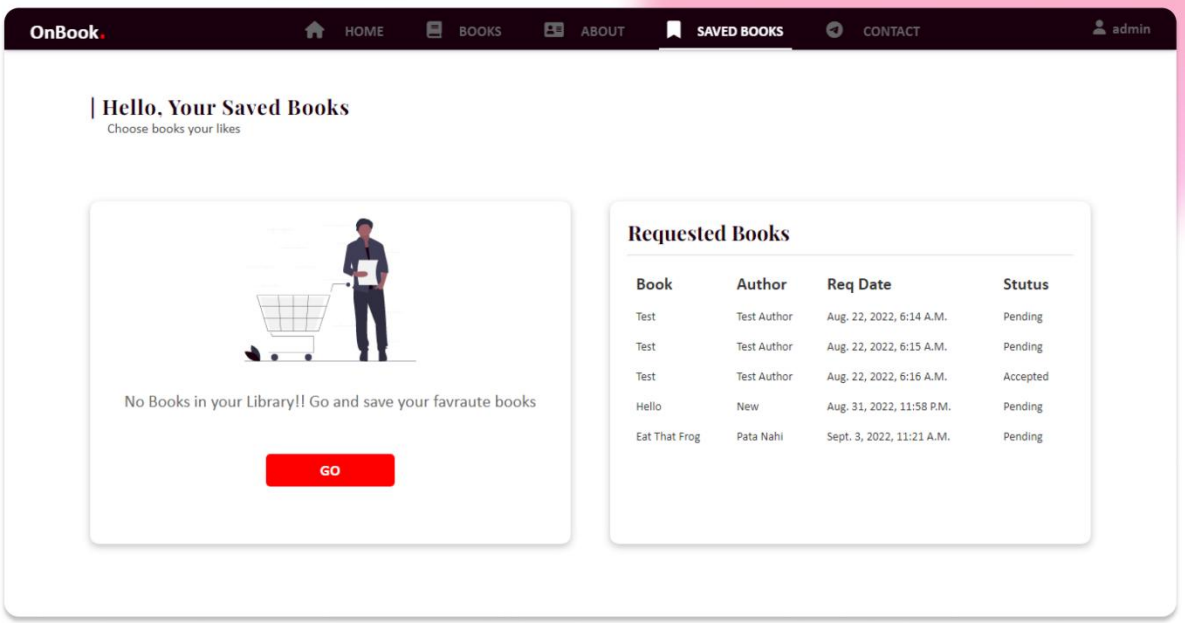
Book Request Form

An Authenticated User can make a request for a book which is note currently on website for read.



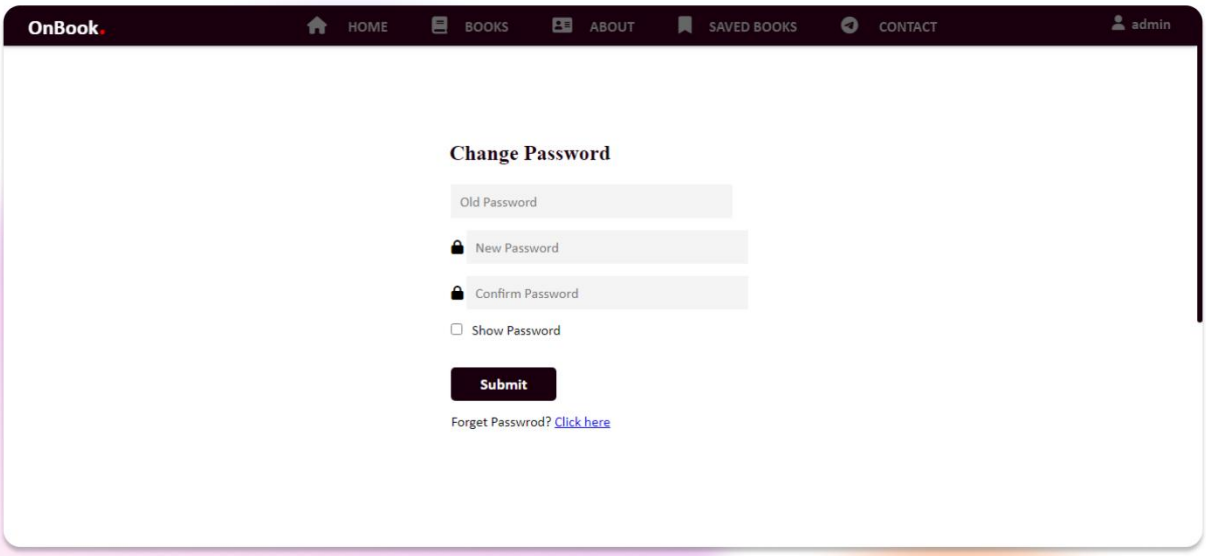
Saved book & Requested Book list

on this page user can see their saved books and also he can track their requested book status



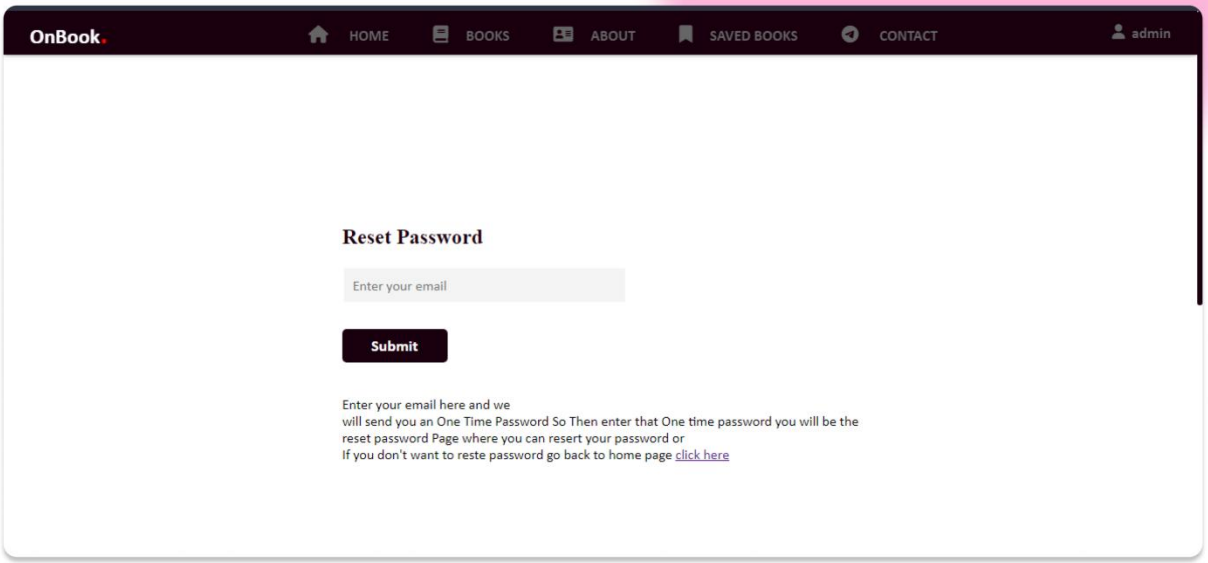
Change Password

Here An Authenticated User can change their password



Reset Password

Here a User can reset their password if they are forget it.



5 | Coding

5.1 | Code snippets

1| settings.py | file djang's config file

```
import os
from pathlib import Path
from django.contrib.messages import constants as messages
# Build paths inside the project like this: BASE_DIR / 'subdir'.
BASE_DIR = Path(__file__).resolve().parent.parent
# Quick-start development settings - unsuitable for production
# SECURITY WARNING: keep the secret key used in production secret!
      SECRET_KEY = 'django-insecure-8_s9330&e42+p2do^26j9hnh-l^i@lx9xj66)lk()1#b2j252s'
```

```
# SECURITY WARNING: don't run with debug turned on in production!
      DEBUG = True
```

```
# 192.168.35.149
      ALLOWED_HOSTS = []
```

```
# Application definition
      INSTALLED_APPS = [
          'jazzmin',
          'home.apps.HomeConfig',
          'account.apps.AccountConfig',
          'django.contrib.admin',
          'django.contrib.auth',
          'django.contrib.contenttypes',
          'django.contrib.sessions',
          'django.contrib.messages',
          'django.contrib.staticfiles',
      ]
```

```
      MIDDLEWARE = [
          'django.middleware.security.SecurityMiddleware',
          'django.contrib.sessions.middleware.SessionMiddleware',
          'django.middleware.common.CommonMiddleware',
          'django.middleware.csrf.CsrfViewMiddleware',
          'django.contrib.auth.middleware.AuthenticationMiddleware',
          'django.contrib.messages.middleware.MessageMiddleware',
          'django.middleware.clickjacking.XFrameOptionsMiddleware',
      ]
```

```
ROOT_URLCONF = 'OnBooks.urls'
```

```
TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [os.path.join(BASE_DIR,'template')],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]
```

```
WSGI_APPLICATION = 'OnBooks.wsgi.application'
```

```
# Database
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': BASE_DIR / 'db.sqlite3',
    }
}
```

```
AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]
```

```
# Internationalization
LANGUAGE_CODE = 'en-us'
TIME_ZONE = 'UTC'
USE_I18N = True
USE_TZ = True
```

```
# Static files (CSS, JavaScript, Images)
STATIC_URL = 'static/'
# Added by Developer
STATICFILES_DIRS = [
    os.path.join(BASE_DIR,'static')
]
```

```
# Default primary key field type
DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'
```

```
# Added manaually for image and pdf upladod
MEDIA_URL ='/media/'
MEDIA_ROOT = os.path.join(BASE_DIR,'media')
# Added manually for admin css
JAZZMIN_SETTINGS = {
    "copyright": "made with Love by Anuj Rathore",
    "welcome_sign": "OnBooks. Admin",
    "site_header": "OnBooks.",
    "site_brand": "OnBooks.",
    "user_avatar": None,
}
```

1| Project urls.py

```
from django.conf import settings
from django.contrib import admin
from django.urls import path, include
from django.conf import settings
from django.conf.urls.static import static

urlpatterns = [
    path("",include("home.urls")),
    path('admin/', admin.site.urls),
    path("account/",include('account.urls')),
]

if settings.DEBUG:
    urlpatterns += static(settings.MEDIA_URL,document_root = settings.MEDIA_ROOT)
```

2| Home App urls.py

```
from django.contrib import admin
from django.urls import path
from home import views

urlpatterns = [
    path('',views.index, name="home"),
    path('index/',views.index, name="home"),
    path('books/',views.books, name="books"),
    path('about/',views.about, name="about"),
    path('contact/',views.contact, name="contact"),
    path('savedbook/',views.savedbook, name="savedbook"),
    path('profile/',views.profile, name="profile"),
    path('request/',views.request, name="request"),
    path('bookdetails/<str:pk_test>/', views.bookdetails, name="bookdetails"),
    path('books/bookdetails/<str:pk_test>/', views.bookdetails, name="bookdetails"),
    path('remove_saved_book/<str:sbid>/', views.remove_saved_book),
    path('profile_edit/<str:uid>/', views.profile_edit, name="profile_edit"),
]
```

3| Account App urls.py

```
from django.urls import path
from account import views

urlpatterns = [
    path('signup/',views.signup, name="signup"),
    path('signin/',views.signin, name="signin"),
    path('logout/',views.logout, name="logout"),
    path('changepassword/',views.changepassword, name="changepassword"),
    path('resetpassword/',views.resetpassword, name="resetpassword"),
]
```

3| Account App views.py Code

```
import imp
from queue import Empty
from django.shortcuts import redirect, render
from django.contrib.auth.models import User ,auth
from django.contrib import messages
from django.contrib.auth.decorators import login_required
from home.models import Reader


# Signup View

def signup(request):
    if request.method == 'POST':
        name = request.POST.get('name')
        username = request.POST.get('username')
        email = request.POST.get('email')
        userpass = request.POST.get('userpass')
        cpass= request.POST.get('cpass')

        if User.objects.filter(username = username).exists():
            messages.error(request,'This Username already exist, Try with another!!')

        elif User.objects.filter(email = email).exists():
            messages.error(request,'This Email already exist, Try with another!!')
        else:
            if userpass != cpass:
                messages.error(request,'Password Miss Match!!')
            elif name =='':
                messages.error(request,'Please Enter your name !!')
            elif username =='':
                messages.error(request,'Please Enter your Username !!')
            elif email =='':
                messages.error(request,'Please Enter your email ')
```

```
            elif userpass =='':
                messages.error(request,'Please Enter Password !')
            else:
                user = User.objects.create_user(
                      username = username,
                      email=email,
                      password = userpass,
                      first_name = name
                      )
                user.save()
                reader = Reader.objects.create(user = user, name = name)
                reader.save()

                User.username = username
                User.username = User.username.capitalize()
                messages.success(request,'hello, '+  User.username+'Your Account created')
                return render(request,'signin.html')


    return render(request,'signup.html')
```

```
# login view

def signin(request):
    if request.method =='POST':
        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)
            User.username = username
            messages.success(request,User.username + '  Login Successfully !!')
            return redirect('/')
        else:
            messages.error(request,'Worng Username or Password')

    return render(request,'signin.html')
```

```
# logout view
def logout(request):
    auth.logout(request)
    messages.success(request,'Logged Out Successfullyy !!')
     return redirect('/')
```

```
# Change password view
def changepassword(request):
    if request.user.is_authenticated:
        if request.method =='POST':
            oldpass = request.POST.get('oldpass')
            newpass = request.POST.get('newpass')
            cnewpass = request.POST.get('cnewpass')

            user = request.user
            userid = User.objects.get(username = user).id

            if user.password != oldpass:
                messages.error(request,'Please Enter right old password!')
            else:
                if newpass != cnewpass:
                    messages.error(request,'Confirm Password did not match')
                else:
                    updateuser = User(
                        id = userid,
                        username = user.username,
                        email = user.email,
                        password = newpass,
                    )
                    updateuser.save()
                    messages.success(request,'Password Changed Successfully!')
                    return redirect('/profile/')


        return render(request,'changepassword.html' )
    else:
        return redirect('/account/signin/')
```

3| Home App Views.py Code

```
from asyncio.windows_events import NULL
from asyncore import read
from email import message
import imp
from multiprocessing import context
from tkinter.messagebox import RETRY
from django.contrib import messages
from django.shortcuts import redirect, render
from home.models import *
from django.core.files.storage import FileSystemStorage
from django.contrib.auth.decorators import login_required
from django.db.models import Q
```

Home page Rendering and function view

```
# index page functions
def index(request):
    if request.method =='POST':
        if request.user.is_authenticated:
            user = request.user
            book = request.POST.get('reqbook')
            author = request.POST.get('reqauthor')
            bookreq = Book_request.objects.create(user = user,book_name = book ,author_name = author)
            bookreq.save()
            messages.success(request,"Request sent successfully, Go to Saved Book to see your request Status")
        else:
            messages.error(request,"To make a Book request login first ")

    cat = Category.objects.all()
    bookslist = Book_details.objects.all()
    return render(request,'index.html',{'category':cat,'bookslist':bookslist})
```

Books page rendering & filter books by category and Searching book code

```
# fetching  book from Book_details table and Category from Category table
def books(request):
    books = Book_details.objects.all()
    if 'q' in request.GET:
        q = request.GET['q']
        squery = Q(Q(book_title__icontains = q) | Q(book_author__icontains = q))
        books = Book_details.objects.filter(squery)

    cat = Category.objects.all()
    CATID = request.GET.get('categories')
    if CATID:
        catName = Category.objects.get(id = CATID).category_name
        books = Book_details.objects.filter(category = CATID)
        Return render(request,'books.html',{'books':books,'Category':cat,'catName':catName})


     return render(request,'books.html',{'books':books,'Category':cat})
```

About page rendering view

```
# simple render about.html
def about(request):
    return render(request,'about.html')
```

Contact Page rendering view

```
# simple render contact.html
def contact(request):
    return render(request,'contact.html')
```

Saved book rendering , view request status & saved book fetching

```
# show data of saved books
def savedbook(request):
    if request.user.is_authenticated:

        if request.method =='POST':
            user = request.user
            book_id = request.POST.get('book_id')
            book = Book_details.objects.get(id = book_id)
            Saved_book(user = user,books = book).save()
            messages.success(request,'Book Saved Successfully !!!!')
            return redirect('/savedbook/')

        user = request.user
        savedbook = Saved_book.objects.filter(user = user)
        reqbook = Book_request.objects.filter(user = user)
        context ={
            'SB':savedbook,
            'reqbook':reqbook,
        }

        # if savedbook.count() == 0 :
        #     return render(request,'emptysaved.html')
        # else:
        return render(request,'savedbook.html', context)

    else :
        return redirect('/account/signin')
```

Remove book from saved books

```
# remove form saved books
def remove_saved_book(request,sbid):
    DELID = sbid
    delBook = Saved_book.objects.filter(id = DELID)
    delBook.delete()
    return redirect('/savedbook/')
```

Profile Page Rendering view , fetching readers/users data from database

```
# fetching data user profile
def profile(request):
    if request.user.is_authenticated:
        user = request.user
        reader = Reader.objects.filter(user = user)
        return render(request,'profile.html',{ 'rd':reader })
    else:
        return redirect('/account/signin')
```

Book Details page rendering and fetching all details of whose id passed in url

```
# show Book details
is_book_saved = False
def bookdetails(request, pk_test):
    book = Book_details.objects.get(id = pk_test)
    if request.user.is_authenticated:
        global is_book_saved
        is_book_saved = Saved_book.objects.filter(Q(user = request.user) & Q(books = book.id))
    context = { 'book':book,'is_book_saved':is_book_saved}
    return render(request,'bookdetails.html', context)
```

Rendering Request page

```
    # render request.html wich contains a form to make a request
    def request(request):
        return render(request,'request.html')
```

Profile Edit view or function

```
    # edit profile or uplaod profile pic
    def profile_edit(request,uid):

        if request.method =='POST':
            if request.FILES.get('pro_pic'):
                pro_pic = request.FILES.get('pro_pic')
                fs = FileSystemStorage()
                imgUrl = fs.save( 'userImg/'+pro_pic.name,pro_pic)

                user = request.user
                userid = User.objects.get(username = user).id
                print(userid)
                reader = Reader(
                    id = uid,
                    user_id = userid,
                    gender = request.POST.get('gender'),
                    name = request.POST.get('name'),
                    city = request.POST.get('city'),
                    age = request.POST.get('age'),
                    phone = request.POST.get('phone'),
                    user_img = imgUrl
                )

                if request.user.is_superuser:
                    updateuser = User(
                        id = userid,
                        username = user.username,
                        email = request.POST.get('email'),
                        password = user.password,
                        is_superuser = True,
                        is_staff = True
                    )
                else:
                    updateuser = User(
                        id = userid,
                        username = user.username,
                        email = request.POST.get('email'),
                        password = user.password,
                        is_superuser = False,
                        is_staff = False,
                    )
```

```
            reader.save()
            updateuser.save()

            messages.success(request,'update sucessfully')
            return redirect('/profile/')

        else:
            imgUrl = request.POST.get('imgurl')
            url = imgUrl[7:]
            user = request.user
            userid = User.objects.get(username = user).id
            print(userid)
            reader = Reader(
                id = uid,
                user_id = userid,
                gender = request.POST.get('gender'),
                name = request.POST.get('name'),
                city = request.POST.get('city'),
                age = request.POST.get('age'),
                phone = request.POST.get('phone'),
                user_img = url
            )

            if request.user.is_superuser:
                updateuser = User(
                    id = userid,
                    username = user.username,
                    email = request.POST.get('email'),
                    password = user.password,
                    is_superuser = True,
                    is_staff = True
                )
            else:
                updateuser = User(
                    id = userid,
                    username = user.username,
                    email = request.POST.get('email'),
                    password = user.password,
                    is_superuser = False,
                    is_staff = False,
                )

            reader.save()
            updateuser.save()

            messages.success(request,'update sucessfully')
            return redirect('/profile/')

    updateID = Reader.objects.get(id = uid)
    return render(request,'profile_edit.html',{'uid':updateID})
```

3| Home App models.py Code

```
from distutils.command.upload import upload
from operator import mod, truediv
from pyexpat import model
from sre_constants import CATEGORY
from unicodedata import category
from django.db import models
from django.contrib.auth.models import User
from django.core.validators import *
```

Readers Models

```
# Create your models here.
    GENDER =(
        ('male','male'),
        ('female','female'),
    )

class Reader(models.Model):
    user = models.OneToOneField( User, on_delete=models.CASCADE)
    name = models.CharField(max_length=122)
    gender = models.CharField(max_length=7, choices= GENDER, null=True,blank=True)
    phone = models.CharField(max_length=122 ,null=True,blank=True)
    city = models.CharField(max_length=122,null = True, blank=True)
    age = models.CharField(max_length=3,null = True, blank=True)
    user_img = models.ImageField( upload_to ="userImg",default='userImg/default.png', null = True,blank=True)

    def __str__(self) :
        return str(self.id)
```

Category Model

```
    class Category(models.Model):
        category_name = models.CharField(max_length=200)
        category_img = models.ImageField(upload_to='categoryimg')

     def __str__(self):
            return str(self.category_name)
```

Book Details Models

```
class Book_details(models.Model):
    book_title = models.CharField(max_length=300)
    book_author = models.CharField(max_length=300)
    book_desc = models.TextField()
    category = models.ForeignKey(Category, on_delete=models.CASCADE)
    book_page = models.IntegerField()
    book_lng = models.CharField(max_length=60)
    book_file = models.FileField(upload_to='documents/')
    book_img = models.ImageField(upload_to='bookimg/')
    book_upload_date = models.DateField(auto_now_add=True)

    def __str__(self):
        return str(self.id)
```

Saved Book Model

```
class Saved_book(models.Model):
    user = models.ForeignKey(User,on_delete=models.CASCADE)
    books = models.ForeignKey(Book_details, on_delete=models.CASCADE)
```

Request Book Model

```
REQ_STATUS =(
    ('pending','pending'),
    ('accepted','accepted'),
    ('rejected','rejected'),
    ('success','success'),
)

class Book_request(models.Model):
    user = models.ForeignKey(User,on_delete=models.CASCADE,default=2)
    book_name = models.CharField(max_length=200)
    author_name = models.CharField(max_length=200)
    req_date = models.DateTimeField(auto_now_add=True)
    req_status = models.CharField(max_length=200,default='pending' ,choices = REQ_STATUS)
```

6 | Test

SignUp Testing

Input	Expected	Actual Output	Result
Blank Any Filed	Error	Error	Please Enter Empty filed
password != cpassword	Error	Error	confirm Password not matches
Email already exist	Error	Error	Email already existplease try with another
username already exist	Error	Error	Username already taken, try with another username
valid inputs	Success	Success	Hello, <User_name> you signup successfully !

Login Testing

Input	Expected	Actual Output	Result
Empty Email	Error	Error	Please Enter Email
Empty Password	Error	Error	Please Enter Email
Wrong Password or Password	Error	Error	Wrong Email or Password
Valid Input	Success	Success	Logined Successfully!

Send Book Request

Input	Expected	Actual Output	Result
Make a Request without login	Error	Error	To make a request login first
Empty Book Name	Error	Error	Please Enter Book name
Empty Author Name	Error	Error	Please Enter Author name
valid inputs	Success	Success	Your Request sent successfully !Go to saved book to track your request

7 | New Tools/ Technologies Learned/Used

1| Figma

- Powerful design systems Increase design consistency with searchable assets and shareable styles in one home—centralized and accessible to your entire company.

- One of the best Tool to for UX/UI design. I also used it form editing Images for website

2| Django

- Django is a framework to development of website, This is latest technology of making website with powerful back-end code

- Django is a high-level Python web framework that enables rapid development of secure and maintainable websites. Built by experienced developers, Django takes care of much of the hassle of web development, so you can focus on writing your app without needing to reinvent the wheel.

3| JavaScript

- JavaScript, often abbreviated to JS, is a programming language that is one of the core technologies of the World Wide Web, alongside HTML and CSS. As of 2022, 98% of websites use JavaScript on the client side for webpage behavior, often incorporating third-party libraries

8 | System Limitations/ Restrictions and Dependencies/Constraints

- Admin will not able to upload the book along with the book request. He needs to upload a book to website and manually change the status of book request

- Reader can not track their reading progress and can not place book mark in pdf

- Reader can re-arrange the book sequence book in saved book page

9 | Future Enhancement & Opportunities

- Admin will able to upload the book along with the book request. And when the book uploaded Status will automatically updated to the done.

- Reader can track their reading status and also he can place bookmark so that when every user/ reader come after some time he will able to continue reading book where he left.

- Reader will able to re-arrange and he can save book according to his reading sequence .

- We can add another page for short quotes from books so user can read small quotes and if user like it so clicking on that quotes he can able to read full book.

10 | References & Bibliography

| Websites

- | docs.djangoproject.com
- | www.stackoverflow.com
- | www.geeksforgeeks
- | www.w3schools.com

| Youtube Videos

- | codeWithHarray
- | Telusko
- | Dennis lvy
- | Traversy Media