KAVIYATRI BAHINABAI CHAUDHARI, NORTH MAHARASHTRA UNIVERSITY, JALGAON

NES's

GANGAMAI COLLEGE OF ENGINEERING

ISO 9001:2008

COMPUTER ENGINEERING DEPARTMENT

Laboratory Manuals

Class: B.E. Computer **Semester: VIII**

Subject: Advanced Technology II Laboratory.

Academic Year:2023-24

Institute Vision

Empowering first generation engineers to excel in technical education based on human values

Institute Mission

- To impart affordable and quality education in order to meet needs of industry and to achieve excellence in teaching learning process
- To achieve excellence in application-oriented research in selected area of Technology to contribute to the development of the region
- To collaborate with industries to promote innovation capabilities of budding engineers
- To develop responsible citizens to awareness and acceptance of ethical values
- To build a support system of all stakeholders to develop the institute

DEPARTMENT OF COMPUTER ENGINEERING

Department Vision

Enriching computer students through quality & value education, developing globally computing computer engineers.

Department Mission

- To create globally competent students having ability to design, develop and test the software's in coordination with latest technology.
- To facilitate continuous teaching, learning process and collaborate local province state,national and international education for research.
- To interact industry expertise for academic & research.
- To impart ethical and social values among students.
- To develop skills set of our graduates so that they can be competent with software industry.

A
Laboratory Manual

For

Advanced Technology II – Lab

For the Bachelor of Engineering in the Computer Engineering

BE Computer

Semester – VIII

2023-2024

Name: ……..

Roll No:……Batch:……..

PRN NO:……

Nagaon Education Society's
GANGAMAI COLLEGE OF ENGINEERING,
NAGAON,DHULE-05
Computer Department

CERTIFICATE

This is to certify that

Mr./Mrs._____

Having Roll No. __

Of VIII Semester for the course Bachelor of Computer Engineering

Of the institute **Gangamai College of Engineering, Nagaon, Dhule**,has

completed the term work satisfactorily of the subject

Engineering - Advanced Technology II Lab

for the academic year 2023 – 2024

as prescribed in the cirriculum

Date:_____ PRN No:_____

Place: Nagaon, Dhule Exam Seat No: _____

Subject Teacher **HOD** **Principal**

Seal of the Institute

PRACTICAL-COURSE OUTCOMES

COURSE OUTCOMES(CO_S)

1. Break down real world problems / application.
2. Demonstrate Full Stack development.
3. Design Full Stack based applications.
4. Decide tools for Full Stack development.
5. Develop Full Stack based applications.

Expt No.	Name of Experiment	Page No.	Starting Date	Ending Date	Remark
1.	Data Visualization using Python.				
2.	Implementation of Django stack				
3.	Create a Ruby on Rails an application (Use Technology)				

DEPARTMENT OF COMPUTER ENGINEERING

Programme Educational Objectives

PEO 1.Core Knowledge

Computer engineering graduates will have the knowledge of basic science and Engineering skills, Humanities, social science, management and conceptual and practical understanding of core computer engineering area with project development.

PEO 2.Employment

Computer engineering graduates will have the knowledge of Industry-based technical skills to succeed in entry level engineering position at various industriesas well as in academics.

PEO 3. Professional Competency

Computer engineering graduates will have the ability to communicate effectively in English, to accumulate and disseminate the knowledge and to work effectively in a team with a sense of social awareness.

GANGAMAI COLLEGE OF ENGINEERING, NAGAON.
DEPARTMENT OF COMPUTER

Name: _

Date of Performance: _ _ /_ _/20_ _

Class: B.E. Computer
Division :
Batch:
Roll No:
Subject: **Advance Technology Lab II**

Date of Completion: _ _ /_ _/20_ _
Grade:

Sign:

Experiment No. 1

Aim: Data Visualization using Python.

1. Objective: To demonstrate various python libraries such as Matplotlib , Seaborn, Pandas

2. Background:

Data visualization is the presentation of data in a pictorial or graphical format. It enables decision makers to see analytics presented visually, so they can grasp difficult concepts or identify new patterns. With interactive visualization, you can take the concept a step further by using technology to drill down into charts and graphs for more detail, interactively changing what data you see and how it's processed.

History of Data Visualization

The concept of using pictures to understand data has been around for centuries, from maps and graphs in the 17th century to the invention of the pie chart in the early 1800s. Several decades later, one of the most cited examples of statistical graphics occurred when Charles Minard mapped Napoleon's invasion of Russia. The map depicted the size of the army as well as the path of Napoleon's retreat from Moscow – and tied that information to temperature and time scales for a more in-depth understanding of the event. It's technology, however, that truly lit the fire under data visualization. Computers made it possible to process large amounts of data at lightning-fast speeds. Today, data visualization has become a rapidly evolving blend of science and art that is certain to change the corporate landscape over the next few years.

Why is data visualization important?

Because of the way the human brain processes information, using charts or graphs to visualize large amounts of complex data is easier than poring over spreadsheets or reports. Data visualization is a quick, easy way to convey concepts in a universal manner – and you can experiment with different scenarios by making slight adjustments.

Data visualization can also:

- Identify areas that need attention or improvement.
- Clarify which factors influence customer behavior.
- Help you understand which products to place where.
- Predict sales volumes.

Python offers multiple great graphing libraries that come packed with lots of different features. No matter if you want to create interactive, live or highly customized plots python has an excellent library for you.

To get a little overview here are a few popular plotting libraries: Matplotlib, Pandas and Seaborn

Outcomes:

Able to understand :

- Matplotlib for low level, provides lots of freedom to visualization
- Pandas Visualization: easy to use interface, built on Matplotlib
- Seaborn: high-level interface, great default styles

Example:

Data Visualization dataset: San Francisco Salaries

```
salaries = pd.read_csv('./Salaries.csv')

salaries.info()

for col in ['BasePay', 'OvertimePay', 'OtherPay', 'Benefits']:

  salaries[col]=pd.to_numeric(salaries[col],errors='coerce')

pay_columns=salaries.columns[3:salaries.columns.get_loc('Year')]

pay_columns
```

```
pays_arrangement = list(zip(*(iter(pay_columns),) * 3))
fig, axes = plt.subplots(2,3)


for i in range(len(pays_arrangement)):
  for j in range(len(pays_arrangement[i])):
# pass in axes to pandas histsalaries[pays_arrangement[i][j]].hist(ax=axes[i,j])
# axis objects have a lot of methods for customizing the look of aplot
axes[i,j].set_title(pays_arrangement[i][j])plt.show()
fig, axes = plt.subplots(2,3)
# set the figure height
fig.set_figheight(5)fig.set_figwidth(12)
for i in range(len(pays_arrangement)):
  for j in range(len(pays_arrangement[i])):#passinaxestopandashist
        salaries[pays_arrangement[i][j]].hist(ax=axes[i,j])
        axes[i,j].set_title(pays_arrangement[i][j])
# add a row of emptiness between the two rows
plt.subplots_adjust(hspace=1)
# add a row of emptiness between the cols
plt.subplots_adjust(wspace=1)
plt.show()
# and here is a cleaner version using tick rotation and plot spacing
fig, axes = plt.subplots(2,3)
# set the figure heightfig.set_figheight(5)
fig.set_figwidth(12)
foriinrange(len(pays_arrangement)):
```

```
for j in range(len(pays_arrangement[i])):
        salaries[pays_arrangement[i][j]].hist(ax=axes[i,j])
        axes[i,j].set_title(pays_arrangement[i][j])
        # set xticks with these labels,
        axes[i,j].set_xticklabels(labels=axes[i,j].get_xticks(),

            #withthisrotation
            rotation=30)
plt.subplots_adjust(hspace=1)
plt.subplots_adjust(wspace=1)
plt.show()
```

Questions and Answers:

Questions: 1. What type of Variables uses in Python?
Questions: 1. Explain Datatype uses in Python?
Questions: 1. How Database Connectivity done in Python?

Handwritten Answer on Next Blank Page of same margin

GANGAMAI COLLEGE OF ENGINEERING, NAGAON.
DEPARTMENT OF COMPUTER

Name: _

Date of Performance: _ _ /_ _/20_ _

Class: B.E. Computer
Division :
Batch:
Roll No:
Subject: **Advance Technology Lab II**

Date of Completion: _ _ /_ _/20_ _
Grade:

Sign:

Experiment No. 2

Aim: Implementation of Django stack.

1. Objective: To demonstrate Django stack.

2. Implementation of Django Stack:

Django is available open-source under the BSD license. We recommend using the latest version of Python 3. The last version to support Python 2.7 is Django 1.11 LTS. See the FAQ for the Python versions supported by each version of Django. Here's how to get it:

Option 1: Get the latest official version

The latest official version is 3.2.5 (LTS). Read the 3.2.5 release notes, then install it with pip:

```
pip install Django=3.2.5
```

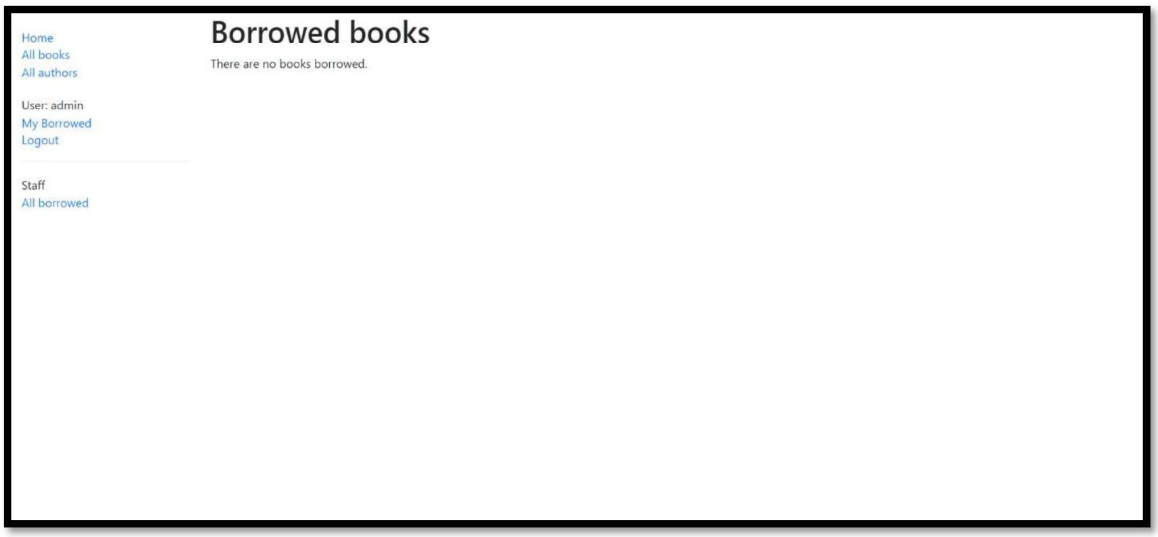
Option 2: Get the latest development version

The latest and greatest Django version is the one that's in our Git repository (our revision-control system). This is only for experienced users who want to try incoming changes and help identify bugs before an official release. Get it using this shell command, which requires Git:

```
git clone https://github.com/django/django.git
```

In a traditional data-driven website, a web application waits for HTTP requests from the web browser (or other client). When a request is received the application works out what is needed based on the URL and possibly information in POST data or GET data. Depending on what is


Home
All books
All authors
User: admin
My Borrowed
Logout
Staff
All borrowed
Author List
There are no authors available.


Home
All books
All authors
User: admin
My Borrowed
Logout
Staff
All borrowed
Borrowed books
There are no books borrowed.


Home
All books
All authors
Login
Author List
There are no authors available.


Please login to see this page.
Home
All books
All authors
Login
Username:
Password:
login
Lost password?

OUTPUT:

Home
All books
All authors

Login

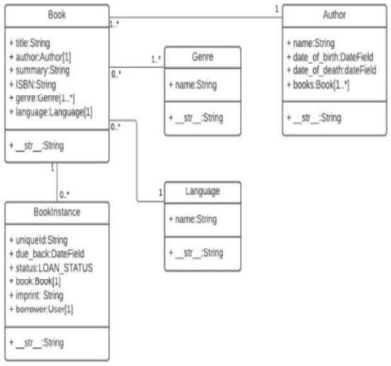
Local Library Home

Welcome to *LocalLibrary*, a very basic Django website developed as a tutorial example on the Mozilla Developer Network.

The tutorial demonstrates how to create a Django skeleton website and application, define URL mappings, views (including Generic List and Detail Views), models and templates.

UML Models

An UML diagram of the site's Django model structure is shown below.



Dynamic content

The library has the following record counts:

- **Books:** 0
- **Copies:** 0
- **Copies available:** 0
- **Authors:** 0

You have visited this page 1 time.

Home
All books
All authors

Login

Book List

There are no books in the library.

```
class AuthorUpdate(PermissionRequiredMixin, UpdateView):model =
    Author
    fields = '__all__' # Not recommended (potential security issue if more


fields added)
    permission_required = 'catalog.can_mark_returned'


class AuthorDelete(PermissionRequiredMixin, DeleteView):model =
    Author
    success_url = reverse_lazy('authors') permission_required =
    'catalog.can_mark_returned'


# Classes created for the forms challenge
class BookCreate(PermissionRequiredMixin, CreateView):model =
    Book
    fields = ['title', 'author', 'summary', 'isbn', 'genre', 'language']permission_required =
    'catalog.can_mark_returned'


class BookUpdate(PermissionRequiredMixin, UpdateView):model =
    Book
    fields = ['title', 'author', 'summary', 'isbn', 'genre', 'language']permission_required =
    'catalog.can_mark_returned'


class BookDelete(PermissionRequiredMixin, DeleteView):model =
    Book
    success_url = reverse_lazy('books') permission_required =
    'catalog.can_mark_returned'
```

```
import datetime
from django.contrib.auth.decorators import login required, permission_required

# from .forms import RenewBookForm
from catalog.forms import RenewBookForm


@login required
@permission required('catalog.can mark returned', raise exception=True) def renew_book_librarian(request, pk):
    """View function for renewing a specific BookInstance by librarian."""
    book_instance = get_object_or_404(BookInstance, pk=pk)

    # If this is a POST request then process the Form data if request.method == 'POST':

        # Create a form instance and populate it with data from the request(binding):
        form = RenewBookForm(request.POST)

        # Check if the form is valid:
        if form.is_valid():
            # process the data in form.cleaned_data as required (here wejust write it to the model due_back field)
            book_instance.due_back = form.cleaned_data['renewal_date']
            book_instance.save()

            # redirect to a new URL:
            return HttpResponseRedirect(reverse('all-borrowed'))

    # If this is a GET (or any other method) create the default formelse:
        proposed_renewal_date = datetime.date.today() + datetime.timedelta(weeks=3)
        form = RenewBookForm(initial={'renewal_date': proposed_renewal_date})

    context = {
        'form': form,
        'book_instance': book_instance,
    }

    return render(request, 'catalog/book_renew_librarian.html', context)


from django.views.generic.edit import CreateView, UpdateView, DeleteView from django.urls import reverse_lazy
from .models import Author


class AuthorCreate(PermissionRequiredMixin, CreateView):model =
    Author
    fields = ['first_name', 'last_name', 'date_of_birth', 'date_of_death'] initial = {'date_of_death': '11/06/2020'}
    permission_required = 'catalog.can_mark_returned'
```

```
from django.views import generic


class BookListView(generic.ListView):
    """Generic class-based view for a list of books."""
    model = Book
    paginate_by = 10


class BookDetailView(generic.DetailView):
    """Generic class-based detail view for a book."""
    model = Book


class AuthorListView(generic.ListView):
    """Generic class-based list view for a list of authors."""
    model = Author paginate by=
    10


class AuthorDetailView(generic.DetailView):
    """Generic class-based detail view for an author."""
    model = Author


from django.contrib.auth.mixins import LoginRequiredMixin


class LoanedBooksByUserListView(LoginRequiredMixin, generic.ListView): """Generic
    class-based view listing books on loan to current user."""model = BookInstance
    template name ='catalog/bookinstance list borrowed user.html'paginate by
    = 10

    def get queryset(self):return
BookInstance.objects.filter(borrower=self.request.user).filter(status_____exac
t='o').order_by('due_back')


# Added as part of challenge!
from django.contrib.auth.mixins import PermissionRequiredMixin


class LoanedBooksAllListView(PermissionRequiredMixin, generic.ListView):
    """Generic class-based view listing all books on loan. Only visible to users with
can_mark_returned permission."""
    model = BookInstance
    permission required = 'catalog.can mark returned' template name =
    'catalog/bookinstance_list_borrowed_all.html'paginate_by = 10

    def get queryset(self):return
BookInstance.objects.filter(status_exact='o').order_by('due_back')


from django.shortcuts import get_object_or_404 from
django.http import HttpResponseRedirect from
django.urls import reverse
```

```
urlpatterns += [
    path('book/<uuid:pk>/renew/', views.renew_book_librarian, name='renew-book-librarian'),
]


# Add URLConf to create, update, and delete authors
urlpatterns += [
    path('author/create/', views.AuthorCreate.as_view(), name='author-create'),
    path('author/<int:pk>/update/', views.AuthorUpdate.as_view(), name='author-update'),
    path('author/<int:pk>/delete/', views.AuthorDelete.as_view(),name='author-delete'),
]

# Add URLConf to create, update, and delete books
urlpatterns += [
    path('book/create/', views.BookCreate.as_view(), name='book-create'),
    path('book/<int:pk>/update/', views.BookUpdate.as_view(), name='book-update'),
    path('book/<int:pk>/delete/', views.BookDelete.as_view(), name='book-delete'),
]
```

views.py

```
from django.shortcuts import render #

Create your views here.

from .models import Book, Author, BookInstance, Genre


def index(request):
    """View function for home page of site."""
    # Generate counts of some of the main objects num_books =
    Book.objects.all().count() num_instances =
    BookInstance.objects.all().count()# Available copies of books
    num_instances_available =
BookInstance.objects.filter(status__exact='a').count()
    num_authors = Author.objects.count() # The 'all()' is implied bydefault.

    # Number of visits to this view, as counted in the session variable. num_visits =
    request.session.get('num_visits', 1) request.session['num_visits'] = num_visits+1

    # Render the HTML template index.html with the data in the contextvariable.
    return render(
        request,
        'index.html',
        context={'num_books': num_books, 'num_instances': num_instances,
                 'num_instances_available': num_instances_available,
'num_authors': num_authors,
                 'num_visits': num_visits},
    )
```

```
        permissions = (("can_mark_returned", "Set book as returned"),)

    def __str__(self):
        """String for representing the Model object."""
        return '{0} ({1})'.format(self.id, self.book.title)


class Author(models.Model):
    """Model representing an author."""
    first_name = models.CharField(max_length=100)
    last_name = models.CharField(max_length=100)
    date_of_birth = models.DateField(null=True, blank=True)
    date_of_death = models.DateField('died', null=True, blank=True)

    class Meta:
        ordering = ['last_name', 'first_name']

    def get_absolute_url(self):
        """Returns the url to access a particular author instance."""
        return reverse('author-detail', args=[str(self.id)])

    def __str__(self):
        """String for representing the Model object."""
        return '{0}, {1}'.format(self.last_name, self.first_name)
```

urls.py

```
from django.urls import pathfrom .

import views


urlpatterns = [
    path('', views.index, name='index'),
    path('books/', views.BookListView.as_view(), name='books'),
    path('book/<int:pk>', views.BookDetailView.as_view(), name='book-
detail'),
    path('authors/', views.AuthorListView.as_view(), name='authors'),
    path('author/<int:pk>',
         views.AuthorDetailView.as_view(), name='author-detail'),
]


urlpatterns += [
    path('mybooks/', views.LoanedBooksByUserListView.as_view(), name='my-
borrowed'),
    path(r'borrowed/', views.LoanedBooksAllListView.as_view(), name='all-borrowed'),
# Added for challenge
]


# Add URLConf for librarian to renew a book.
```

```
        ordering = ['title', 'author']

    def display_genre(self):
        """Creates a string for the Genre. This is required to display genre in
Admin."""
        return ', '.join([genre.name for genre in self.genre.all()[:3]])

    display_genre.short_description = 'Genre'

    def get_absolute_url(self):
        """Returns the url to access a particular book instance."""
        return reverse('book-detail', args=[str(self.id)])

    def __str__(self):
        """String for representing the Model object."""
        return self.title


import uuid # Required for unique book instances from datetime
import date

from django.contrib.auth.models import User # Required to assign User as a borrower


class BookInstance(models.Model):
    """Model representing a specific copy of a book (i.e. that can be borrowed from
the library)."""
    id = models.UUIDField(primary_key=True, default=uuid.uuid4,
                          help_text="Unique ID for this particular book
across whole library")
    book = models.ForeignKey('Book', on_delete=models.RESTRICT, null=True) imprint =
    models.CharField(max_length=200)
    due_back = models.DateField(null=True, blank=True)
    borrower = models.ForeignKey(User, on_delete=models.SET_NULL,
null=True, blank=True)

    @property
    def is_overdue(self):
        if self.due_back and date.today() > self.due_back:return True
        return False

    LOAN_STATUS = (
        ('d', 'Maintenance'),
        ('o', 'On loan'),
        ('a', 'Available'),
        ('r', 'Reserved'),
    )

    status = models.CharField(

        max_length=1,
        choices=LOAN_STATU
        S,              blank=True,
        default='d',
        help_text='Book availability')

    class Meta:
        ordering = ['due_back']
```

models.py

```
from  django.db  import  models #

Create your models here.

from django.urls import reverse # To generate URLS by reversing URLpatterns


class Genre(models.Model):
    """Model representing a book genre (e.g. Science Fiction, NonFiction)."""
    name  =  models.CharField(
        max_length=200,
        help_text="Enter a book genre (e.g. Science Fiction, French Poetry
etc.)"
        )

    def__str__(self):
        """String for representing the Model object (in Admin site etc.)"""
        return self.name

class Language(models.Model):
    """Model representing a Language (e.g. English, French, Japanese, etc.)"""
    name = models.CharField(max  length=200,
                            help  text="Enter  the  book's  natural  language  (e.g.
English, French, Japanese etc.)")

    def__str__(self):
        """String for representing the Model object (in Admin site etc.)"""
        return self.name


class Book(models.Model):
    """Model representing a book (but not a specific copy of a book)."""
    title = models.CharField(max  length=200)
    author  =  models.ForeignKey('Author',  on_delete=models.SET_NULL,
null=True)
    # Foreign Key used because book can only have one author, but authors can have
multiple books
    # Author as a string rather than object because it hasn't been declaredyet in file.
    summary = models.TextField(max  length=1000, help  text="Enter a briefdescription
of the book")
    isbn = models.CharField('ISBN', max_length=13,
                            unique=True,      help_text='13
                            Character <a
href="https://www.isbn-international.org/content/what-isbn'
                                      '">ISBN number</a>')


    genre = models.ManyToManyField(Genre, help  text="Select a genre forthis book")
    # ManyToManyField used because a genre can contain many books and aBook can
cover many genres.
    # Genre class has already been defined so we can specify the objectabove.
    language  =  models.ForeignKey('Language',  on_delete=models.SET_NULL,
null=True)


    class Meta:
```

```
@admin.register(carinfo)
class Carlist(admin.ModelAdmin):
	list_display = ['carnumber',
'carstartlocation','carsecondlocation','carthridlocation','carfourthlocatio
n','carendlocation','availableSeatsStop4', 'carstatus']

@admin.register(stationMapping)
class stationMappingDetails(admin.ModelAdmin):
	list_display =
['carnumber','runningdays','startLocation','nextLocation','availSeat','totalSeat','active']
```

app.py

```
from django.apps import AppConfig

class CatalogConfig(AppConfig):
	name = 'catalog'
```

forms.py

```
from django.core.exceptions import ValidationError from
django.utils.translation import gettext_lazy as _import datetime  #
for checking renewal date range.

from django import forms

class RenewBookForm(forms.Form):
	"""Form for a librarian to renew books."""
	renewal_date = forms.DateField(
			help_text="Enter a date between now and 4 weeks (default 3).")

	def clean_renewal_date(self):
		data = self.cleaned_data['renewal_date']

		# Check date is not in past.
		if data < datetime.date.today():
			raise ValidationError(_('Invalid date - renewal in past'))


# Check date is in range librarian allowed to change (+4 weeks)if data >
datetime.date.today() + datetime.timedelta(weeks=4):
			raise ValidationError(
				_('Invalid date - renewal more than 4 weeks ahead'))

		# Remember to always return the cleaned data.return
		data
```

Password_reset_email.html

```
Someone asked for password reset for email {{ email }}. Follow the linkbelow:
{{ protocol}}://{{ domain }}{% url 'password_reset_confirm' uidb64=uidtoken=token %}
```

Style.css

```
.sidebar-nav { margin-
        top: 20px;padding:
        0;
        list-style: none;
}
```

Manage.py

```
#!/usr/bin/env python
"""Django's command-line utility for administrative tasks."""
import os
import sys


def main():
        """Run administrative tasks."""
        os.environ.setdefault('DJANGO_SETTINGS_MODULE',
'locallibrary.settings')
        try:
                from django.core.management import execute_from_command_lineexcept
        ImportError as exc:
                raise ImportError(



                ) from exc
        execute_from_command_line(sys.argv)


if name == ' main__': main()
```

Admin.py

```
from django.contrib import admin

from.models import carinfo
from .models import UserData, stationMapping

@admin.register(UserData)
class UserDetails(admin.ModelAdmin):
        list_display = ['username','email','password']
```

```
{
                <table>
                    <tr>
                        <td>{{ form.new_password1.errors }}
                            <label for="id_new_password1">New
password:</label></td>
                        <td>{{ form.new_password1 }}</td>
                    </tr>
                    <tr>
                        <td>{{ form.new_password2.errors }}
                            <label for="id_new_password2">Confirm
password:</label></td>
                        <td>{{ form.new_password2 }}</td>
                    </tr>
                    <tr>
                        <td></td>
                        <td><input type="submit" value="Change my password"
/></td>
                    </tr>
                </table>
            </form>
        {% else %}
            <h1>Password reset failed</h1>
            <p>The password reset link was invalid, possibly because it hasalready been
used. Please request a new password reset.</p>
        {% endif %}

{% endblock %}
```

Password_reset_done.html

```
{% extends "base_generic.html" %}

{% block content %}

<p>We've emailed you instructions for setting your password. If theyhaven't arrived
in a few minutes, check your spam folder.</p>

{% endblock %}
```

Password_reset_form.html

```
{% extends "base_generic.html" %}

{% block content %}

<form action="" method="post">{% csrf_token %}
        {% if form.email.errors %}{{ form.email.errors }}{% endif %}
            <p>{{ form.email }}</p>
        <input type="submit" class='btn btn-default btn-lg' value="Resetpassword" />
</form>

{% endblock %}
```

```
<form method="post" action="{% url 'login' %}">
{% csrf_token %}
<table>
<tr>
      <td>{{ form.username.label_tag }}</td>
      <td>{{ form.username }}</td>
</tr>
<tr>
      <td>{{ form.password.label_tag }}</td>
      <td>{{ form.password }}</td>
</tr>
</table>

<input type="submit" value="login" />
<input type="hidden" name="next" value="{{ next }}" />
</form>
```

```
{# Assumes you setup the password_reset view in your URLconf #}
<p><a href="{% url 'password_reset' %}">Lost password?</a></p>

{% endblock %}
```

password_reset_complete.html

```
{% extends "base_generic.html" %}

{% block content %}

<h1>The password has been changed!</h1>
<p><a href="{% url 'login' %}">log in again?</a></p>

{% endblock %}
```

Password_reset.html

```
% extends "base_generic.html" %}

{% block content %}

      {% if validlink %}
            <p>Please enter (and confirm) your new password.</p>
            <form action="" method="post">
                  <div style="display:none">
                        <input type="hidden" value="{{ csrf_token }}"
name="csrfmiddlewaretoken">
                  </div>
```

Password_reset_done.html

Password_reset_done.html

```
                        <span class="page-current">
                            Page {{ page_obj.number }} of {{
page_obj.paginator.num_pages }}.
                        </span>
                        {% if page_obj.has_next %}
                            <a href="{{ request.path }}?page={{
page_obj.next_page_number }}">next</a>

                        {% endif %}
                    </span>
                </div>
            {% endif %}
        {% endblock %}
        </div>
    </div>

    </div>
    </body>
    </html>
```

logged_out.html

```
{% extends "base_generic.html" %}

{% block content %}
<p>Logged out!</p>

<a href="{% url 'login'%}">Click here to login again.</a>
```

Login.html

```
{% extends "base_generic.html" %}

{% block content %}

{% if form.errors %}
<p>Your username and password didn't match. Please try again.</p>
{% endif %}

{% if next %}
    {% if user.is_authenticated %}
    <p>Your account doesn't have access to this page. To proceed,please login with an
    account that has access.</p>
    {% else %}
    <p>Please login to see this page.</p>
    {% endif %}
{% endif %}
```

```
Xr2"crossorigin="anonymous">


    <!-- Add additional CSS in static file -->
    {% load static %}
    <link rel="stylesheet" href="{% static 'css/styles.css' %}">
</head>
<body>

<div class="container-fluid">

<div class="row">
    <div class="col-sm-2">
    {% block sidebar %}
    <ul class="sidebar-nav">
        <li><a href="{% url 'index' %}">Home</a></li>
        <li><a href="{% url 'books' %}">All books</a></li>
        <li><a href="{% url 'authors' %}">All authors</a></li>
    </ul>

    <ul class="sidebar-nav">
      {% if user.is_authenticated %}
        <li>User: {{ user.get_username }}</li>
        <li><a href="{% url 'my-borrowed' %}">My Borrowed</a></li>
        <li><a href="{% url 'logout'%}?next={{request.path}}">Logout</a></li>
      {% else %}
        <li><a href="{% url 'login'%}?next={{request.path}}">Login</a></li>
      {% endif %}
    </ul>

      {% if user.is_staff %}
      <hr />
      <ul class="sidebar-nav">
      <li>Staff</li>
      {% if perms.catalog.can_mark_returned %}
      <li><a href="{% url 'all-borrowed' %}">All borrowed</a></li>
      {% endif %}
      </ul>
        {% endif %}


{% endblock %}
    </div>
    <div class="col-sm-10 ">
    {% block content %}{% endblock %}

    {% block pagination %}
        {% if is_paginated %}
            <div class="pagination">
                <span class="page-links">
                    {% if page_obj.has_previous %}
                        <a href="{{ request.path }}?page={{
page_obj.previous_page_number }}">previous</a>
                    {% endif %}
```

Example:

Index.html

```
{% extends "base_generic.html" %}

{% block content %}
<h1>Local Library Home</h1>

<p>Welcome to <em>LocalLibrary</em>, a very basic Django website developed as a <a
href="https://developer.mozilla.org/en-US/docs/Learn/Server-
side/Django/Tutorial_local_library_website">tutorial example</a> on the Mozilla Developer
Network.</p>
<p>The tutorial demonstrates how to create a Django skeleton website and application, define
URL mappings, views (including Generic List and Detail Views), models and templates.</p>


<h2>UML Models</h2>
<p>An UML diagram of the site's Django model structure is shown below. </p>

<div>
{% load static %}
<img src="{% static "images/local_library_model_uml.png" %}" alt="My image"
style="width:555px;height:540px;"/>
</div>


<h2>Dynamic content</h2>

<p>The library has the following record counts:</p>
<ul>
<li><strong>Books:</strong> {{ num_books }}</li>
<li><strong>Copies:</strong> {{ num_instances }}</li>
<li><strong>Copies available:</strong> {{ num_instances_available }}</li>
<li><strong>Authors:</strong> {{ num_authors }}</li>
</ul>


<p>You have visited this page {{ num_visits }} time{{ num_visits|pluralize
}}.</p>

{% endblock %}
```

Base_generic.html

```
<!DOCTYPE html>
<html lang="en">
<head>

    {% block title %}<title>Local Library</title>{% endblock %}
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@4.5.3/dist/css/bootstrap.min.css"


TX8t27EcRE3e/ihU7zmQxVncDAy5uIKz4rEkgIXeMed4M0jlfIDPvg6uqKI2x
```

6. python3 manage.py collectstatic
7. python3 manage.py test # Run the standard tests. These should all pass.
8. python3 manage.py createsuperuser # Create a superuser
9. python3 manage.py runserver
10. Open a browser to http://127.0.0.1:8000/admin/ to open the admin site
11. Create a few test objects of each type.
12. Open tab to http://127.0.0.1:8000 to see the main site, with your new objects.

Outcomes:

Able to deploy project in Django stack.

Questions:

1. Discuss Models in detail.
2. What is the use of view?
3. State the difference between flask and Django

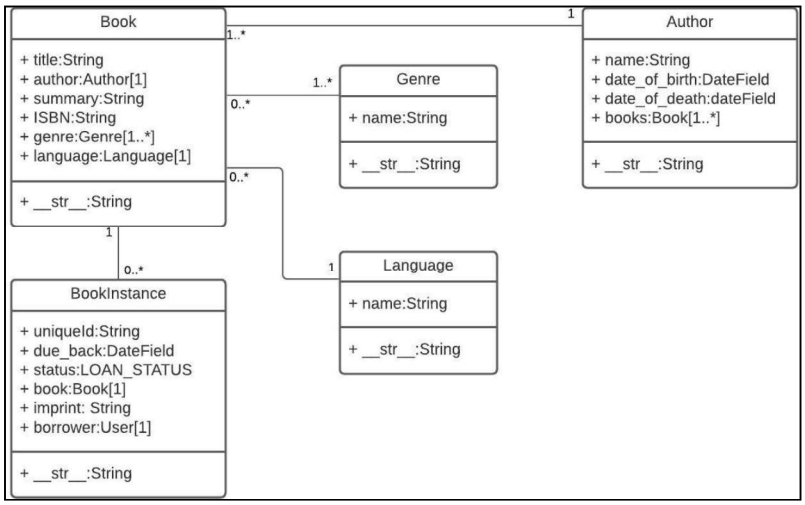
A template can be used to define the structure of any type of file; it doesn't have to be HTML!

3. Application:

This web application creates an online catalog for a small local library, where users can browse available books and manage their accounts.

The main features that have currently been implemented are:

- There are models for books, book copies, genre, language and authors.
- Users can view list and detail information for books and authors.
- Admin users can create and manage models. The admin has been optimised (the basic registration is present in admin.py, but commented out).
- Librarians can renew reserved books



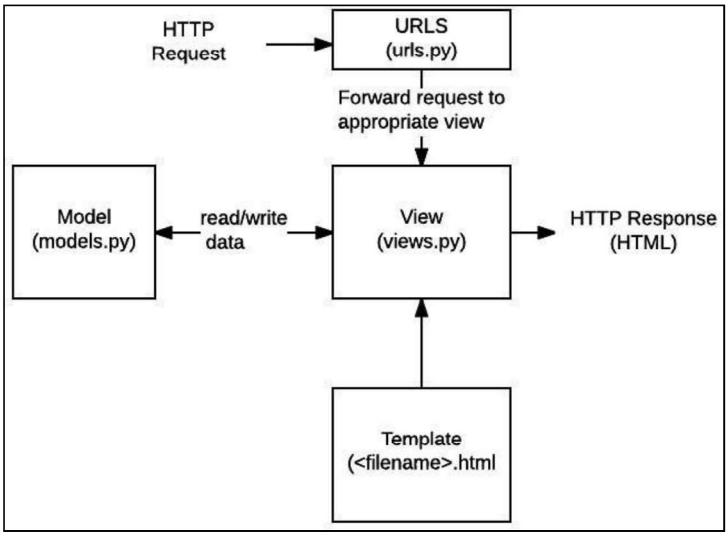
Code: https://github.com/mdn/django-locallibrary-tutorial

To get this project up and running locally on computer:

1. Set up the Python development environment. We recommend using a Python virtual environment.
2. Assuming you have Python setup, run the following commands (if you're on Windows you may use py or py -3 instead of python to start Python):
3. pip3 install -r requirements.txt
4. python3 manage.py make migrations
5. python3 manage.py migrate

required it may then read or write information from a database or perform other tasks required to satisfy the request. The application will then return a response to the web browser, often dynamically creating an HTML page for the browser to display by inserting the retrieved data into placeholders in an HTML template.

Django web applications typically group the code that handles each of these steps into separate files:



- **URLs:** While it is possible to process requests from every single URL via a single function, it is much more maintainable to write a separate view function to handle each resource. A URL mapper is used to redirect HTTP requests to the appropriate view based on the request URL. The URL mapper can also match particular patterns of strings or digits that appear in a URL and pass these to a view function as data.
- **View:** A view is a request handler function, which receives HTTP requests and returns HTTP responses. Views access the data needed to satisfy requests via *models*, and delegate the formatting of the response to *templates*.
- **Models:** Models are Python objects that define the structure of an application's data, and provide mechanisms to manage (add, modify, delete) and query records in the database.
- **Templates:** A template is a text file defining the structure or layout of a file (such as an HTML page), with placeholders used to represent actual content. A *view* can dynamically create an HTML page using an HTML template, populating it with data from a *model*

GANGAMAI COLLEGE OF ENGINEERING, NAGAON.
DEPARTMENT OF COMPUTER

Name: _

Date of Performance: _ _ /_ _/20_ _

Class: B.E. Computer
Division :
Batch:
Roll No:
Subject: **Advance Technology Lab II**

Date of Completion: _ _ /_ _/20_ _
Grade:

Sign:

Experiment No. 3

Aim: **Create a Ruby on Rails an application**

1. Objective: to develop an application by using full stack Ruby on rail.

2. Steps to develop an application:

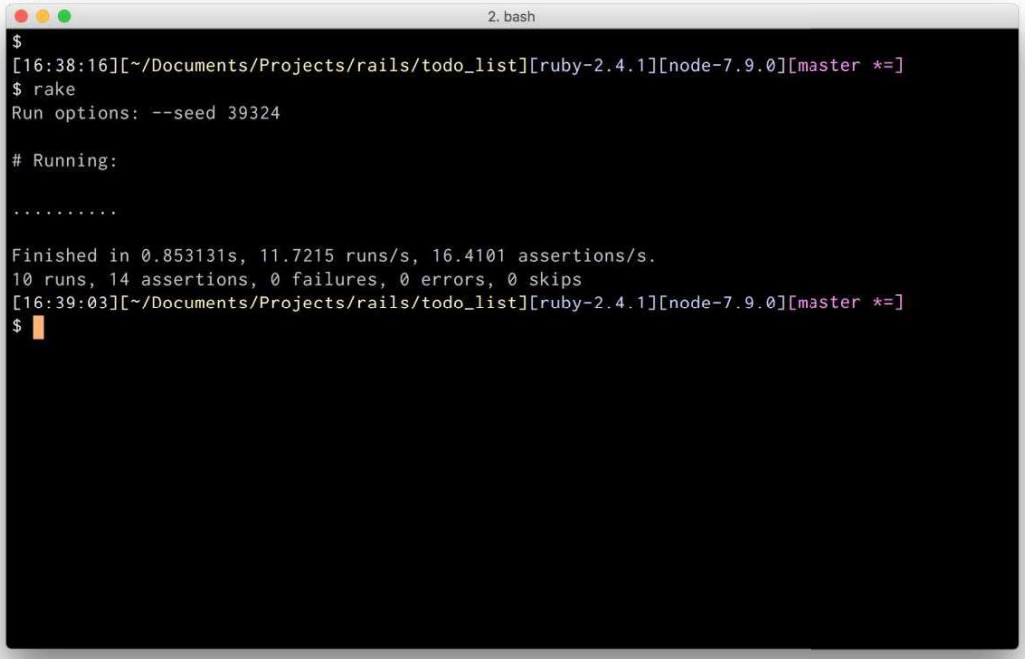
Install Ruby On Rails on Ubuntu

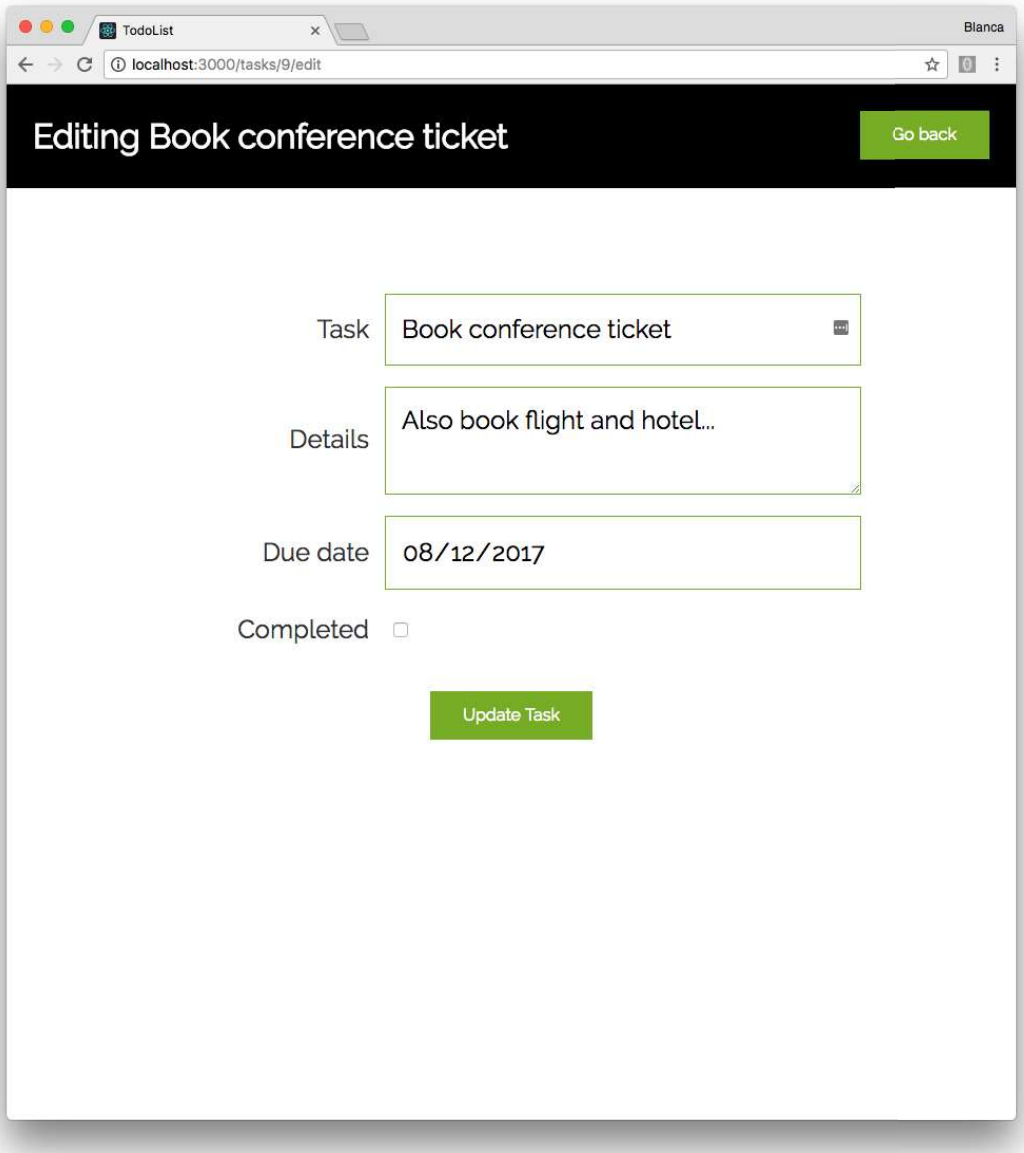
The first step is to install some dependencies for Ruby and Rails.
To make sure we have everything necessary for Webpacker support in Rails, we're first going to start by adding the Node.js and Yarn repositories to our system before installing them.

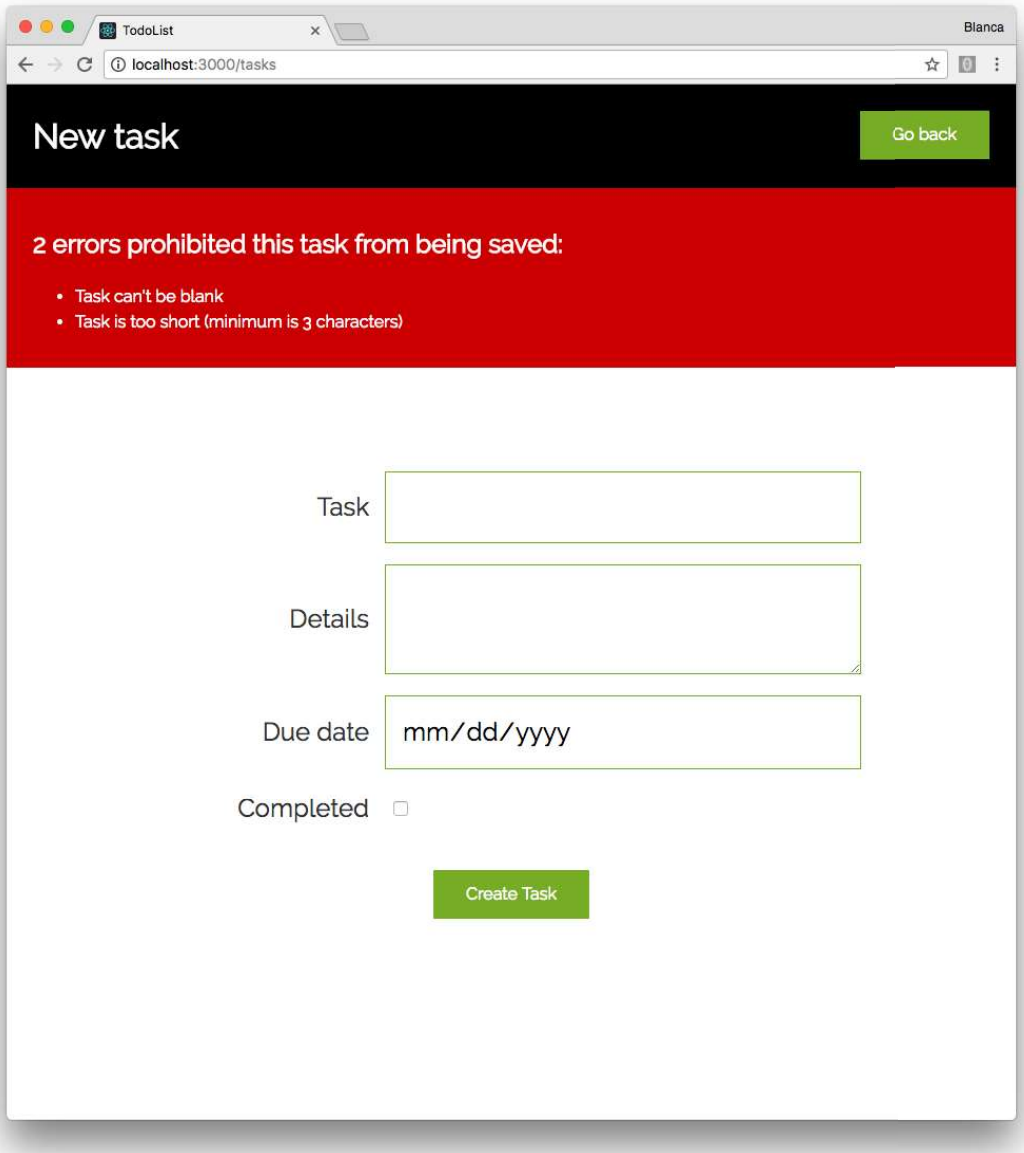
```
$sudo apt install curl
$sudo apt-get update
$sudo apt-get install git-core zlib1g-dev build-essential libssl-dev libreadline-dev libyaml-dev libsqlite3-dev sqlite3 libxml2-dev libxslt1-dev libcurl4-openssl-dev software-properties-common libffi-dev nodejs yarn
```

Installing with rbenv is a simple two step process. First you install rbenv, and then ruby- build:

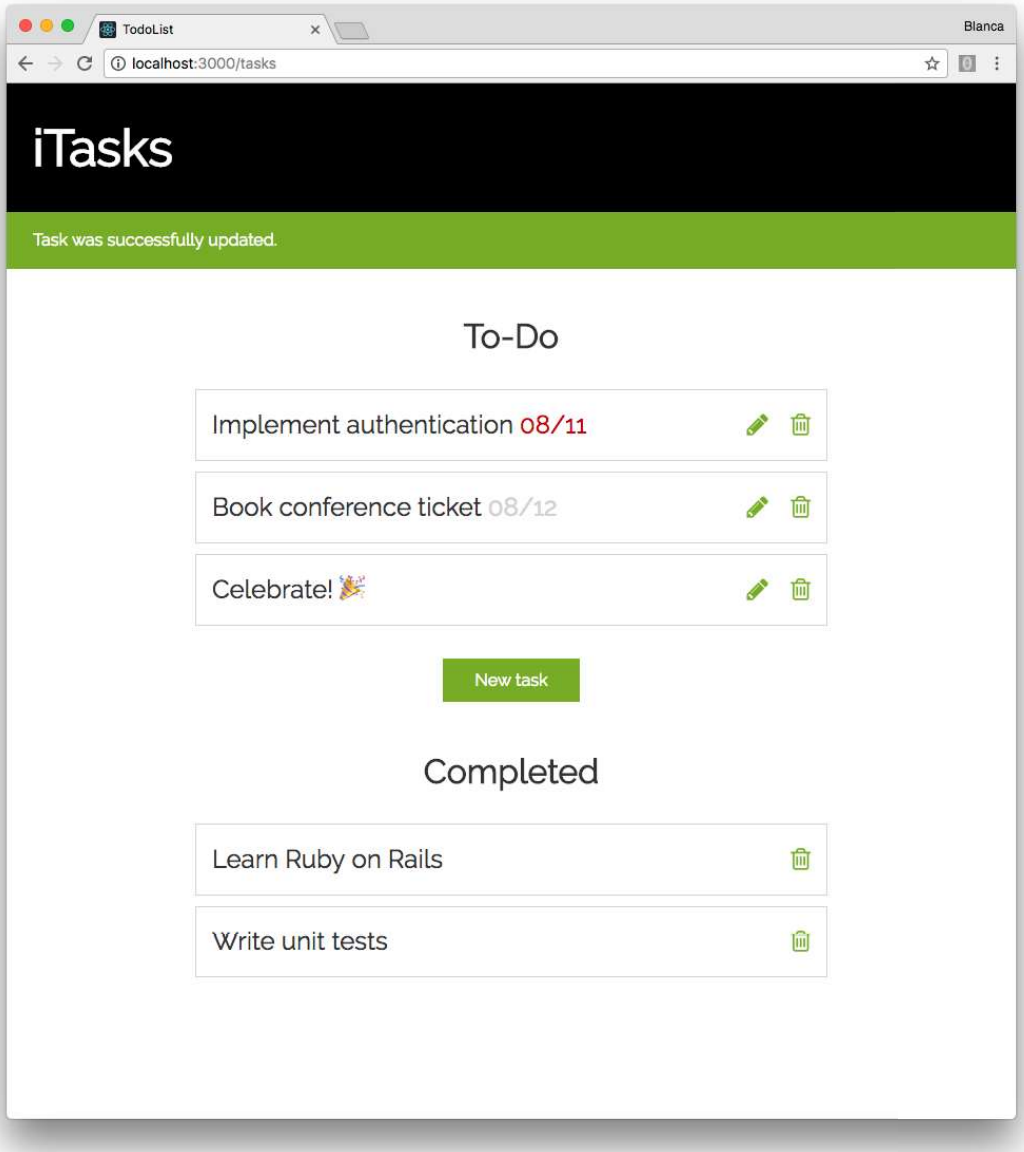
```
cd
git clone https://github.com/rbenv/rbenv.git ~/.rbenv
echo 'export PATH="$HOME/.rbenv/bin:$PATH"' >> ~/.bashrc
echo 'eval "$(rbenv init -)"' >> ~/.bashrc
exec $SHELL
```


2. bash
$
[16:38:16][~/Documents/Projects/rails/todo_list][ruby-2.4.1][node-7.9.0][master *=]
$ rake
Run options: --seed 39324
Running:
.....
Finished in 0.853131s, 11.7215 runs/s, 16.4101 assertions/s.
10 runs, 14 assertions, 0 failures, 0 errors, 0 skips
[16:39:03][~/Documents/Projects/rails/todo_list][ruby-2.4.1][node-7.9.0][master *=]
$


TodoList
Blanca
localhost:3000/tasks/9/edit
Editing Book conference ticket
Go back
Task
Book conference ticket
Details
Also book flight and hotel...
Due date
08/12/2017
Completed
Update Task


TodoList
Blanca
localhost:3000/tasks
New task
Go back
2 errors prohibited this task from being saved:
Task can't be blank
Task is too short (minimum is 3 characters)
Task
Details
Due date
mm/dd/yyyy
Completed
Create Task

OUTPUT:


TodoList
Blanca
localhost:3000/tasks
iTasks
Task was successfully updated.
To-Do
Implement authentication 08/11
Book conference ticket 08/12
Celebrate!
New task
Completed
Learn Ruby on Rails
Write unit tests

Functionality:

- As a user, I can add a task to the list.
- As a user, I can see all the tasks on the list in an overview.
- As a user, I can drill into a task to see more information about the task.
- As a user, I can delete a task.
- As a user, I can mark a task as completed.
- As a user, when I see all the tasks in the overview, if today's date is past the task'sdeadline, highlight it.

```
        <%= form.submit class: 'action-button' %>
      </div>
    <% end %>
  </div>
<% end %>
```

Application.html.erb

```
<!DOCTYPE html>
<html>
  <head>
    <title>iTasks</title>
    <%= csrf_meta_tags %>

    <%= stylesheet_link_tag    'application',
'https://fonts.googleapis.com/css?family=Raleway', media: 'all', 'data-
turbolinks-track': 'reload' %>
    <%= javascript_include_tag 'application', 'data-turbolinks-track':
'reload' %>
  </head>

  <body>
    <%= yield %>
  </body>
</html>
```

Mailer.html.erb

```
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <style>
      /* Email styles need to be inline */
    </style>
  </head>

  <body>
    <%= yield %>
  </body>
</html>
```

```
        <% end %>
      </ul>
    </div>
  <% end %>

  <div class="form-container">
    <div class="form-field">
      <div class="form-label-container">
        <%= form.label :task, class: 'form-label' %>
      </div>
      <div class="form-input-container">
        <%= form.text_field :task, id: :task_task, class: 'form-input',
disabled: readonly %>
      </div>
    </div>

    <div class="field form-field">
      <div class="form-label-container">
        <%= form.label :details, class: 'form-label' %>
      </div>
      <div class="form-input-container">
        <%= form.text_area :details, id: :task_details, class: 'form-input',
disabled: readonly %>
      </div>
    </div>

    <div class="form-field">
      <div class="form-label-container">
        <%= form.label :due_date, class: 'form-label' %>
      </div>
      <div class="form-input-container">
        <%= form.date_field :due_date, id: :task_due_date, class: 'form-
input', disabled: readonly %>
      </div>
    </div>

    <div class="field form-field">
      <div class="form-label-container">
        <%= form.label :completed, class: 'form-label' %>
      </div>
      <div class="form-input-container">
        <%= form.check_box :completed, id: :task_completed, class: 'form-
input', disabled: readonly %>
      </div>
    </div>

    <% unless readonly %>
      <div class="field form-buttons">
```

```
      <% end %>
    </ul>
  </div>
</div>
```

_tasl.html.erb

```
<%= link_to task, class: 'task-name' do %>
  <%= task.task %>
  <% if task.due_date.present? %>
    <time class="task-time <%= task.due_date <= Date.today ? 'is-due' : '' %>"
datetime="<%= task.due_date.strftime('%FT%T') %>">
      <%= task.due_date.strftime("%m/%d") %>
    </time>
  <% end %>
<% end %>
```

Edit.html.erb

```
<h1 class="task-header">
  Editing <%= @task.task %>
</h1>

<%= link_to 'Go back', tasks_path, class: ['action-button', 'back-button'] %>

<%= link_to 'Log out', destroy_user_session_path, method: :delete, class:
['action-button', 'log-out-button'] %>

<%= render 'form', task: @task, readonly: false %>
```

_form.html.erb

```
<%= form_with(model: task, local: true) do |form| %>
  <% if task.errors.any? %>
    <div class="errors-container">
      <h2>
        <%= pluralize(task.errors.count, "error") %> prohibited this task from
being saved:
      </h2>

      <ul class="errors-list">
        <% task.errors.full_messages.each do |message| %>
          <li class="errors-item">
            <%= message %>
          </li>
```

```
<%= link_to 'Log out', destroy_user_session_path, method: :delete, class:
['action-button',  'log-out-button'] %>

<div class="tasks-container">
  <div class="tasks-todo">
    <h2 class="tasks-status">
      To-Do
    </h2>

    <ul class="tasks-list">
      <% @tasks.todo.each do |task| %>
        <% if task.user == current_user %>
          <li class="tasks-item">
            <%= render 'task', task: task %>
            <div class="task-buttons">
              <%= link_to fa_icon("pencil"), edit_task_path(task), class:
'task-button' %>
              <%= link_to fa_icon("trash-o"), task, class: 'task-button',
                          method: :delete, data: { confirm: 'Are you sure?' }
%>
            </div>
          </li>
        <% end %>
      <% end %>
    </ul>
  </div>

  <div class="tasks-buttons">
    <%= link_to 'New task', new_task_path, class: "action-button" %>
  </div>

  <div class="tasks-completed">
    <h2 class="tasks-status">
      Completed
    </h2>

    <ul class="tasks-list">
      <% @tasks.completed.each do |task| %>
        <% if task.user == current_user %>
          <li class="tasks-item">
            <%= render 'task', task: task %>
            <div class="task-buttons">
              <%= link_to fa_icon("trash-o"), task, class: 'task-button',
                          method: :delete, data: { confirm: 'Are you sure?' }
%>
            </div>
          </li>
        <% end %>
```

todo-list/app/models/**user.rb**

```
class User < ApplicationRecord
  # Include default devise modules. Others available are:
  # :confirmable, :lockable, :timeoutable and :omniauthable
  devise :database_authenticatable, :registerable,
         :recoverable, :rememberable, :trackable, :validatable

  has_many :tasks
end
```

New.html.erb

```
<h1 class="task-header">
  New task
</h1>

<%= link_to 'Go back', tasks_path, class: ['action-button', 'back-button'] %>

<%= link_to 'Log out', destroy_user_session_path, method: :delete, class:
['action-button', 'log-out-button'] %>

<%= render 'form', task: @task, readonly: false %>
```

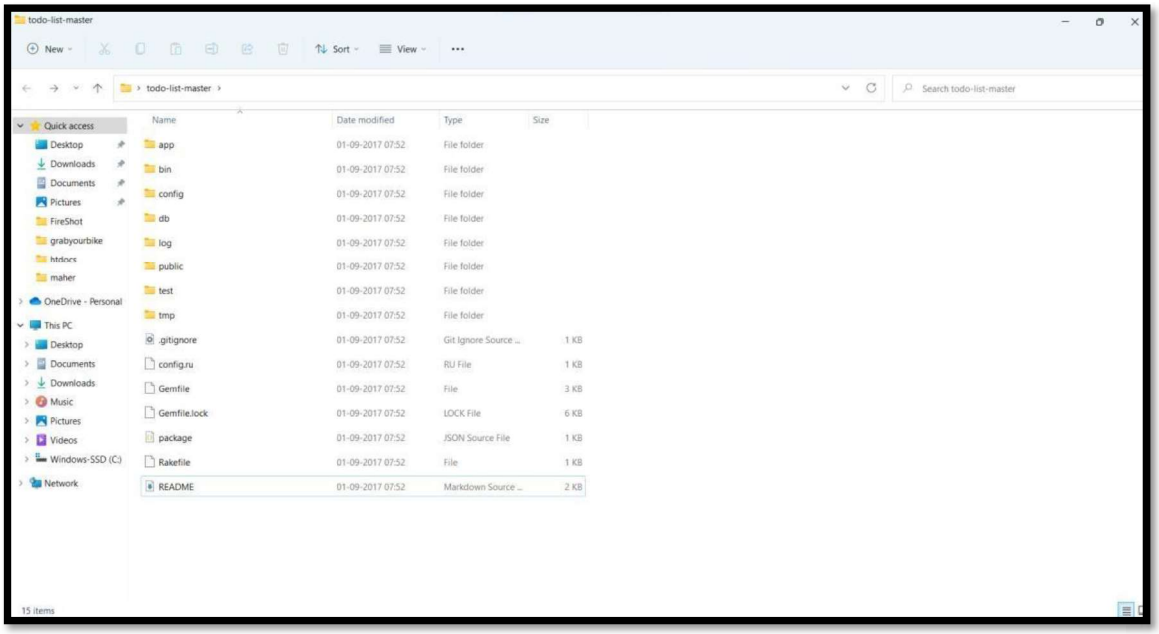
Show.html.erb

```
<h1 class="task-header">
  <%= @task.task %>
</h1>

<%= link_to 'Go back', tasks_path, class: ['action-button', 'back-button'] %>

<%= render 'form', task: @task, readonly: true %>
```

Index.html.erb

```
<h1 class="tasks-header">
  iTasks
</h1>

<p class="notice-container"><%= notice %></p>
```

Project Directory:



todo-list/app/models/application_record.rb

```
class ApplicationRecord < ActiveRecord::Base
  self.abstract_class = true
end
```

todo-list/app/models/**task.rb**

```
class Task < ApplicationRecord
  belongs_to :user

  validates :task, presence: true,
                   length: { minimum: 3 }

  scope :completed, -> {
    where(:completed => true)
  }

  scope :todo, -> {
    where(:completed => false)
  }
end
```

If you would like to set a password for the user, you can do the followingsudo -u
postgres psql
postgres=# \password chris

Final Steps

And now for the moment of truth. Let's create your first Rails application: #### If
you want to use SQLite (not recommended)
rails new myapp
If you want to use MySQL rails new
myapp -d mysql
If you want to use Postgres
Note that this will expect a postgres user with the same username
as your app, you may need to edit config/database.yml to match the# user
you created earlier
rails new myapp -d postgresql
Move into the application directorycd
myapp
If you setup MySQL or Postgres with a username/password, modify the
config/database.yml file to contain the username/password that you specified#
Create the database
rake db:create rails
server
You can now visit http://localhost:3000 to view your new website!
Now that you've got your machine setup, it's time to start building some Rails applications.
If you received an error that said Access denied for user 'root'@'localhost' (using password: NO) then you need to update your config/database.yml file to match the database username and password.

Outcomes:
Able to deploy project in Ruby on rail

Questions:

1. What is ruby on rails.
2. Write steps to install ruby on rails on ubuntu.
3. Explain 'Yield'+ in ruby on rail

If you get a different result for some reason, it means your environment may not be setup properly.

Setting Up A Database

Rails ships with sqlite3 as the default database. Chances are you won't want to use it because it's stored as a simple file on disk.

If you're new to Ruby on Rails or databases in general, I strongly recommend setting up PostgreSQL.

If you're coming from PHP, you may already be familiar with MySQL.

Setting Up MySQL

Rails ships with sqlite3 as the default database. Chances are you won't want to use it because it's stored as a simple file on disk.

sudo apt-get install mysql-server mysql-client libmysqlclient-dev

Installing the libmysqlclient-dev gives you the necessary files to compile the mysql2 gem which is what Rails will use to connect to MySQL when you setup your Rails app.

Setting Up PostgreSQL

For PostgreSQL, we're going to add a new repository to easily install a recent version of Postgres.

sudo apt install postgresql-11 libpq-dev

The postgres installation doesn't setup a user for you, so you'll need to follow these steps to create a user with permission to create databases. Feel free to replace chris with your username.

sudo -u postgres createuser chris -s

```
git clone https://github.com/rbenv/ruby-build.git ~/.rbenv/plugins/ruby-build
echo 'export PATH="$HOME/.rbenv/plugins/ruby-build/bin:$PATH"' >> ~/.bashrc
exec $SHELL

rbenv install 3.0.1
rbenv global 3.0.1
ruby –v
```

The last step is to install Bundler

```
gem install bundler
```

Installing Rails

Choose the version of Rails you want to install:
6.1.3.2 (Recommended)

```
gem install rails -v 6.1.3.2
```

If you're using rbenv, you'll need to run the following command to make the rails executable available:

```
rbenv rehash
```

Now that you've installed Rails, you can run the rails -v command to make sure you have everything installed correctly:

```
rails -v
# Rails 6.1.3.2
```