

List of Lab Exercises

1. Write Python script to print prime number form m to n. where $m < n$
2. Write Python script to create "Book" class with properties "id", "author" and "price". Create 4 Book objects and print details of books on console
3. Write Python script to list files and their sizes from a directory
4. Write Python script for performing simple mathematical calculations using GUI.
5. Write python script to generate Login Screen (GUI) and perform authentication using "client" and "server" as username and password respectively
6. Write Python script to create "Student" table with columns "sno", "sname" and "result" in MySQL server and insert minimum 3 records into the table and print them all on console
7. Write Python script for simple chat application using networking
8. Design and Develop Ruby on Rails web application, which contains a welcome screen and displays the welcome message to the user with his name after entering name using Request and Response objects.
9. Design and Develop Ruby on Rails web application to manipulate Book details using MySQL database.
10. Web pages as follows Add book details screen/page List book details screen/ edit the book details

1. Prime Number Printer (Python)

Aim:

The aim of this experiment is to develop a Python script that efficiently prints prime numbers within a specified range $[m, n]$, where m is less than n . The script should utilize a loop and a primality check function to identify and print the prime numbers in the given range.

Algorithm:

Step 1: Prompt the user for two integers, m and n , representing the range within which prime numbers are generated.

Step 2: Define a function `primeGenerator(i, j)` that iterates from i to j , checks each number for primality, and prints prime numbers.

Step 3: If m is less than n , call `primeGenerator(m, n)`; otherwise, print an error message.

Step 4: Display the prime numbers within the specified range or an error message if the inputs are invalid.

Program:

```
m = int(input("Enter the m: "))
n = int(input("Enter the n: "))

def primeGenerator(i, j):
    while i <= j:
        k = 1
        temp = 0
        while k <= i:
            if i % k == 0:
                temp += 1
            k += 1
        if temp == 2:
            print(i)
        i += 1

if m < n:
    primeGenerator(m, n)
else:
    print("Enter valid numbers!!")
```

Output:

Enter the m: 12

Enter the n: 24

13

17

19

23

2. Book Class (Python)

Aim:

The aim of this Python script is to define a "Book" class with properties such as "id," "author," and "price." The script then instantiates four Book objects and prints their details, demonstrating the creation and utilization of a class in Python.

Algorithm:

Step 1: Create a class named Book. Include an init method to initialize the properties of the class (book_id, author, and price).

Step 2: Create four instances of the Book class with different details.

Step 3: Display the details of each Book object using the print function.

Step 4: Run the script to see the printed details of the four Book objects

Program:

```
class Book:
    def __init__(self, book_id, author, price):
        self.book_id = book_id
        self.author = author
        self.price = price
# Instantiate Book objects
book1 = Book(1, "Author1",19.99)
book2 = Book(2, "Author2",29.99)
book3 = Book(3, "Author3",39.99)
book4 = Book(4, "Author4",49.99)
# Print details of each book
print("Book 1 Details:")
print(f"ID: {book1.book_id}, Author: {book1.author}, Price: ${book1.price}\n")
print("Book 2 Details:")
print(f"ID: {book2.book_id}, Author: {book2.author}, Price: ${book2.price}\n")
print("Book 3 Details:")
print(f"ID: {book3.book_id}, Author: {book3.author}, Price: ${book3.price}\n")
print("Book 4 Details:")
print(f"ID: {book4.book_id}, Author: {book4.author}, Price: ${book4.price}\n")
```

Output:

Book 1 Details:

ID: 1, Author: Author1, Price: \$19.99

Book 2 Details:

ID: 2, Author: Author2, Price: \$29.99

Book 3 Details:

ID: 3, Author: Author3, Price: \$39.99

Book 4 Details:

ID: 4, Author: Author4, Price: \$49.99

3. (Python) Script to List Files and their Sizes

Aim:

The aim of this script is to list the files and their sizes within a specified directory.

Algorithm:

Step 1: Import the necessary module subprocess.

Step 2: Define the command to run the command variable holds the Ubuntu terminal command to be executed, in this case, "ls -l".

Step 3: Try-Except block for error handling

Step 4: Run the command

Program:

```
import subprocess
command = "ls -l"
try:
    output = subprocess.check_output(command, shell=True,
stderr=subprocess.STDOUT)
    output = output.decode('utf-8')
    print(output)
except subprocess.CalledProcessError as e:
    print(f"Error: {e}")
```

Output:

```
===== RESTART: /home/rathinam/dhanushlab3.py =====
total 240
-rwxrwxr-x 1 rathinam rathinam 124 Feb 7 10:10 abhi.sh
-rwxrwxr-x 3 rathinam rathinam 4096 Dec 19 14:04 Android
drwxrwxr-x 11 rathinam rathinam 4096 Jan 30 11:46 AndroidStudioProjects
-rw-rw-r-- 1 rathinam rathinam 493 Nov 21 15:13 biggest number.py
-rw-rw-r-- 1 rathinam rathinam 16158 Jul 31 2023 boo boo.tar.gz
-rw-rw-r-- 1 rathinam rathinam 14 Jan 5 09:36 car
drwxr-xr-x 8 rathinam rathinam 4096 Jan 24 10:25 Desktop
-rw-rw-r-- 1 rathinam rathinam 379 Feb 7 15:48 dhanushlab3.py
drwxr-xr-x 3 rathinam rathinam 4096 Oct 10 13:05 Documents
drwxr-xr-x 2 rathinam rathinam 4096 Feb 7 15:27 Downloads
drwxrwxr-x 23 rathinam rathinam 4096 Aug 3 2023 eclipse-workspace
-rw-rw-r-- 1 rathinam rathinam 118823 Oct 27 09:48 Firefox_wallpaper.png
-rw-rw-r-- 1 rathinam rathinam 21 Jan 5 09:31 flowers
-rw-rw-r-- 1 rathinam rathinam 19 Dec 20 10:34 ml.
-rwxrwxr-x 1 rathinam rathinam 69 Dec 20 10:41 ml.sh
-rw-rw-r-- 1 rathinam rathinam 18 Jan 5 09:43 movies
drwxr-xr-x 2 rathinam rathinam 4096 Jul 20 2023 Music
-rw-rw-r-- 1 rathinam rathinam 35 Jan 5 10:12 names
drwxrwxr-x 3 rathinam rathinam 4096 Sep 27 10:05 oradiag_rathinam
drwxr-xr-x 3 rathinam rathinam 4096 Feb 7 15:45 Pictures
-rwxrwxr-x 1 rathinam rathinam 253 Jan 10 10:31 prg.sh
-rwxrwxr-x 1 rathinam rathinam 36 Jan 10 10:36 prg.sh.save
-rwxrwxr-x 1 rathinam rathinam 196 Feb 7 09:45 priya.sh
drwxr-xr-x 2 rathinam rathinam 4096 Jul 20 2023 Public
drwxrwxr-x 3 rathinam rathinam 4096 Nov 2 12:30 R
drwx----- 7 rathinam rathinam 4096 Dec 19 14:02 snap
drwxr-xr-x 2 rathinam rathinam 4096 Jul 20 2023 Templates
drwxr-xr-x 2 rathinam rathinam 4096 Jul 20 2023 Videos
```

4. (Python) Script for GUI calculator

Aim:

The aim of this script for performing simple mathematical calculations using GUI.

Algorithm:

1. Create a Tkinter window (root) titled "Simple Calculator".
2. Add two entry widgets to the window to allow users to input numbers.
3. Define a list of arithmetic operations (+, -, *, /).
4. For each operation, create a button labeled with the corresponding symbol (+, -, *, /). Associate each button with the operate function using lambda functions, passing the respective operation as an argument.
5. Inside the operate function, retrieve the values entered in the entry widgets, convert them to floats, and perform the selected operation. Handle division by zero gracefully. Update the result label with the calculated result or an error message.
6. Start the main event loop using root.mainloop() to display the GUI and handle user interactions.

Program:

```
import tkinter as tk

def operate(operation):
    try:
        num1 = float(entry1.get())
        num2 = float(entry2.get())
        result = {
            '+': num1 + num2,
            '-': num1 - num2,
            '*': num1 * num2,
            '/': num1 / num2 if num2 != 0 else 'Cannot divide by zero'
        }.get(operation, 'Invalid operation')
        label_result.config(text="Result: " + str(result))
    except ValueError:
        label_result.config(text="Please enter valid numbers")

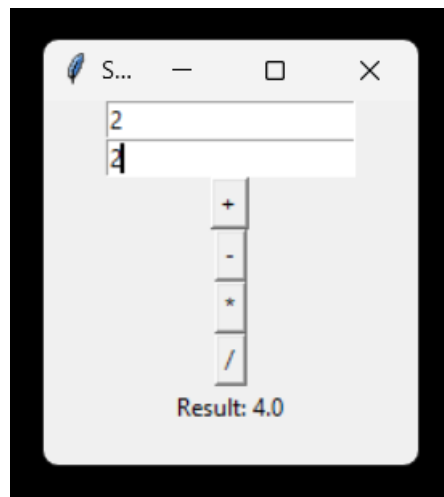
root = tk.Tk()
root.title("Simple Calculator")

entry1 = tk.Entry(root)
entry1.pack()

entry2 = tk.Entry(root)
entry2.pack()
```

```
operations = ['+', '-', '*', '/']  
for op in operations:  
    button = tk.Button(root, text=op, command=lambda operation=op:  
operate(operation))  
    button.pack()  
  
label_result = tk.Label(root, text="")  
label_result.pack()  
  
root.mainloop()
```

Output:



5. (Python) Script for GUI Login Page

Aim:

The aim of this script to generate Login Screen (GUI) and perform authentication using "client" and "server" as username and password respectively

Algorithm:

1. Create a tkinter window with a login form.
2. When the user clicks the login button, check if the entered username is "client" and the password is "server".
3. If the username and password match, display a success message and close the login window.
4. If the username or password is incorrect, display an error message.
5. Allow the user to attempt login again.

Program:

```
import tkinter as tk
from tkinter import *

window = Tk()
window.geometry("500x500")

titleText = Label(window, text="Login Page")
titleText.place(x=30, y=40)

usernameText = Label(window, text="Enter the username:")
usernameText.place(x=30, y=70)
userName = tk.Entry(window, width=40)
userName.place(x=30, y=98)

passwordText = Label(window, text="Enter the password:")
passwordText.place(x=30, y=120)
passwordEntry = Entry(window, width=40, show="*")
passwordEntry.place(x=30, y=140)

def verify():
    userNameData = userName.get()
    passWordData = passwordEntry.get()
    if(userNameData == "client"):
        if(passWordData == "server"):
            window2()
```

```

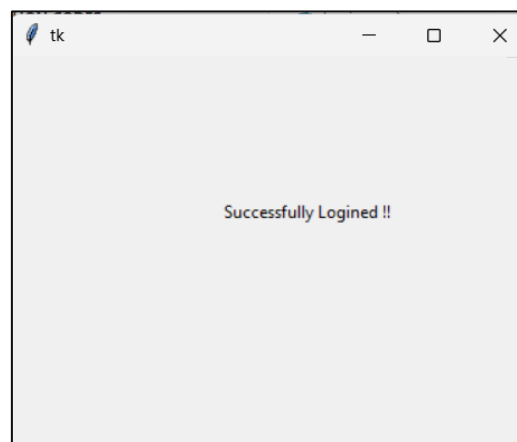
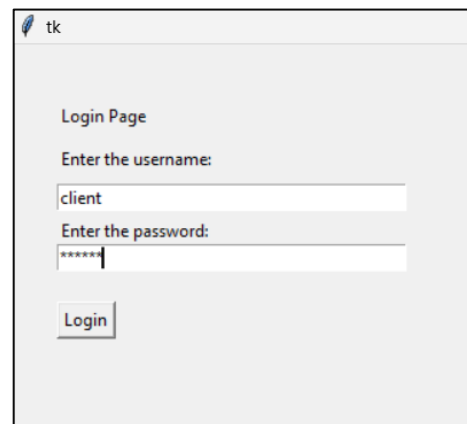
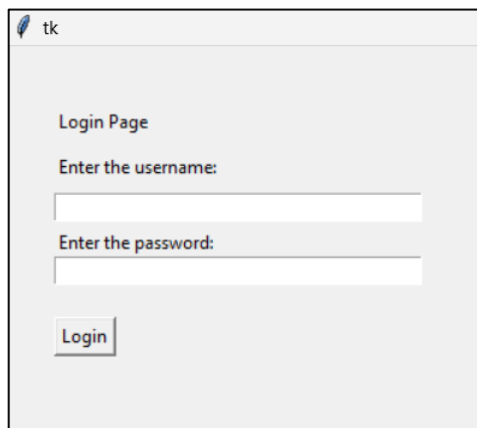
        else:
            userNameError = Label(window,text="Invalid Password !!")
            userNameError.place(x=30, y=220)
        else:
            passwordError = Label(window,text="Invalid Username !!")
            passwordError.place(x=30, y=220)

loginBtn = Button(window, text="Login", command=verify)
loginBtn.place(x=30, y=180)

def window2():
    window.destroy()
    window2 = Tk()
    window2.geometry("500x500")
    sucessText = Label(window2, text="Successfully Logined !!")
    sucessText.place(x=150,y=100)
    window2.mainloop()

window.mainloop()

```

Output:

6. (Python) Script to create table, insert and display data

Aim:

The aim of this Python script to create "Student" table with columns "sno", "sname" and "result" in MySQL server and insert minimum 3 records into the table and print them all on console

Algorithm:

1. Start XAMPP and ensure both Apache and MySQL services are running.
2. Install the necessary Python packages, specifically mysql-connector-python, to interact with MySQL databases.
3. Connect to the MySQL database running on localhost with the username "root" and no password, and select the "studentdata" database.
4. Create a cursor object to execute SQL queries. Execute an SQL command to create a table named "student" with columns for student number (sno), student name (stdName), and result.
5. Construct an SQL INSERT statement to add a record with specific values for student number, student name, and result.
6. Execute the INSERT statement and commit the transaction to make the changes permanent. Print the number of records inserted.

Program:

```
import mysql.connector

mydb = mysql.connector.connect(
    host="localhost",
    user="root",
    password="",
    database="studentdata"
)

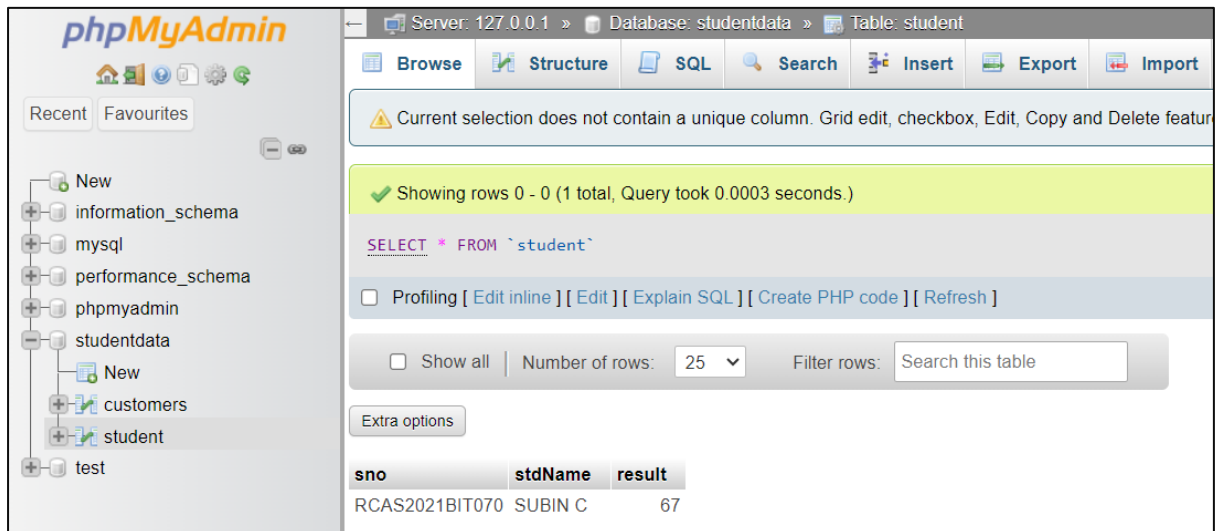
mycursor = mydb.cursor()

mycursor.execute("CREATE TABLE student (sno VARCHAR(255), stdName
VARCHAR(255),result INT(20))")

sql = "INSERT INTO student (sno, stdName, result) VALUES ('RCAS2021BIT070',
'SUBIN C', 67)"
mycursor.execute(sql)
```

```
mydb.commit()
```

```
print(mycursor.rowcount, "record inserted.")
```

Output:

The screenshot shows the phpMyAdmin interface. On the left is the database navigation tree with 'studentdata' selected and 'student' table expanded. The main panel shows the 'Browse' view of the 'student' table. A message states: 'Current selection does not contain a unique column. Grid edit, checkbox, Edit, Copy and Delete features are disabled.' Below this, a green bar indicates 'Showing rows 0 - 0 (1 total, Query took 0.0003 seconds.)'. The SQL query 'SELECT * FROM `student`' is shown. Below the query are links for 'Profiling', 'Edit inline', 'Edit', 'Explain SQL', 'Create PHP code', and 'Refresh'. There are controls for 'Show all', 'Number of rows' (set to 25), and a 'Filter rows' search box. An 'Extra options' button is also present. At the bottom, a table displays the data:

sno	stdName	result
RCAS2021BIT070	SUBIN C	67

7. (Python) simple chat application using networking

Aim:

The aim of this Python script to create simple chat application using networking

Algorithm:

Server Side:

1. Create a socket object for the server.
2. Bind the server socket to the localhost address and port 9999.
3. Listen for incoming connections.
4. Accept a client connection, then enter a loop to receive and send messages.

Client Side:

1. Create a socket object for the client.
2. Connect the client socket to the server at the localhost address and port 9999.
3. Enter a loop to send and receive messages with the server.
4. Prompt the user to enter a message to send to the server, and display messages received from the server.

Program:

Server.py

```
import socket

server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server.bind(("localhost", 9999))

server.listen()

client, addr = server.accept()
print("Connection established .....")

done = False

while not done:
    msg = client.recv(1024).decode('utf-8')
    if msg == 'quit':
        done = True
    else:
        print(msg)
        client.send(input("Enter a message to send to the client (type 'quit' to exit): ").encode('utf-8'))
```

client.py

```
import socket

client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

client.connect(("localhost", 9999))
print("Server is listening send the message .....")

done = False

while not done:
    client.send(input("Enter a message to send to the server (type 'quit' to exit): ").encode('utf-8'))
    msg = client.recv(1024).decode('utf-8')
    if msg == 'quit':
        done = True
    else:
        print(msg)
```

Output:

PROBLEMS	1	OUTPUT	DEBUG CONSOLE	TERMINAL	PORTS
<div> <div> PS E:\ProgramingLanguages\Python Codes\CHAT> python server.py Connection established Hello Enter a message to send to the client (type 'quit' to exit): Hi PS E:\ProgramingLanguages\Python Codes\CHAT> </div> <div> PS E:\ProgramingLanguages\Python Codes\CHAT > python client.py Server is listening send the message Enter a message to send to the server (type 'quit' to exit): Hello Hi Enter a message to send to the server (type 'quit' to exit): quit Enter a message to send to the server (type 'quit' to exit): </div> </div>					

8. Welcome screen for the custom user's name (Ruby Rails)

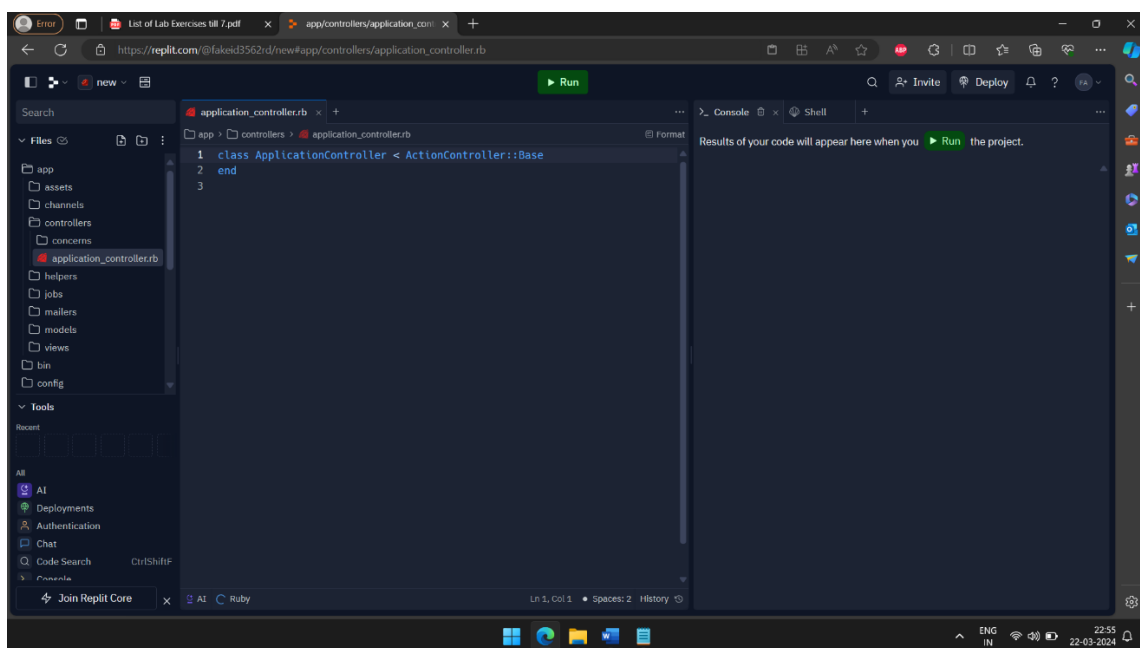
Aim:

Design and Develop Ruby on Rails web application, which contains a welcome screen and displays the welcome message to the user with his name after entering name using Request and Response objects.

Program and Steps:

Step 1:

Visit the website replit.com



Step 2:

create rails project by using the following steps

+ Create Repl -> Search Template -> Rails -> Enter the project name -> Create Repl

Step 3:

Create an application with the help of following shell commands

rails generate controller welcome index

select the rails version (Mentioned in the image)

rubyPackages_3_2.railties 7.1.0

```

~/new$ rails generate controller welcome index
rails: command not installed. Multiple versions of this command were found in Nix.
Select one to run (or press Ctrl-C to cancel):
>
zammad 5.4.1 Zammad, a web-based, open source.
rubyPackages.railties 7.1.0
rubyPackages_3_2.railties 7.1.0
rubyPackages_3_3.railties 7.1.0
mastodon 4.2.1 Self-hosted, globally interconneb

```

Step 4:

Configure the application and make it as a root

Directory: config/routes.rb

add the following code

Rails.application.routes.draw do

get 'welcome/index'

**# Define your application routes per the DSL in
<https://guides.rubyonrails.org/routing.html>**

Defines the root path route ("/")

root "welcome#index"

end

Step 5:

Then add the HTML code to get the name and display welcome text

Directory: app/views/welcome/index.html.erb

Inside the index.html.erb

<h1>Welcome</h1>

<% if @name.present? %>

<p>Welcome, <%= @name %>!</p>

<% else %>

<form action="/welcome/index" method="get">

<label for="name">Please enter your name:</label>

<input type="text" id="name" name="name">

<input type="submit" value="Submit">


```
</form>  
<% end %>
```

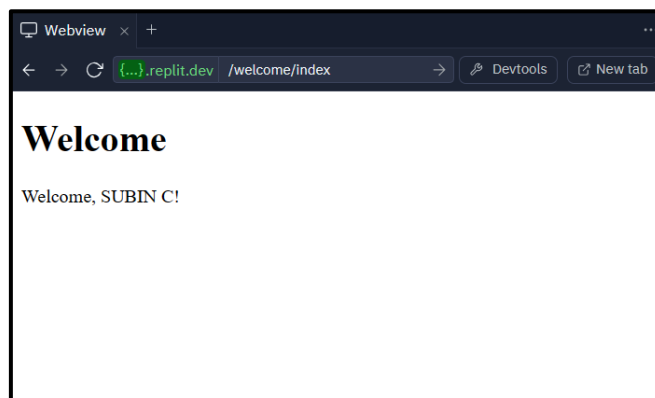
step 6:

Write an get method

Directory: app/controllers/welcome_controller.rb

Write the following code

```
class WelcomeController < ApplicationController  
  def index  
    @name = params[:name]  
  end  
end
```

Output:

9. Manipulate the Book details (Ruby Rails)

Aim:

Design and Develop Ruby on Rails web application to manipulate Book details using MySQL database.

Program and steps:

Step 1:

Setup Ruby on Rails Environment. Ensure you have Ruby and Rails installed on your system.

Step 2:

Generate Model for Books

With the help of following command

```
rails generate model Book title:string author:string description:text
```

Step 3:

Generate Controller for Books

With the help of following command

```
rails generate controller Books
```

Step 4:

Run Database Migration

Run the migration to create the books table in the MySQL database:

```
rails db:create
```

then run the following command to update the changes

```
rails db:migrate
```

Step 5:

Add One Book Data

You can add one book data through Rails console

run the following command to open the rails shell

```
rails console
```

run the following command to add the new book data

```
Book.create(title: "Example Book", author: "John Doe", description: "This is an example book.")
```

Step 6:

Create Views

Create views for listing books and displaying book details.

Directory: `app/views/books/index.html.erb` (List books):

html

```
<h1>Listing Books</h1>
<% @books.each do |book| %>
  <div>
    <h2><%= book.title %></h2>
    <p><strong>Author:</strong> <%= book.author %></p>
    <p><strong>Description:</strong> <%= book.description %></p>
  </div>
<% end %>
```

Step 7:

Define Routes

Define routes for the book-related actions in `config/routes.rb`:

```
Rails.application.routes.draw do
  resources :books
  root to: 'books#index'
end
```

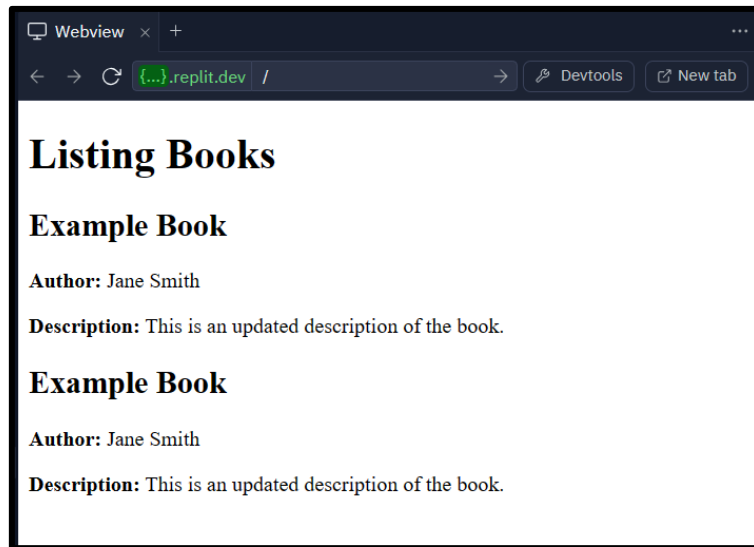
Step 8:

Change the controller code to `@books` is correctly assigned to the list of books retrieved from the database.

Code:

```
class BooksController < ApplicationController
  def index
    @books = Book.all
  end
end
```

Output:



10. Add, Display, Edit Book details (Ruby Rails)

Aim:

Web pages as follows Add book details screen/page List book details screen/ edit the book details

Program and steps:

Step 1:

Setup Ruby on Rails Environment. Ensure you have Ruby and Rails installed on your system.

Step 2:

Generate Model for Books

With the help of following command

```
rails generate model Book title:string author:string description:text
```

Step 3:

Generate Controller for Books

With the help of following command

```
rails generate controller Books
```

Step 4:

Run Database Migration

Run the migration to create the books table in the MySQL database:

```
rails db:create
```

then run the following command to update the changes

```
rails db:migrate
```

Step 5:

Add One Book Data

You can add one book data through Rails console

run the following command to open the rails shell

```
rails console
```

run the following command to add the new book data

```
Book.create(title: "Example Book", author: "John Doe", description: "This is an example book.")
```

Step 6:

Create Views

Create views for listing books and displaying book details.

Directory: `app/views/books/index.html.erb` (List books):

html

```
<h1>Listing Books</h1>
<% @books.each do |book| %>
  <div>
    <h2><%= book.title %></h2>
    <p><strong>Author:</strong> <%= book.author %></p>
    <p><strong>Description:</strong> <%= book.description %></p>
  </div>
<% end %>
```

Step 7:

Define Routes

Define routes for the book-related actions in `config/routes.rb`:

```
Rails.application.routes.draw do
  resources :books
  root to: 'books#index'
end
```

Step 8:

Change the controller code to `@books` is correctly assigned to the list of books retrieved from the database.

Code:

```
class BooksController < ApplicationController
  def index
    @books = Book.all
  end
end
```

Step 8:

Edit the book details using the following command in Rails shell

Command:

```
Book.find_by(title: "Example Book").update(author: "Jane Smith", description: "This is an updated description of the book.")
```

Output:

