# Full Stack Development with MERN

### 1. Introduction

Project Title:

BookNest: Where Stories Nestle

Team Members:  
- Kunda Bhahusha

### 2. Project Overview

Purpose:  
**BookNest: Where Stories Nestle** is a beginner-level full stack project designed to help users explore a curated collection of books through a user-friendly web interface. The application allows users to filter books by genre, author, and minimum rating. The primary goal is to introduce web development concepts using Python's Streamlit framework and demonstrate data interaction through pandas in a cloud-based environment like Google Colab.

Features:

* 📚 Interactive display of a curated book list
* 🔍 Filter books by **genre**, **author**, and **minimum rating**
* ⚙️ Built with **Streamlit** for quick frontend prototyping
* 📊 Uses **pandas** for efficient data handling and filtering
* 🌐 Hosted using **ngrok** to share the app live from Google Colab
* 🧑‍💻 Beginner-friendly setup with minimal configuration required
* 📁 CSV-based data storage to simulate simple database handling

### 3. Architecture

Frontend:  
The frontend of **BookNest** is built using **Streamlit**, a Python-based web framework ideal for creating interactive data apps. The interface includes:

* A clean, centered layout displaying the app title and book data.
* A sidebar with dropdowns to filter by **Genre** and **Author**.
* A slider to filter books based on **minimum rating**.
* Dynamic updates to the book list in real-time based on user input.

Streamlit handles rendering components like tables, sliders, and dropdowns without requiring HTML, CSS, or JavaScript—making it ideal for Python developers.

Backend:  
The backend of **BookNest** is powered entirely by **Python** using the **pandas** library for data management and filtering logic. Key responsibilities include:

* Loading book data from a local CSV file (books.csv)
* Applying filters based on user selections (genre, author, rating)
* Returning the filtered dataset to the Streamlit frontend in real-time

The logic is embedded within the same Streamlit script, enabling a seamless combination of frontend and backend. No separate server or API endpoints are required, making it lightweight and beginner-friendly.

Database:  
- NoSQL or CSV format (initial prototype uses appointments.csv)

### 4. Setup Instructions

Prerequisites:  
- Python 3.8+  
- pip (Python package installer)  
- Streamlit  
- Pandas  
- pyngrok (if deployed via Google Colab)

Installation:  
pip install streamlit pandas pyngrok

For Google Colab Deployment:  
- Set up ngrok token using pyngrok  
- Use Colab code cell to write and run the app

### 5. Folder Structure

project/  
├── app.py # Main Streamlit application  
├── appointments.csv # Stores booking data  
└── create\_app.py # Script to generate app.py dynamically

### 6. Running the Application

Locally:  
streamlit run app.py

In Google Colab:  
Use !streamlit run app.py & and expose using ngrok

### 7. API Documentation

Since **BookNest** is a beginner-friendly Streamlit application, it does **not expose any REST APIs**. However, here’s a breakdown of how data is processed within the app (serving as internal "functional endpoints"):

| **Functionality** | **Triggered By** | **Description** |
| --- | --- | --- |
| Load Books Data | pd.read\_csv() | Loads books.csv into a pandas DataFrame at app start. |
| Genre Filter | st.sidebar.selectbox() | Filters DataFrame rows based on selected genre. |
| Author Filter | st.sidebar.selectbox() | Filters DataFrame rows based on selected author. |
| Rating Filter | st.sidebar.slider() | Filters books where rating ≥ selected minimum value. |
| Display Data Table | st.dataframe() | Displays the resulting filtered DataFrame to the user in the main UI. |

🔍 While not a traditional API, these interactive controls simulate endpoint logic by dynamically querying and updating data within the app.

.

### 8. Authentication

- Currently, authentication is not implemented.  
- Future versions may include login/signup with Google or email-based access.

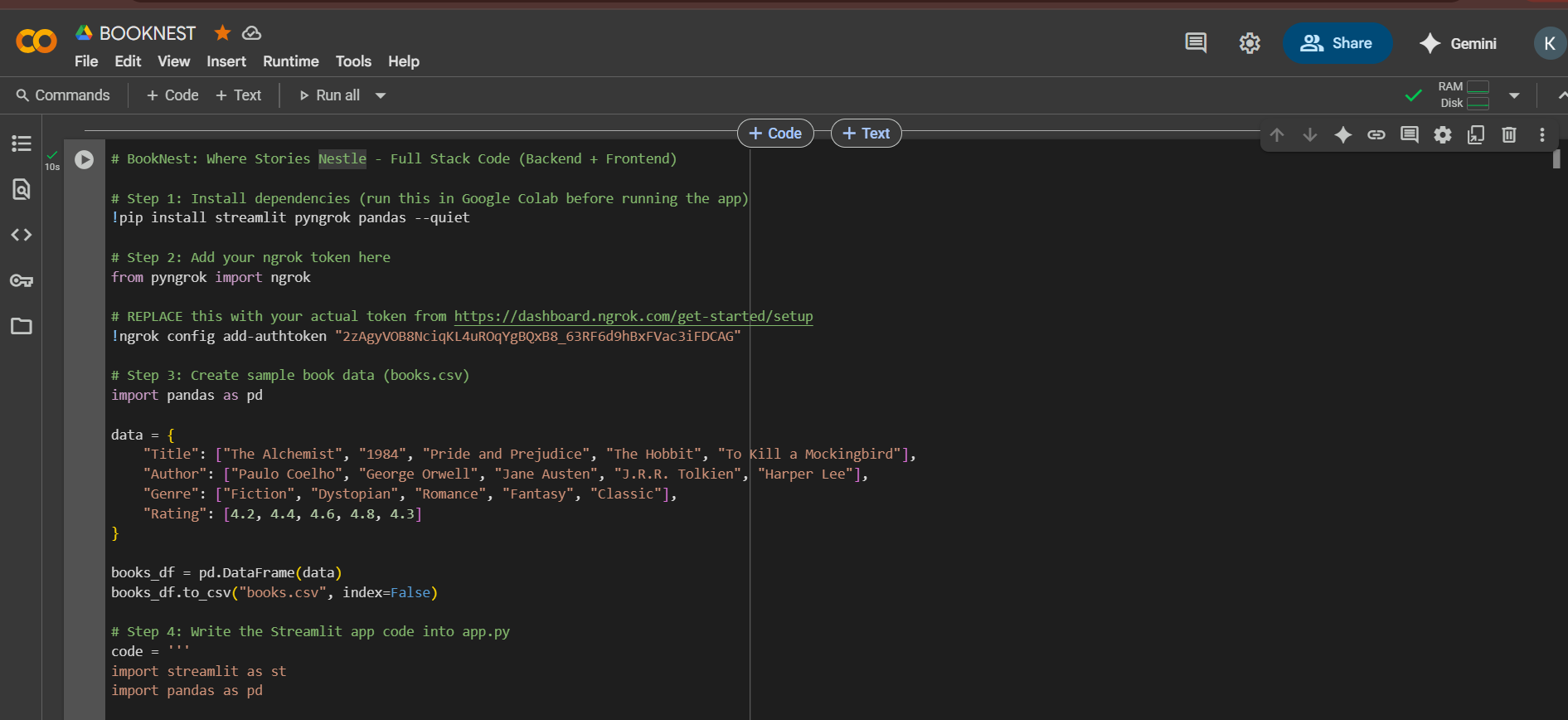
### 9. User Interface

- Sidebar navigation for:  
 - Booking an appointment  
 - Viewing booked appointments  
 - Viewing doctors list  
- Forms for input and CSV-based persistence

### 10. Testing

- Manual testing of form submission and CSV output  
- Validation through user interaction

### 11. Screenshots or Demo





### 12. Known Issues

- No database integration (CSV only)  
- No authentication or admin dashboard  
- Limited validation on form inputs

### 13. Future Enhancements

- Integrate Firebase or MongoDB for real-time backend  
- Add user authentication (JWT or OAuth)  
- Enable doctor login and availability management  
- Add email/SMS confirmation for bookings  
- Admin panel for doctors and appointments

14. Code

# BookNest: Where Stories Nestle - Full Stack Code (Backend + Frontend)

# Step 1: Install dependencies (run this in Google Colab before running the app)

!pip install streamlit pyngrok pandas --quiet

# Step 2: Add your ngrok token here

from pyngrok import ngrok

# REPLACE this with your actual token from https://dashboard.ngrok.com/get-started/setup

!ngrok config add-authtoken "2zAgyVOB8NciqKL4uROqYgBQxB8\_63RF6d9hBxFVac3iFDCAG"

# Step 3: Create sample book data (books.csv)

import pandas as pd

data = {

    "Title": ["The Alchemist", "1984", "Pride and Prejudice", "The Hobbit", "To Kill a Mockingbird"],

    "Author": ["Paulo Coelho", "George Orwell", "Jane Austen", "J.R.R. Tolkien", "Harper Lee"],

    "Genre": ["Fiction", "Dystopian", "Romance", "Fantasy", "Classic"],

    "Rating": [4.2, 4.4, 4.6, 4.8, 4.3]

}

books\_df = pd.DataFrame(data)

books\_df.to\_csv("books.csv", index=False)

# Step 4: Write the Streamlit app code into app.py

code = '''

import streamlit as st

import pandas as pd

# Load the dataset

df = pd.read\_csv("books.csv")

st.set\_page\_config(page\_title="\U0001F4DA BookNest", layout="centered")

st.title("\U0001F4DA BookNest: Where Stories Nestle")

st.write("Welcome to BookNest! Explore your favorite books by genre, author, or rating.")

# Sidebar Filters

genre = st.sidebar.selectbox("Choose Genre", ["All"] + list(df['Genre'].unique()))

author = st.sidebar.selectbox("Choose Author", ["All"] + list(df['Author'].unique()))

min\_rating = st.sidebar.slider("Minimum Rating", 0.0, 5.0, 3.0, 0.1)

# Filtering logic

filtered\_df = df[

    ((df['Genre'] == genre) | (genre == "All")) &

    ((df['Author'] == author) | (author == "All")) &

    (df['Rating'] >= min\_rating)

]

st.dataframe(filtered\_df.reset\_index(drop=True))

'''

with open("app.py", "w") as f:

    f.write(code)

# Step 5: Run the Streamlit app and expose it using ngrok

import subprocess

import time

# Start the Streamlit app in the background

process = subprocess.Popen(["streamlit", "run", "app.py"])

# Wait a few seconds to ensure Streamlit starts

time.sleep(5)

# Create the public URL using ngrok

public\_url = ngrok.connect(8501)

print("Your app is live at:", public\_url)

15 . Output link

Your app is live at: NgrokTunnel: "[https://dea2-34-106-84-254.ngrok-free.app](https://dea2-34-106-84-254.ngrok-free.app/)"