Report On

Cloud Based Library Management System

Submitted in partial fulfillment of the requirements of the Course project in

Semester VI of Third Year Computer Engineering

by

Prathmesh Tiwari (Roll No. 56)

Vedant Vasaikar (Roll No. 57)

Prashant Rajput (Roll No. )

Supervisor

Prof. Aarti Puharan



**University of Mumbai**

**Vidyavardhini's College of Engineering & Technology**

**Department of Computer Engineering**

A black and white logo

Description automatically generated

**(2023-24)**

# Vidyavardhini's College of Engineering & Technology Department of Computer Engineering

**CERTIFICATE**

This is to certify that the project entitled “Cloud Based Library Management System” is a bonafide work of "Prathmesh Tiwari (Roll No. 56), Vedant Vasaikar (Roll No. 57), Name3 Surname3 (Roll No. xx), Name4 Surname4 (Roll No. xx)" submitted to the University of Mumbai in partial fulfillment of the requirement for the Course project in semester VI of Third Year Computer Engineering.

**Supervisor**

Prof. Name Surname

Internal Examiner External Examiner

Dr Megha Trivedi Dr. H.V. Vankudre

Head of Department Principal

**Abstract:**

Our cloud-based library management system is a revolutionary solution that transforms traditional library operations into a streamlined and efficient process. Leveraging the power of Firebase, a versatile cloud platform, our system ensures secure data storage, real-time synchronization, and seamless user authentication. Administrators benefit from a comprehensive set of tools for managing book catalogs, monitoring inventory levels, and processing book requests with ease and efficiency. On the user side, our system offers a user-friendly interface for browsing available books, making requests, and tracking assigned books. Real-time updates on book availability provide users with the most up-to-date information, enhancing their overall experience. The system's robust authentication mechanisms ensure data security, protecting sensitive information and ensuring only authorized access. Overall, our cloud-based library management system represents a modern and scalable solution that enhances accessibility, efficiency, and security in library operations, benefiting both administrators and users alike.

## Index

1. **Introduction** 
   1. Introduction
   2. Problem Statement
   3. Scope

1. **Content Analysis** 
   1. Block Diagram
   2. Description
   3. Working
2. **Module Description**
3. **Description of Requirement**
   1. Software
   2. Hardware and its programming
4. **Code**
5. **Results**
6. **Conclusion**
7. **References**

## Chapter 1: Introduction

**1.1 Introduction:**

The cloud-based library management system presented herein represents a significant advancement in the realm of library administration and user experience. Traditional library management systems often face challenges related to scalability, accessibility, and security. In response to these challenges, our project leverages cloud computing technology, specifically Firebase, to address these issues and provide a modern solution for library management.

By utilizing Firebase's features such as Firestore database and Authentication, our system ensures efficient data storage, real-time synchronization, and secure user authentication. This enables administrators to manage book catalogs, monitor inventory, and process book requests with ease and efficiency. Meanwhile, users benefit from a user-friendly interface for browsing available books, making requests, and tracking assigned books. Real-time updates on book availability enhance user experience, while robust authentication mechanisms guarantee data security. This project aims to revolutionize library operations by providing a scalable, accessible, and secure solution that enhances both administrator productivity and user satisfaction.

**1.2 Problem Statement:**

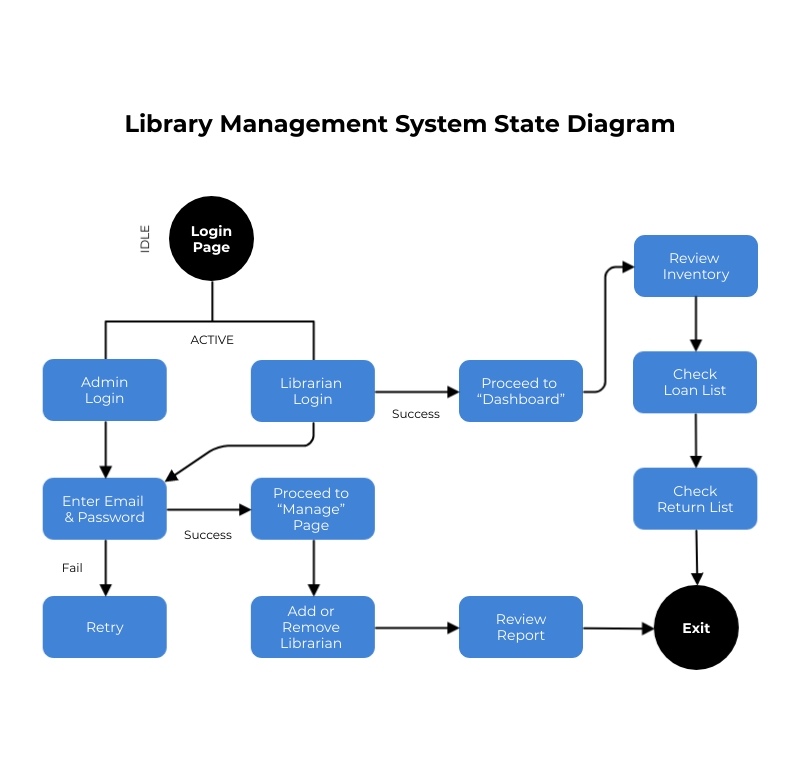
## The conventional library management systems face significant challenges related to scalability, accessibility, and security. These systems often struggle to efficiently handle large volumes of data, resulting in performance bottlenecks and hindering scalability. Moreover, limited accessibility to library resources remotely impedes user engagement and satisfaction, particularly in today's digital era where seamless access is expected. Security concerns further exacerbate the situation, with outdated security measures leaving user data vulnerable to breaches. To address these challenges, our project proposes a cloud-based library management system powered by Firebase. By leveraging cloud infrastructure, the system aims to overcome scalability limitations and enhance accessibility, allowing users to access resources remotely. Robust security features implemented through Firebase Authentication will ensure the confidentiality and integrity of user data, providing a comprehensive solution that revolutionizes library management. With Firebase, the system will utilize Firestore database for efficient data storage and real-time synchronization, enabling seamless management of book catalogs, user accounts, and transaction records. Administrators will have access to streamlined functionalities for managing inventory levels, processing book requests, and monitoring system performance. Users, on the other hand, will benefit from a user-friendly interface for browsing available books, making requests, and tracking assigned books. Real-time updates on book availability will enhance the user experience, while Firebase Authentication will guarantee secure access to the system. Overall, our cloud-based library management system represents a modern, scalable, and secure solution that addresses the challenges faced by traditional systems, providing enhanced accessibility, efficiency, and security for administrators and users alike.

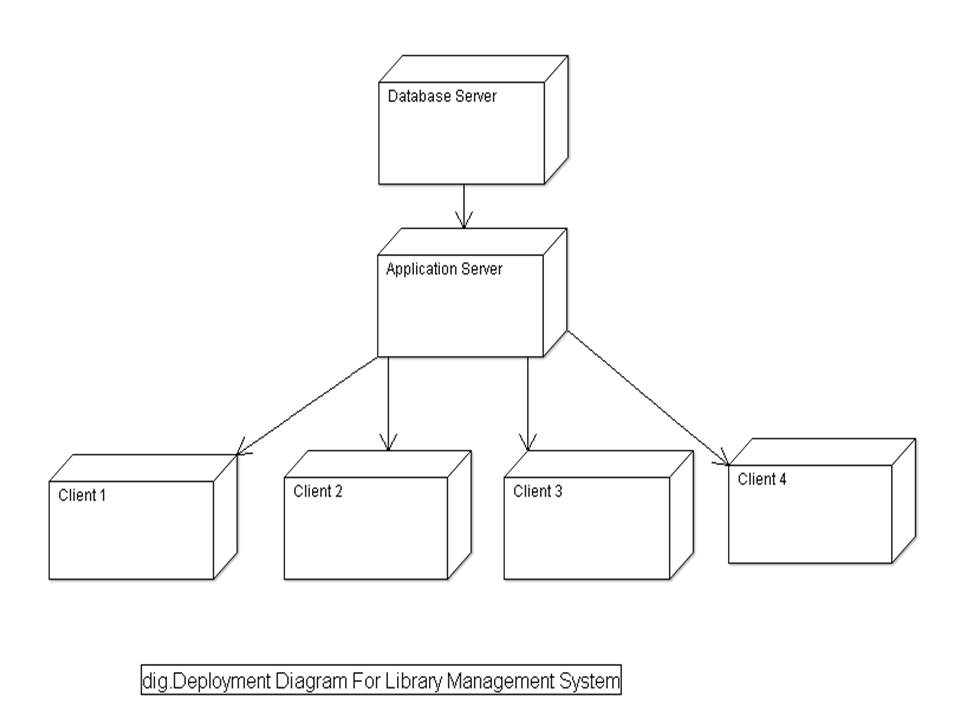
**1.3 Scope:**

The scope of our cloud-based library management system project encompasses several key components aimed at revolutionizing traditional library operations. Firstly, the system will focus on developing a robust and scalable infrastructure using Firebase, leveraging its Firestore database for efficient data storage and real-time synchronization. This infrastructure will enable seamless management of book catalogs, user accounts, and transaction records, ensuring smooth and efficient library operations. Secondly, the project will prioritize enhancing accessibility for users by providing a user-friendly interface that allows for easy browsing of available books, making requests, and tracking assigned books. This includes implementing features such as real-time updates on book availability and intuitive search functionalities to enhance the overall user experience. Additionally, the system will incorporate robust security measures, utilizing Firebase Authentication to ensure secure access to the system and protect sensitive user data from unauthorized access or breaches. By addressing these key aspects, our project aims to provide a comprehensive solution that improves scalability, accessibility, and security in library management, ultimately enhancing efficiency and user satisfaction.

## Chapter 2: Content Analysis

**2.1 Block Diagram:**

****



**2.2 Description:**

Our cloud-based library management system is comprised of several interconnected modules, each serving a specific purpose to ensure the smooth functioning of the system. These modules work collaboratively to provide administrators and users with a seamless and efficient library management experience.

1. Firebase Integration Module:

The Firebase Integration module serves as the backbone of our system, providing the necessary infrastructure for data storage, real-time synchronization, and user authentication. It integrates Firebase's Firestore database and Authentication services, allowing us to store and retrieve data securely while ensuring seamless synchronization across multiple devices in real-time. This module lays the foundation for other modules to build upon, enabling efficient management of user accounts, book catalogs, and transaction records.

2. Administrator Interface Module:

The Administrator Interface module caters to the needs of library administrators, offering a comprehensive set of tools and functionalities to manage various aspects of the library system. Administrators can add new books to the catalog, update existing book information, monitor inventory levels, process book requests, and generate reports on library activities. This module provides a user-friendly interface that simplifies administrative tasks, empowering administrators to efficiently oversee library operations.

3. User Interface Module:

The User Interface module focuses on providing a seamless and intuitive interface for library users to interact with the system. Users can browse available books, search for specific titles or authors, make book assignment requests, and track assigned books. The interface is designed to be user-friendly and accessible, catering to users of all technical backgrounds and abilities. Real-time updates on book availability ensure that users have the most up-to-date information, enhancing their overall experience.

4. Security Module:

The Security module is dedicated to ensuring the confidentiality and integrity of user data within the system. Leveraging Firebase Authentication, this module implements robust security measures to authenticate users and protect sensitive information from unauthorized access or breaches. It employs industry-standard encryption techniques and access controls to safeguard user data, instilling confidence in the system's reliability and security.

5. Data Analytics Module:

The Data Analytics module provides administrators with insights into library usage patterns, user preferences, and inventory management trends. By analyzing data collected from user interactions and transaction records, this module generates actionable insights that enable administrators to make informed decisions about resource allocation, collection development, and service improvements. It empowers administrators to optimize library operations and enhance user satisfaction based on data-driven insights.

6. Book Request Management Module:

The Book Request Management module facilitates the process of handling book assignment requests from users. When users request to borrow a book, this module manages the assignment process, including checking the availability of the requested book, assigning the book to the user, and updating the inventory accordingly. It also handles the scheduling of book return dates and sends notifications to users about the status of their requests. By automating the book assignment process, this module reduces administrative burden and improves efficiency.

7. Inventory Management Module:

The Inventory Management module is responsible for tracking and managing the library's inventory of books. It monitors the quantity of each book available in the catalog, updates inventory levels when books are borrowed or returned, and generates alerts for low-stock items. Additionally, this module provides administrators with tools to categorize books by genre, author, or other criteria, facilitating easier navigation and search for users.

8. Reporting and Analytics Module:

The Reporting and Analytics module offers advanced reporting capabilities to administrators, allowing them to generate customizable reports on various aspects of library operations. This module enables administrators to track key performance indicators, such as book circulation rates, user engagement metrics, and popular book genres. By visualizing data in graphs and charts, administrators gain valuable insights into library usage trends, which can inform decision-making processes and strategic planning initiatives.

Overall, our cloud-based library management system comprises these modules, each playing a crucial role in providing a modern, scalable, and secure solution for library management. Together, they form a cohesive ecosystem that streamlines administrative tasks, enhances user experience, and ensures the efficient operation of the library system.

**2.3 Working:**

The working of our cloud-based library management system begins with the integration of Firebase, which serves as the backbone of the system. Firebase Authentication ensures secure access to the system for both administrators and users. Upon logging in, administrators are presented with a comprehensive interface where they can manage the library catalog, monitor inventory levels, and process book requests. Users, on the other hand, access a user-friendly interface where they can browse available books, search for specific titles or authors, and make book assignment requests. When a user requests to borrow a book, the system checks the availability of the requested book and assigns it to the user if available. Real-time updates on book availability and notifications about the status of book requests enhance the user experience. The system also includes modules for inventory management, which tracks book quantities and generates alerts for low-stock items, and reporting and analytics, which provides administrators with insights into library usage trends and performance metrics. Additionally, robust security measures ensure the confidentiality and integrity of user data throughout the system. Overall, the system provides a modern, scalable, and secure solution for efficient library management, benefiting both administrators and users alike. The functioning of our cloud-based library management system is a multifaceted process that encompasses several key components working together seamlessly to ensure efficient library operations. At the core of the system lies the Firebase integration, which provides the necessary infrastructure for data storage, synchronization, and authentication. Firebase Authentication enables secure access to the system, allowing administrators and users to interact with the system's functionalities. Administrators are empowered with tools to manage the library catalog, including adding new books, updating existing entries, and monitoring inventory levels. They can also process book requests from users, overseeing the assignment of books and tracking their circulation within the library. Meanwhile, users experience a user-friendly interface where they can browse through available books, search for specific titles or authors, and submit requests to borrow books. The system employs real-time updates to ensure users have access to the most current information regarding book availability and request status. Additionally, the system includes modules for inventory management, which tracks book quantities and generates alerts for low-stock items, and reporting and analytics, which provides administrators with valuable insights into library usage patterns and performance metrics. These insights inform decision-making processes and enable administrators to optimize library resources and services. Furthermore, robust security measures implemented throughout the system ensure the confidentiality and integrity of user data, safeguarding against unauthorized access or breaches. By providing a modern, scalable, and secure solution for library management, our system enhances efficiency, accessibility, and user satisfaction, making it an indispensable tool for libraries in the digital age.

**Chapter 3: Module Description**

* User Authentication and Access Control:

1. Implement secure user authentication mechanisms to ensure only authorized users can access the system.
2. Utilize access control lists (ACLs) to define user roles and permissions, such as librarian, administrator, or patron.

* Catalog Management:

1. Enable librarians to add, edit, and delete library resources, including books, journals, and multimedia items.
2. Implement search and filter functionalities to facilitate easy navigation and retrieval of resources.

* Circulation Management:

1. Enable librarians to manage circulation processes, including check-in, check-out, and renewal of library materials.
2. Provide real-time visibility into the availability and status of library items.

* User Management:

1. Allow librarians to register new users, update user information, and manage user accounts.
2. Implement features such as fine management and hold requests to enhance user satisfaction.

* Reporting and Analytics:

1. Generate comprehensive reports on library usage, circulation statistics, and inventory management.
2. Utilize data analytics to gain insights into user preferences, popular resources, and emerging trends.

* Integration with External Systems:

1. Integrate with external systems, such as library databases, digital repositories, and payment gateways, to enhance functionality and interoperability.
2. Ensure seamless data synchronization and compatibility with existing library infrastructure.

* Customization and Extensibility:

1. Provide libraries with the flexibility to customize the system according to their unique requirements and workflows.
2. Support plugin architecture and API integrations to extend the system's capabilities and integrate with third-party services.

* User-Friendly Interface:

1. Design an intuitive and user-friendly interface using Streamlit's interactive components and visualizations.
2. Prioritize usability and accessibility to ensure a positive user experience for librarians and patrons alike.

* Book Request Management Module:

1. The Book Request Management module automates the process of handling book assignment requests from users.
2. When a user requests to borrow a book, this module checks the availability of the requested book, assigns it to the user if available, and updates the inventory accordingly.
3. It also manages the scheduling of book return dates, sends notifications to users about the status of their requests, and handles the reservation of books in case of unavailability.
4. By streamlining the book request process, this module reduces administrative burden and ensures efficient management of library resources.

* Inventory Management Module:

1. The Inventory Management module tracks and manages the library's inventory of books, ensuring optimal utilization of resources and timely replenishment of stock. It monitors the quantity of each book available in the catalog, updates inventory levels in real-time, and generates alerts for low-stock items.
2. Administrators can categorize books by genre, author, or other criteria, facilitating easier navigation and search for users.
3. The module provides insights into inventory trends and usage patterns, enabling administrators to make informed decisions about collection development and resource allocation.

**Chapter 4: Description of Requirement**

The requirements for developing the library management system, including software, hardware, and programming aspects:

* 1. **Software Requirements:**

1. Python: The core programming language for developing the application.

2. Firebase Firestore: Firestore is used as the database to store user data, book data, and book requests.

3. Firebase Admin SDK: Required to interact with Firebase services programmatically from Python.

4. Streamlit: Used for building the web application interface. Streamlit allows for easy and interactive data visualization and user interaction.

5. Pandas: Utilized for data manipulation and to create data frames for displaying user and book data.

6. Datetime: Required for handling date and time operations within the application.

**4.2 Hardware Requirements:**

1. Server: A server to host the application. This could be a cloud-based server or a local server depending on deployment preferences.

2. Storage: Adequate storage space to store user data, book data, and any other application-related files.

3. Internet Connectivity: Required for accessing Firebase services and for users to access the web application.

**4.3 Programming:**

1. Backend Logic: Python is used to write the backend logic for the application. This includes functions for interacting with Firebase Firestore, handling user authentication, fetching data, and processing requests.

2. Frontend Interface: Streamlit is used to design the frontend interface of the application. This involves creating interactive elements such as text inputs, buttons, and data tables for users to interact with.

3. Database Operations: Firebase Firestore is accessed using the Firebase Admin SDK to perform database operations such as adding, updating, and retrieving data.

4. Authentication: User authentication is implemented to ensure secure access to the application. This involves verifying user credentials against stored data in Firestore.

5. Data Presentation: Data fetched from Firestore is presented to users in a readable format using Pandas data frames and Streamlit components.

**Overall Process:**

1. Database Setup: Set up Firebase Firestore and initialize it with collections for users, books, and book requests.

2. Backend Development: Write Python functions to interact with Firebase Firestore for CRUD operations, user authentication, and data processing.

3. Frontend Development: Design the user interface using Streamlit components, including input fields, buttons, and data tables.

4. Integration: Integrate the frontend and backend components to ensure seamless communication between the user interface and the database.

5. Testing: Test the application thoroughly to ensure all features work as expected and handle edge cases appropriately.

6. Deployment: Deploy the application on a server accessible to users, ensuring proper security measures are in place.

These requirements cover the software, hardware, and programming aspects necessary for developing the library management system. Each component plays a crucial role in creating a functional and user-friendly application.

**Chapter 5:** Implementation Code:

import firebase\_admin

from firebase\_admin import credentials , firestore

import streamlit as st

import pandas as pd

import datetime

cred = credentials.Certificate("cloudcomputing-7dbb6-firebase-adminsdk-noc16-a0f0816021.json")

*#firebase\_admin.initialize\_app(cred)*

db = firestore.client()

*# users = [*

*#     {*

*#         "student\_name": "Vedant",*

*#         "clg\_id": "1234567890",*

*#         "study\_years": "2021"*

*#     },*

*#     {*

*#         "student\_name": "Advait",*

*#         "clg\_id": "0987654321",*

*#         "study\_years": "2022"*

*#     },*

*#     {*

*#         "student\_name": "Leander",*

*#         "clg\_id": "4567890123",*

*#         "study\_years": "2023"*

*#     },*

*#     {*

*#         "student\_name": "Shivaraj",*

*#         "clg\_id": "3210987654",*

*#         "study\_years": "2024"*

*#     },*

*#     {*

*#         "student\_name": "Dhrub",*

*#         "clg\_id": "9876543210",*

*#         "study\_years": "2025"*

*#     },*

*#     {*

*#         "student\_name": "Pratik",*

*#         "clg\_id": "5678901234",*

*#         "study\_years": "2021"*

*#     },*

*#     {*

*#         "student\_name": "Irfan",*

*#         "clg\_id": "2345678901",*

*#         "study\_years": "2022"*

*#     },*

*#     {*

*#         "student\_name": "Chaitanya",*

*#         "clg\_id": "1098765432",*

*#         "study\_years": "2023"*

*#     },*

*#     {*

*#         "student\_name": "Prathmesh",*

*#         "clg\_id": "8901234567",*

*#         "study\_years": "2024"*

*#     },*

*#     {*

*#         "student\_name": "Prashant",*

*#         "clg\_id": "4321098765",*

*#         "study\_years": "2025"*

*#     }*

*# ]*

*# for user in users:*

*#     db.collection("users").add(user)*

*# print("Data added successfully.")*

*# books = [*

*#     {*

*#         "Name": "Book 1",*

*#         "Author": "Author 1",*

*#         "Genre": "Genre 1",*

*#         "Quantity": 5*

*#     },*

*#     {*

*#         "Name": "Book 2",*

*#         "Author": "Author 2",*

*#         "Genre": "Genre 2",*

*#         "Quantity": 3*

*#     },*

*#     {*

*#         "Name": "Book 3",*

*#         "Author": "Author 3",*

*#         "Genre": "Genre 3",*

*#         "Quantity": 7*

*#     },*

*#     {*

*#         "Name": "Book 4",*

*#         "Author": "Author 4",*

*#         "Genre": "Genre 1",*

*#         "Quantity": 2*

*#     },*

*#     {*

*#         "Name": "Book 5",*

*#         "Author": "Author 5",*

*#         "Genre": "Genre 2",*

*#         "Quantity": 4*

*#     },*

*#     {*

*#         "Name": "Book 6",*

*#         "Author": "Author 6",*

*#         "Genre": "Genre 3",*

*#         "Quantity": 6*

*#     },*

*#     {*

*#         "Name": "Book 7",*

*#         "Author": "Author 7",*

*#         "Genre": "Genre 1",*

*#         "Quantity": 8*

*#     },*

*#     {*

*#         "Name": "Book 8",*

*#         "Author": "Author 8",*

*#         "Genre": "Genre 2",*

*#         "Quantity": 1*

*#     },*

*#     {*

*#         "Name": "Book 9",*

*#         "Author": "Author 9",*

*#         "Genre": "Genre 3",*

*#         "Quantity": 10*

*#     },*

*#     {*

*#         "Name": "Book 10",*

*#         "Author": "Author 10",*

*#         "Genre": "Genre 1",*

*#         "Quantity": 3*

*#     }*

*# ]*

*# for book in books:*

*#     db.collection("books").add(book)*

def fetch\_books():

    books = db.collection("books").get()

    bookList = []

    for book in books:

        book\_data = {

            "Name": book.get("Name"),

            "Author": book.get("Author"),

            "Genre": book.get("Genre"),

            "Quantity": book.get("Quantity")

        }

        bookList.append(book\_data)

    return bookList

def login(username):

    query = db.collection("users").where("student\_name", "==", username).limit(1).get()

    for doc in query:

        stored\_clg\_id = doc.get("clg\_id")

    return stored\_clg\_id

def fetch\_book\_requests():

    requests = db.collection("book\_requests").get()

    request\_list = []

    for request in requests:

        request\_data = {

            "Name": request.get("Name"),

            "bookname": request.get("bookname"),

            "assign\_date": request.get("assign\_date"),

            "return\_date": request.get("return\_date")

        }

        request\_list.append(request\_data)

    return request\_list

def make\_book\_request(username,book\_name,assign\_date,return\_date):

    if not username:

        st.warning("PLease Log in first")

        return

    data={

    "Name": username ,

    "bookname":book\_name,

    "assign\_date": assign\_date.isoformat(),

    "return\_date": return\_date.isoformat()

    }

    db.collection("book\_requests").add(data)

    st.success("Request sent successfully!")

def user\_login(username, password):

    query = db.collection("users").where("student\_name", "==", username).limit(1).get()

    for doc in query:

        stored\_clg\_id = doc.get("clg\_id")

    if stored\_clg\_id :

        return stored\_clg\_id == password

    return False

def userbooks(coll):

    books = db.collection(coll).get()

    user\_books = []

    for doc in books:

        book\_data = {

            "bookname": doc.get("bookname"),

            "assign\_date": doc.get("assign\_date"),

            "return\_date": doc.get("return\_date")

        }

        user\_books.append(book\_data)

    return user\_books

def main():

    st.title("Library")

    selection = st.sidebar.radio("Navigation", ["User", "Admin"])

    if selection == "Admin":

        st.header("Admin Panel")

        password = st.text\_input("Enter Admin Password", type="password")

        if password == "admin1234":

            st.header("Add Books")

            Name = st.text\_input("Name")

            Author = st.text\_input("Author")

            Genre = st.text\_input("Genre")

            Quantity = st.number\_input("Quantity", min\_value=0)

            if st.button("Submit"):

                data= {

                "Name": Name,

                "Author":Author,

                "Genre": Genre ,

                "Quantity": Quantity

                }

                db.collection("books").add(data)

                st.success("Book added successfully!")

            st.header("Book Requests")

            requests = fetch\_book\_requests()

            if requests:

                for request in requests:

                    st.write(f"Name: {request['Name']}")

                    st.write(f"Book Name: {request['bookname']}")

                    st.write(f"Assign Date: {request['assign\_date']}")

                    st.write(f"Return Date: {request['return\_date']}")

                    if st.button(f"Assign Book {request['bookname']}"):

                        data={

                          "bookname": request['bookname'],

                          "assign\_date": request['assign\_date'],

                          "return\_date": request['return\_date']

                        }

                        table=request['Name']+"\_books"

                        db.collection(table).add(data)

                        query = db.collection("books").where("Name", "==",request['bookname'] ).limit(1).get()

                        for doc in query:

                            current\_quantity = doc.get("Quantity")

                            new\_quantity = current\_quantity - 1

                            doc\_ref = db.collection("books").document(doc.id)

                            doc\_ref.update({"Quantity": new\_quantity})

                        request\_query = db.collection("book\_requests").where("bookname", "==", request['bookname']).limit(1).get()

                        for request\_doc in request\_query:

                            request\_doc\_ref = db.collection("book\_requests").document(request\_doc.id)

                            request\_doc\_ref.delete()

                        st.success("Book Assigned Sucessfully")

            else:

                st.write("No book requests found.")

    elif selection == "User":

        st.header("User Login")

        username = st.text\_input("Username")

        password = st.text\_input("Password", type="password")

        if st.button("Login"):

            if user\_login(username, password):

                st.success("Login successful!")

                coll= username+"\_books"

                user\_books=userbooks(coll)

                st.title("Assigned books")

                if user\_books:

                    df2 = pd.DataFrame(user\_books)

                    st.dataframe(df2)

                else:

                    st.write("No user books found.")

*# username= "Vedant"*

        st.title("Available Books")

        books = fetch\_books()

        if books:

            st.write("List of Books:")

            df = pd.DataFrame(books)

            st.dataframe(df)

        else:

            st.write("No books found.")

        st.header("Make Book Assignment Request")

        book\_name = st.text\_input("Book Name")

        assign\_date = st.date\_input("Assign Date")

        return\_date = st.date\_input("Return Date")

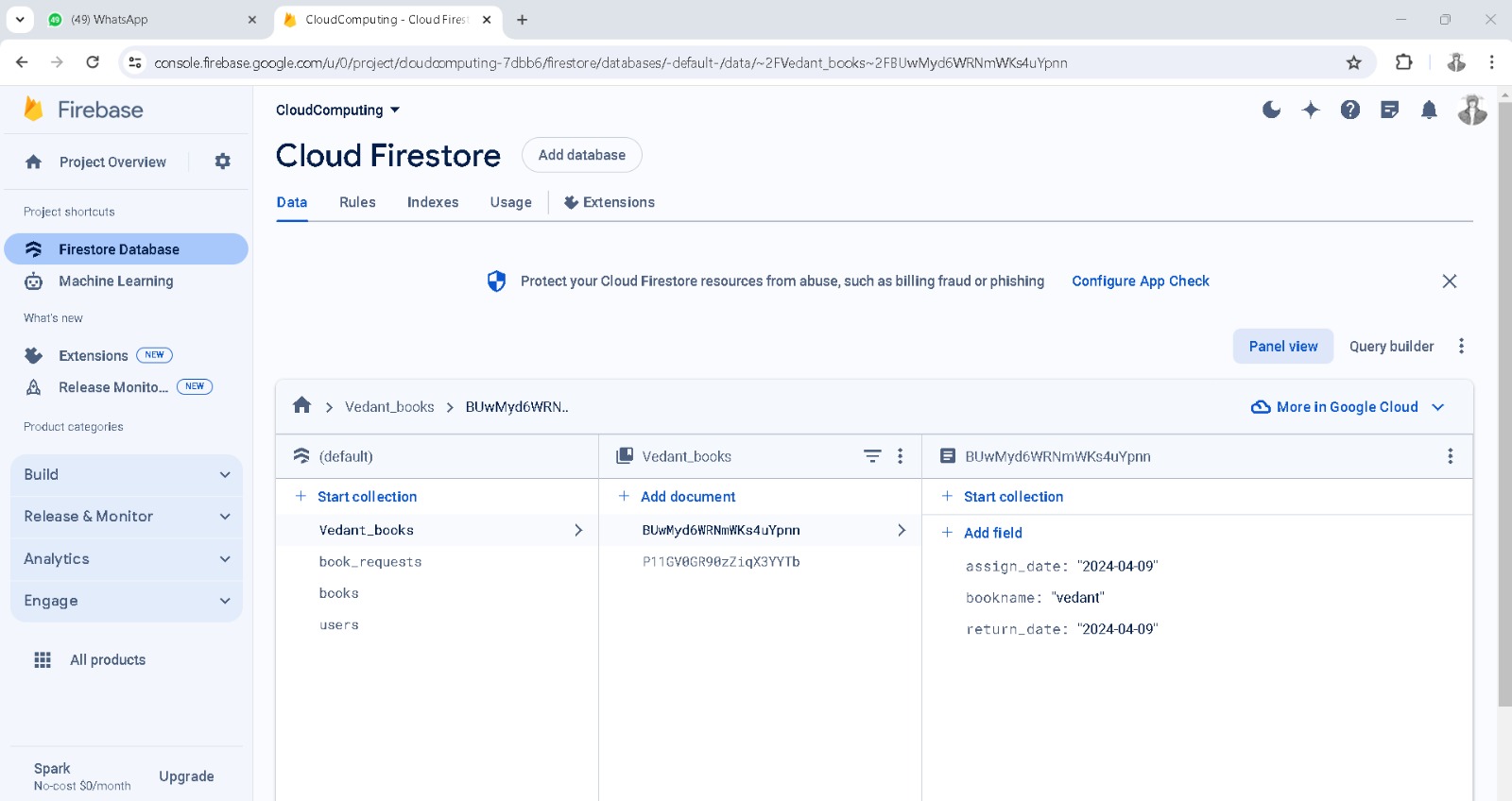
        if st.button("Send Request"):

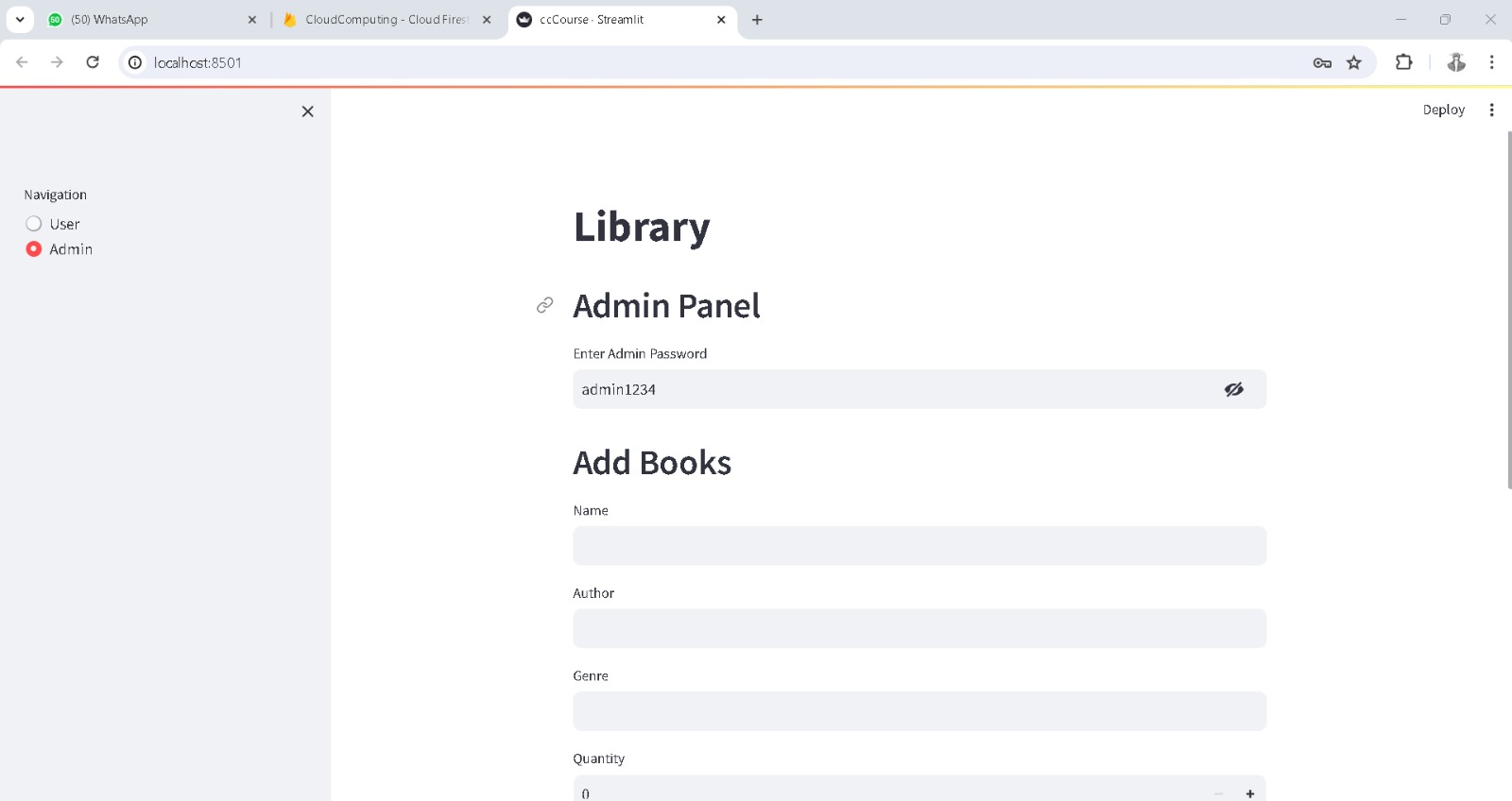
            make\_book\_request(username, book\_name, assign\_date, return\_date)

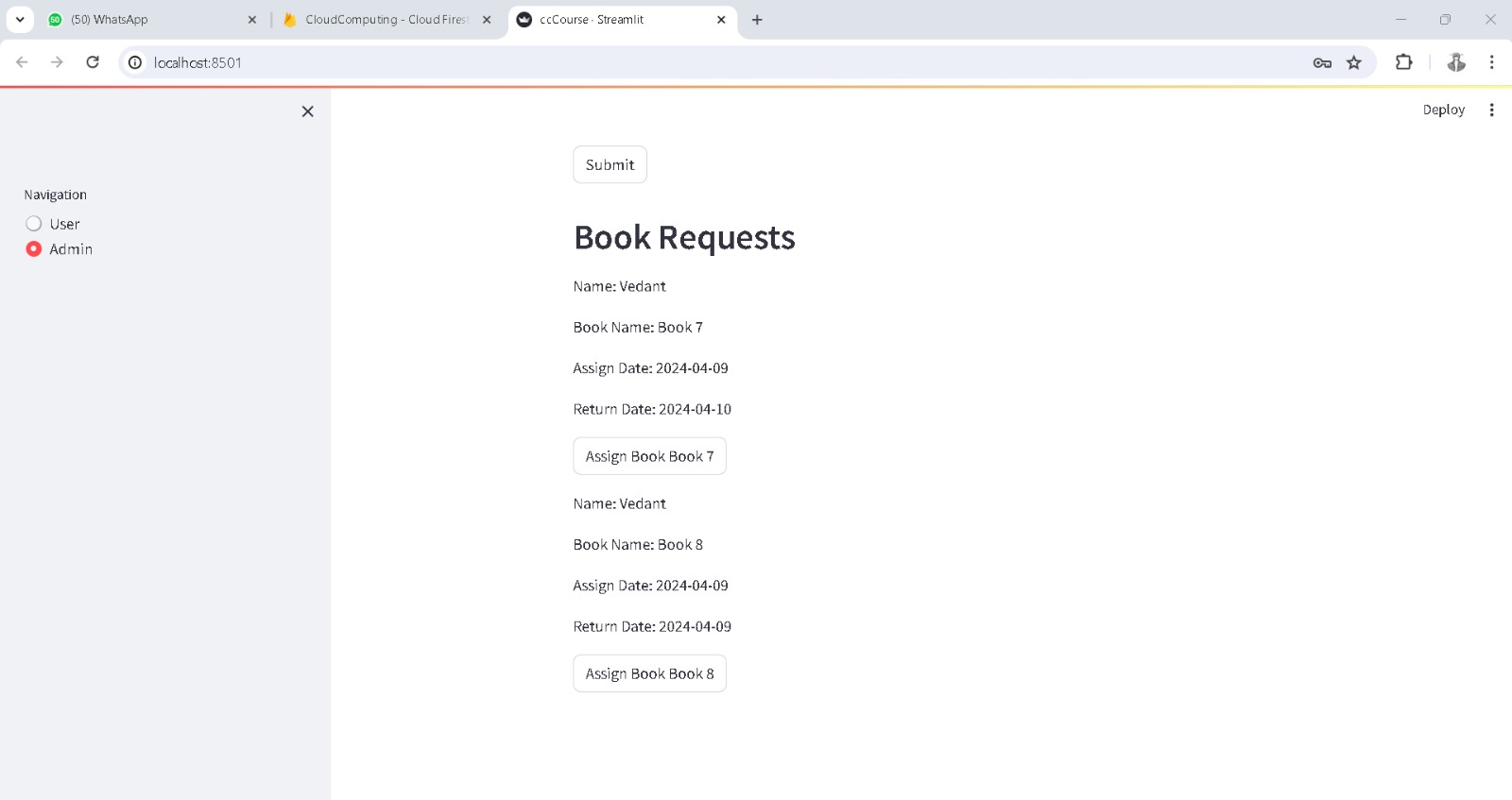
if \_\_name\_\_ == "\_\_main\_\_":

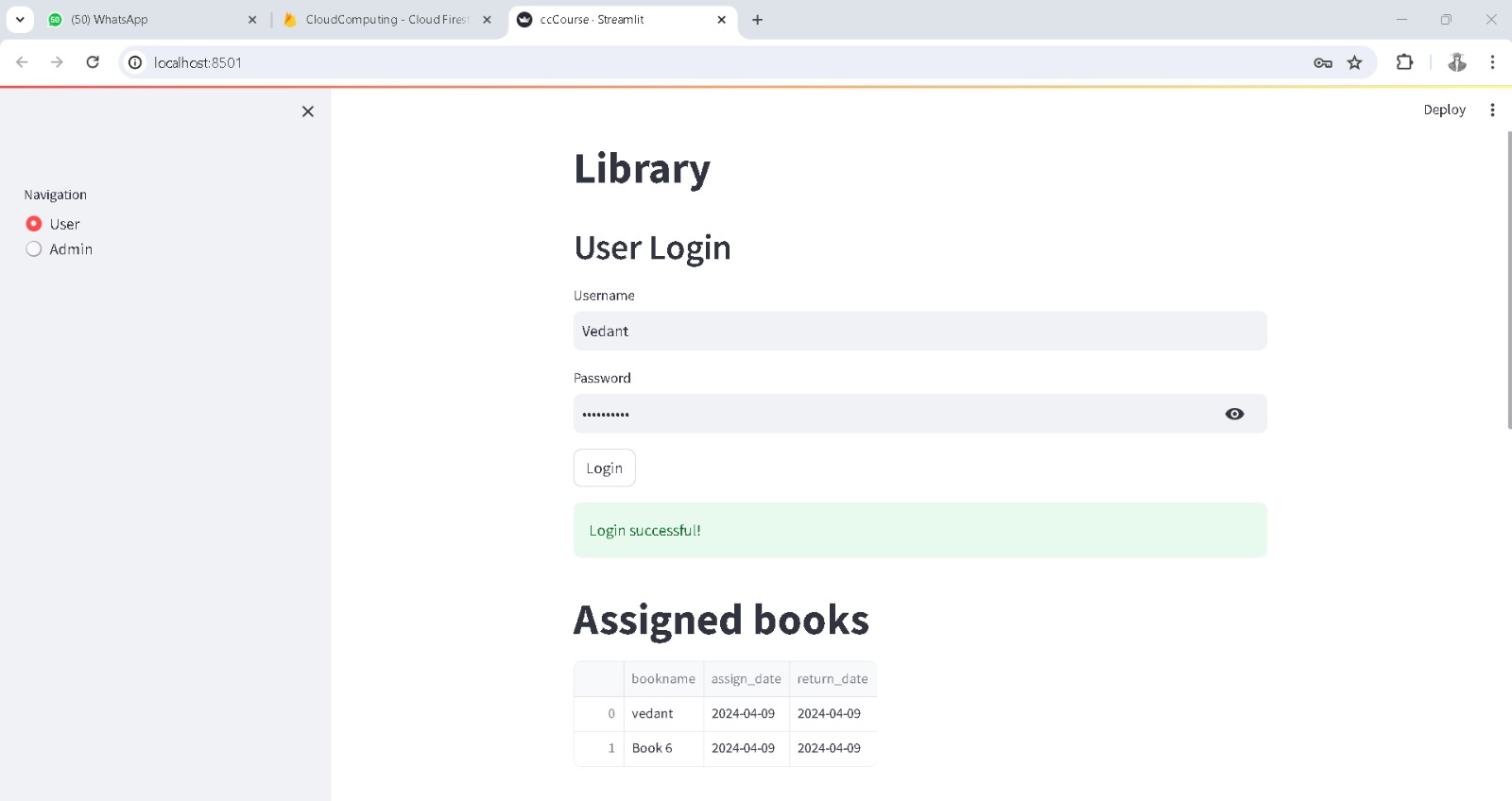
    main()

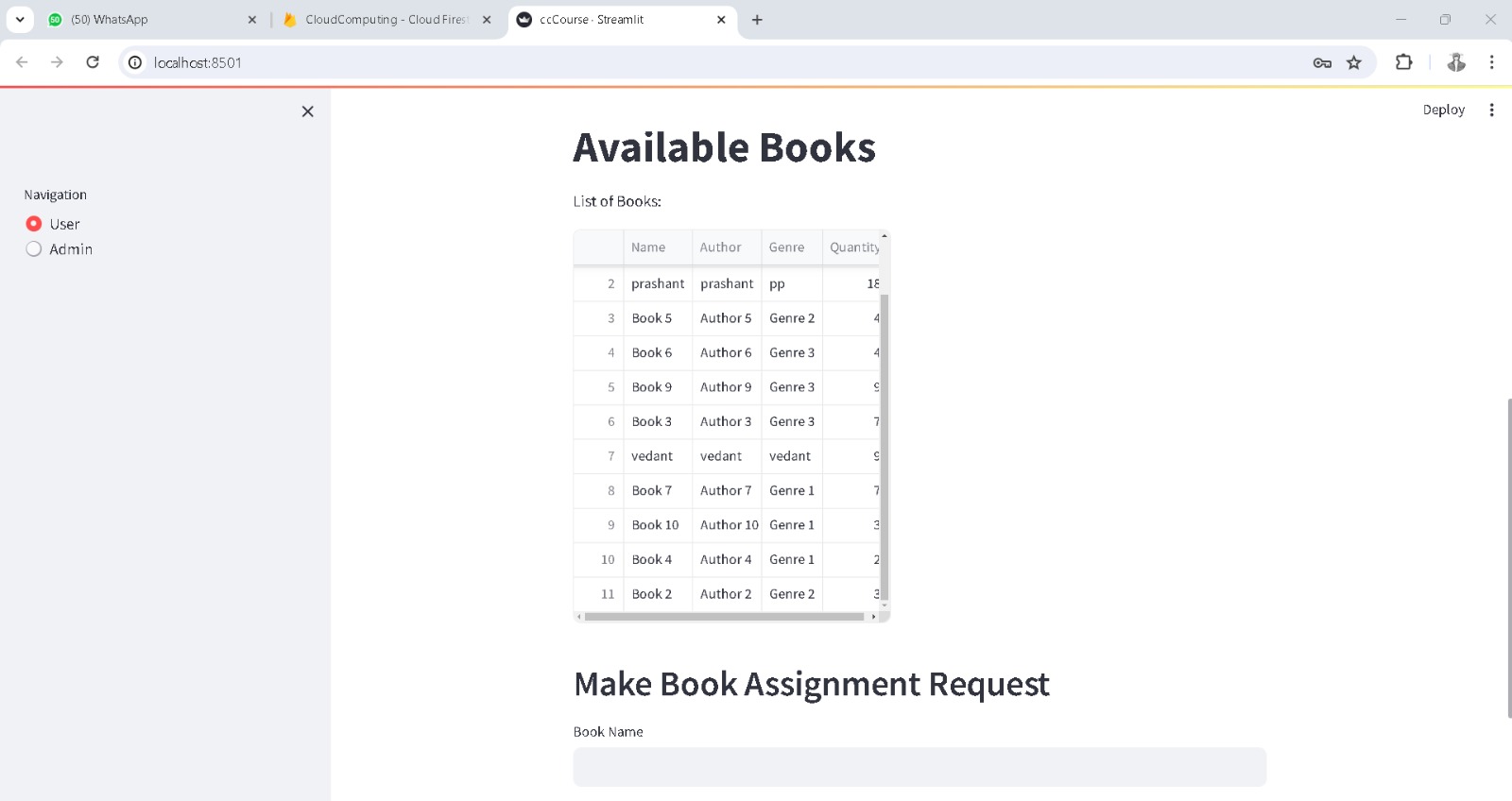
## Chapter 5: Result

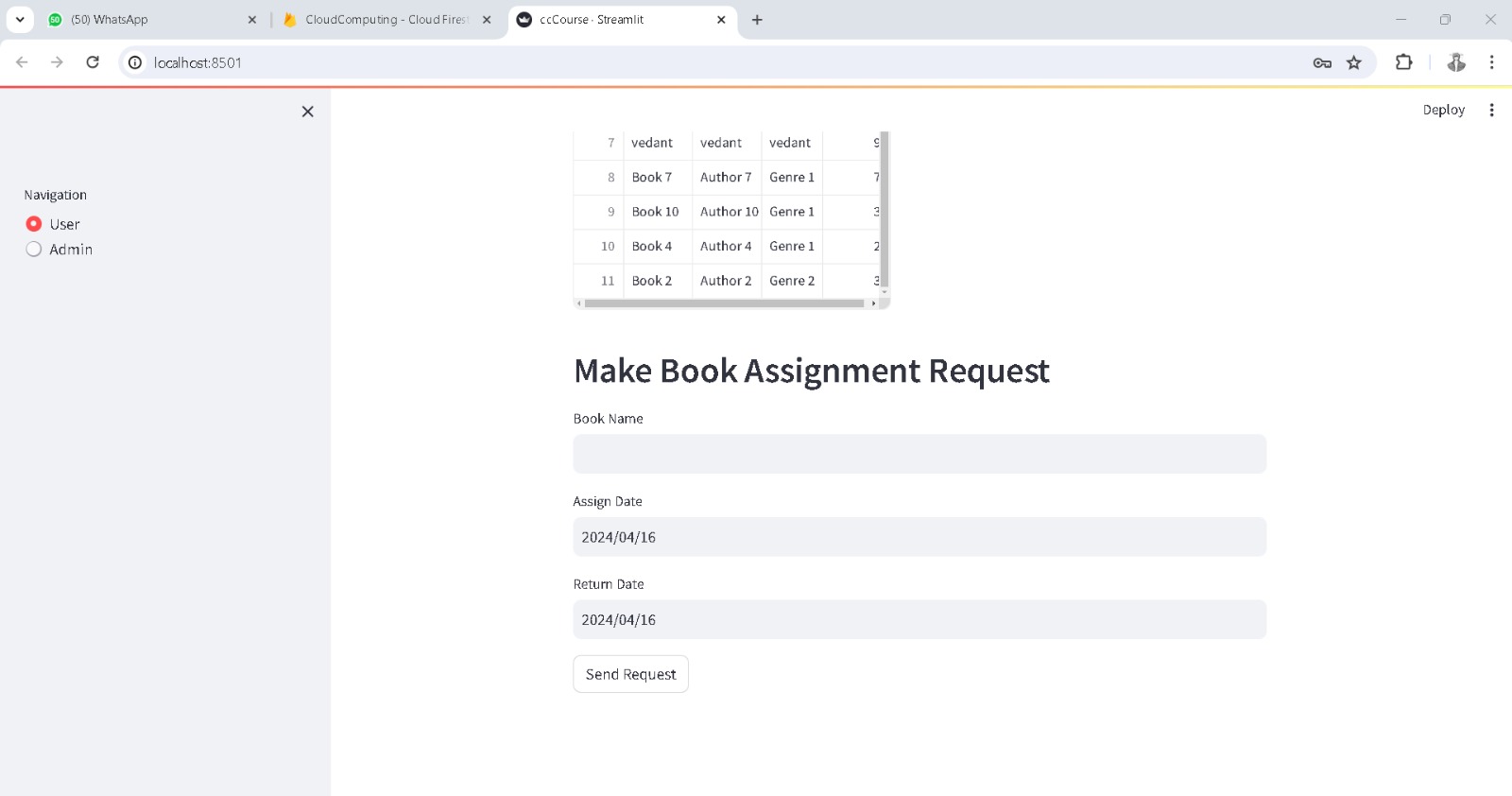












## Chapter 6: Conclusion :

The library management system, crafted through the amalgamation of Python, Firebase Firestore, and Streamlit, represents a sophisticated solution tailored to streamline the intricate processes of library resource management. Seamlessly blending a robust backend logic with an intuitive frontend interface, the system caters to the diverse needs of both administrators and users alike. Administrators are empowered with a user-friendly dashboard that simplifies tasks such as adding new books to the database and managing book requests with utmost efficiency. This includes assigning books to users and maintaining accurate inventory records, ensuring optimal resource allocation. Concurrently, users are presented with a seamless browsing experience, allowing them to effortlessly peruse the catalog of available books, submit assignment requests, and effortlessly track their currently borrowed items. The system's incorporation of Firebase Firestore serves as the backbone, guaranteeing secure and reliable data storage and retrieval, critical for maintaining the integrity of library records. Furthermore, the utilization of Streamlit's interactive components elevates user engagement to new heights, enhancing the overall user experience. With its scalability and adaptability, the system is poised to accommodate future enhancements and customizations, ensuring it remains at the forefront of modern library management practices. In essence, the library management system stands as a testament to technological innovation, optimizing operations, maximizing resource utilization, and ultimately fostering a more efficient and user-centric library environment.

### **REFERENCE**

1. <https://www.google.com/search?q=generate+image+for+system+design+for+warehouse+management&sca_esv=575801097&rlz=1C1RXQR_enIN1077IN1077&tbm=isch&sxsrf=AM9HkKlV_d1sv8GLm2fxvZ1-ue1CS7i87A:1698072501087&source=lnms&sa=X&ved=2ahUKEwi1quuBtYyCAxXQeN4KHRk5BWcQ_AUoAXoECAIQAw&biw=1536&bih=730&dpr=1.25#imgrc=DjO_JVbJvb9bYM>
2. <https://online.visual-paradigm.com/diagrams/templates/entity-relationship-diagram/er-diagram-inventory-management-system/>
3. <https://chat.openai.com/c/2566ea32-83ef-4ae3-b3f6-8055f9a0fad1>