DBMS MINI PROJECT

University Course Allocation System using Firebase & Flask

Aim

To develop a web-based University Course Allocation System for Rajalakshmi Institute of Technology, allowing staff and students to manage course assignments, registrations, and allocations efficiently using Firebase, Python (Flask), HTML, CSS, and JavaScript.

Algorithm

Step 1: Setup Firebase

1. Create a Firebase project from Firebase Console.
2. Enable Authentication and Cloud Firestore.
3. Set up Firestore Rules for security.

Step 2: Setup Flask Backend

1. Install dependencies:

   ```
   pip install flask firebase-admin
   ```

2. Connect Firebase with Flask (firebase_config.py).
3. Create API endpoints for handling course and staff management.

Step 3: Build Frontend with HTML, CSS, and JavaScript

1. Create index.html for user interface.
2. Implement Firebase Authentication for user registration.
3. Allow course addition and staff allocation using Firestore.

Step 4: Integrate Flask API with Frontend

1. Use AJAX (fetch API) to call Flask backend from JavaScript.
2. Display data dynamically from Firestore.

Step 5: Test & Deploy

1. Run Flask backend using:

   ```
   python app.py
   ```

2. Open the HTML page in a browser.
3. Register users, add courses, allocate staff, and verify Firestore updates.

Procedure & Code Implementation

FireBase Setup:

Step 1: Create a Firebase Project

1. Go to Firebase Console: https://console.firebase.google.com/
2. Click "Add Project" and give it a suitable name (e.g., University-Course-Allocation).
3. Disable Google Analytics (optional) and click Create Project.
4. Once created, click "Continue" to enter the Firebase Dashboard.

Step 2: Enable Firestore Database

1. On the left sidebar, go to "Build" → "Firestore Database".
2. Click Create Database.
3. Choose "Start in Test Mode" (for development) and click Next.
4. Select your preferred location (e.g., "asia-south1 (Mumbai)" for India or "us-central1" for the USA).

5. Click Enable and wait for the database setup.

Step 3: Set Firestore Security Rules

1. In Firestore Database, go to the Rules tab.
2. Replace the existing rules with:

```
rules_version = '2';
service cloud.firestore {
  match /databases/{database}/documents {
    match /courses/{document=**} {
      allow read, write: if true;
    }
    match /staff/{document=**} {
      allow read, write: if true;
    }
    match /students/{document=**} {
      allow read, write: if true;
    }
  }
}
```

3. Click Publish.

Step 4: Get Firebase Configuration

1. On the left menu, go to Project Settings (⚙ icon on top left).
2. Scroll down to "Your apps" and click "Add app".
3. Select Web App (</>) and enter an app name (e.g., RIT_Course_Allocation).
4. Click Register App (no need for Firebase Hosting).
5. Copy the Firebase SDK Configuration .

```
const firebaseConfig = {
    apiKey: "YOUR_API_KEY",
    authDomain: "YOUR_PROJECT_ID.firebaseapp.com",
    projectId: "YOUR_PROJECT_ID",
```

```
    storageBucket: "YOUR_PROJECT_ID.appspot.com",
    messagingSenderId: "YOUR_MESSAGING_SENDER_ID",
    appId: "YOUR_APP_ID"
};
```

6. Paste this inside your HTML file where Firebase is initialized.

1.Firebase Configuration

Create firebase_config.py

This file connects Flask with Firebase.

```
import firebase_admin
from firebase_admin import credentials, firestore, auth
import os

# Use an environment variable or relative path for better
portability
firebase_key_path = os.getenv("FIREBASE_KEY_PATH",
"universitycall-firebase-adminsdk-fbsvc-349cbfbdae.json")

try:
    cred = credentials.Certificate(firebase_key_path)
    firebase_admin.initialize_app(cred)
    db = firestore.client()
    print("Firebase initialized successfully!")
except Exception as e:
    print(f"Error initializing Firebase: {e}")
```

2.Flask Backend (app.py)

Install Dependencies

```
pip install flask firebase-admin
```

Flask Code:

```
from flask import Flask, render_template, request, jsonify
import os
from firebase_config import db, auth



app = Flask(__name__)

# Homepage
@app.route('/')
def index():
    return render_template('index.html')

# Add Course
@app.route('/add_course', methods=['POST'])
def add_course():
    try:
        data = request.json
        db.collection('courses').add(data)
        return jsonify({"message": "Course added successfully!"}),
201
    except Exception as e:
        return jsonify({"error": str(e)}), 500

# Fetch Courses
@app.route('/get_courses', methods=['GET'])
def get_courses():
    try:
        courses = db.collection('courses').stream()
        return jsonify([course.to_dict() for course in courses]),
200
    except Exception as e:
        return jsonify({"error": str(e)}), 500
```

```
# Add Staff
@app.route('/add_staff', methods=['POST'])
def add_staff():
    try:
        data = request.json
        db.collection('staff').add(data)
        return jsonify({"message": "Staff added successfully!"}),
201
    except Exception as e:
        return jsonify({"error": str(e)}), 500

# Fetch Staff
@app.route('/get_staff', methods=['GET'])
def get_staff():
    try:
        staff = db.collection('staff').stream()
        return jsonify([s.to_dict() for s in staff]), 200
    except Exception as e:
        return jsonify({"error": str(e)}), 500

if __name__ == '__main__':
    app.run(debug=True)
```

3.Frontend (HTML + CSS + JavaScript)

Create index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>RIT Course Allocation</title>
    <style>
        body {
            font-family: 'Arial', sans-serif;
```

```
        margin: 0;
        padding: 0;
        background-color: #f4f4f4;
        text-align: center;
    }
    .container {
        width: 90%;
        max-width: 600px;
        margin: 20px auto;
        background: white;
        padding: 20px;
        border-radius: 10px;
        box-shadow: 0px 0px 10px rgba(0, 0, 0, 0.1);
    }
    h1 {
        color: #2c3e50;
    }
    input, select, button {
        width: 100%;
        padding: 10px;
        margin: 10px 0;
        border: 1px solid #ddd;
        border-radius: 5px;
        font-size: 16px;
    }
    button {
        background-color: #2980b9;
        color: white;
        border: none;
        cursor: pointer;
        transition: 0.3s;
    }
    button:hover {
        background-color: #1c5985;
    }
    ul {
```

```
            list-style: none;
            padding: 0;
        }
        li {
            background: #ecf0f1;
            margin: 5px 0;
            padding: 10px;
            border-radius: 5px;
        }
    </style>
    <script type="module">
        import { initializeApp } from
"https://www.gstatic.com/firebasejs/10.7.1/firebase-app.js";
        import { getAuth, createUserWithEmailAndPassword }
from "https://www.gstatic.com/firebasejs/10.7.1/firebase-
auth.js";
        import { getFirestore, collection, addDoc, getDocs } from
"https://www.gstatic.com/firebasejs/10.7.1/firebase-
firestore.js";

        // Firebase Configuration
        const firebaseConfig = {
            apiKey: "AIzaSyCK7UVW-JSJ1f4d-
jmbZu0fAwi4MMdsM2Y",
            authDomain: "universitycall.firebaseapp.com",
            projectId: "universitycall",
            storageBucket: "universitycall.firebasestorage.app",
            messagingSenderId: "654742801296",
            appId: "1:654742801296:web:f4f31e6b7d14d237684115",
            measurementId: "G-3QFPMX19C0"
};

        // Initialize Firebase
        const app = initializeApp(firebaseConfig);
        const auth = getAuth(app);
        const db = getFirestore(app);
```

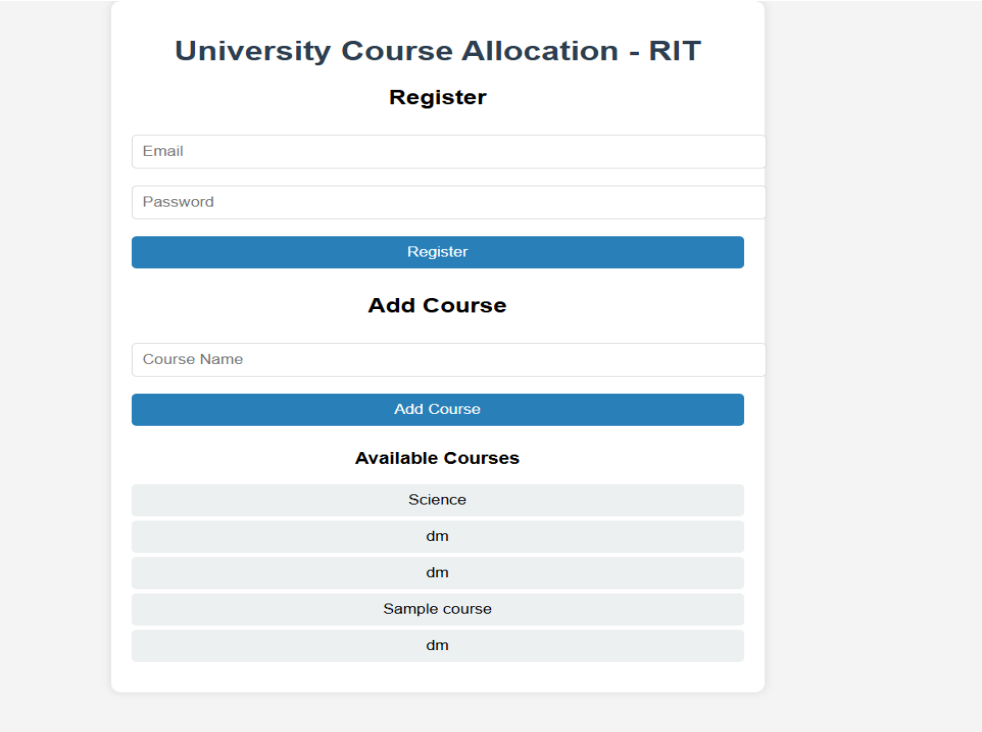
```
    // User Registration
    window.register = async function () {
        let email = document.getElementById("email").value;
        let password =
document.getElementById("password").value;
        try {
            await createUserWithEmailAndPassword(auth, email,
password);
            alert("User Registered Successfully!");
        } catch (error) {
            alert("Registration Failed: " + error.message);
        }
    };

    // Add Course
    window.addCourse = async function () {
        let courseName =
document.getElementById("courseName").value;
        if (courseName.trim() === "") {
            alert("Please enter a course name");
            return;
        }
        try {
            await addDoc(collection(db, "courses"), { name:
courseName });
            alert("Course Added Successfully!");
            fetchCourses();
        } catch (error) {
            alert("Error Adding Course: " + error.message);
        }
    };

    // Fetch Courses
    window.fetchCourses = async function () {
```

```
        const courseList =
document.getElementById("courseList");
        courseList.innerHTML = "";
        try {
            const querySnapshot = await getDocs(collection(db,
"courses"));
            querySnapshot.forEach((doc) => {
                let li = document.createElement("li");
                li.textContent = doc.data().name;
                courseList.appendChild(li);
            });
        } catch (error) {
            console.error("Error fetching courses: ", error);
        }
    };

    // Fetch courses on page load
    window.onload = fetchCourses;
  </script>
</head>
<body>
  <div class="container">
    <h1>University Course Allocation - RIT</h1>
    <h2>Register</h2>
    <input type="email" id="email" placeholder="Email">
    <input type="password" id="password"
placeholder="Password">
    <button onclick="register()">Register</button>

    <h2>Add Course</h2>
    <input type="text" id="courseName" placeholder="Course
Name">
    <button onclick="addCourse()">Add Course</button>
    <h3>Available Courses</h3>
    <ul id="courseList"></ul>
  </div>
```

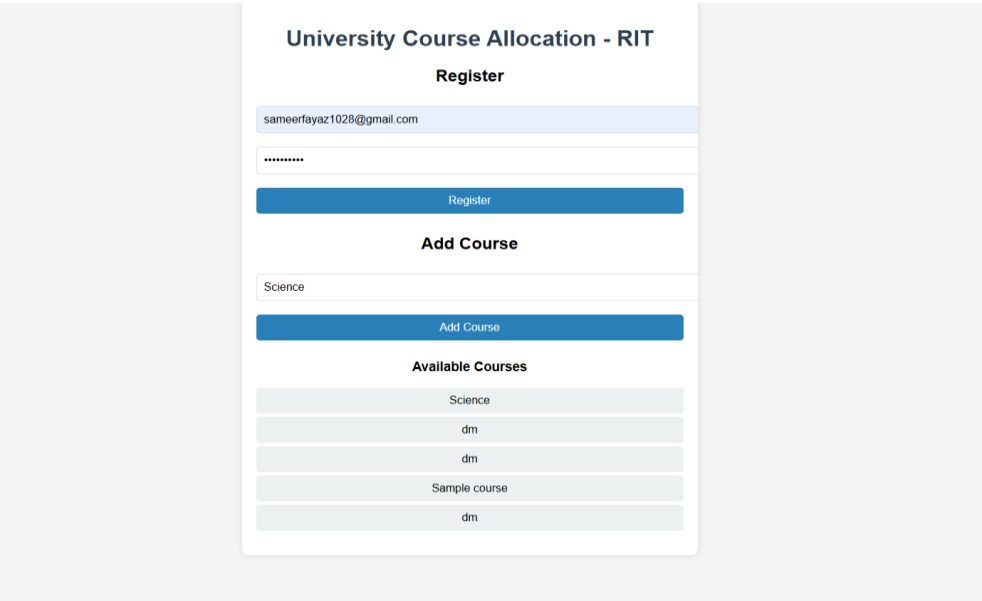
```
</body>
</html>
```

OUTPUT:-

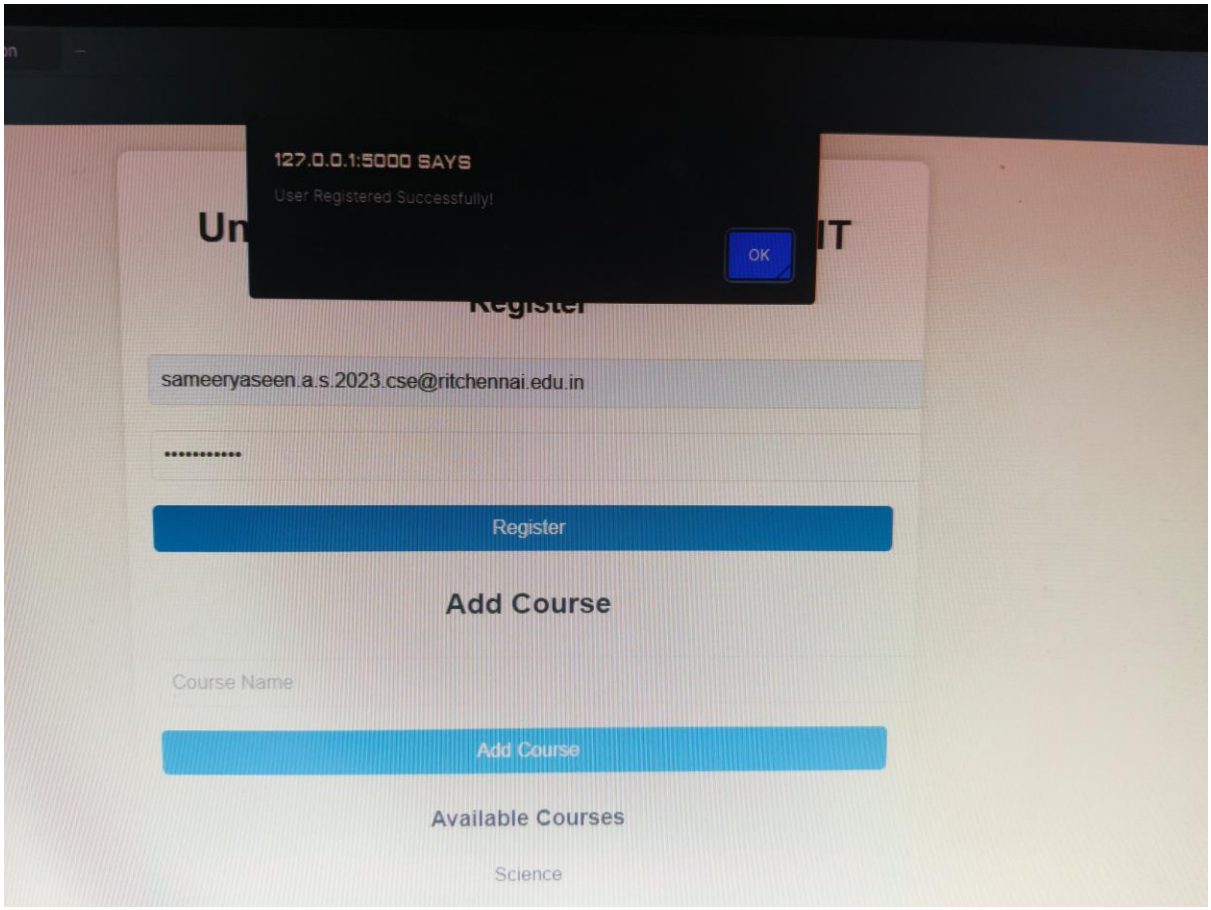
Register users successfully with Firebase Authentication.



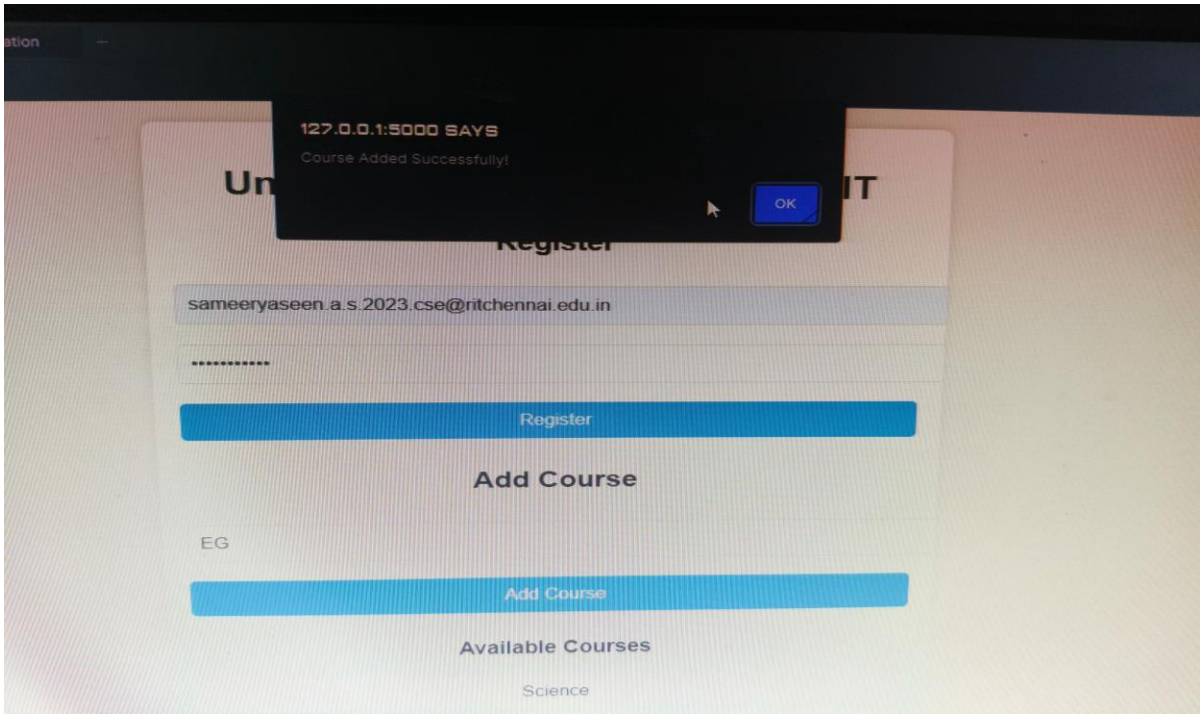
Add courses dynamically and store them in Firestore.

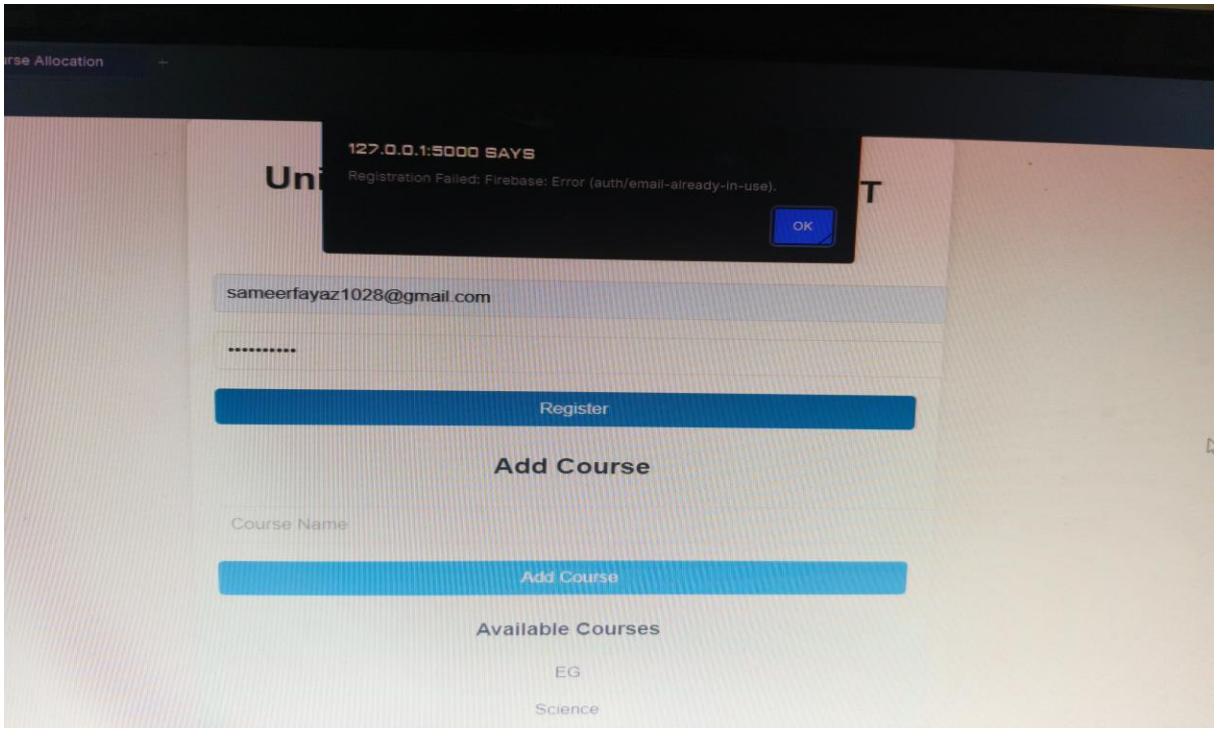


Fetch and display courses on the webpage.



Courses persist in Firestore even after page refresh.





Applications of the Project

The University Course Allocation System using Firebase, Flask, and JavaScript has several applications in academic institutions:

1. Automated Course Allocation
   - Helps universities assign courses to students and faculty efficiently.
2. Student Course Registration
   - Allows students to register for courses online with real-time availability updates.
3. Faculty Course Assignment
   - Assigns courses to faculty members based on preferences and workload.
4. Database Management
   - Stores information about courses, students, and faculty securely.
5. Real-time Updates
   - Changes in course availability or faculty assignments reflect instantly.

6. Reduces Administrative Work
    - Minimizes manual efforts in course allocation and registration.

Advantages of the Project

The system provides several benefits:

1. Time-Saving & Efficient
    - Eliminates the need for manual course assignments, reducing administrative workload.
2. Scalability
    - Can handle a growing number of students and courses without performance issues.
3. Real-Time Updates
    - Firebase ensures instant updates when courses or faculty assignments change.
4. Data Security
    - Firestore Database provides a secure environment with authentication controls.
5. Easy Accessibility
    - Can be accessed from anywhere, eliminating the need for in-person registration.
6. User-Friendly Interface
    - Simple and intuitive design for students and faculty to interact with the system.
7. Paperless Management
    - Reduces paperwork and supports eco-friendly digital operations.
8. Error Reduction
    - Prevents errors caused by manual course allocation and conflicts in scheduling.

Disadvantages of the Project

Despite its advantages, the system has some limitations:

1. Requires Internet Connectivity
    - As Firebase is cloud-based, users need a stable internet connection.
2. Dependence on Google Firebase
    - If Firebase services experience downtime, the system may be affected.
3. Initial Setup Complexity
    - Setting up authentication, Firestore rules, and API configurations requires technical knowledge.
4. Limited Customization
    - Firebase has restrictions compared to fully self-hosted databases like MySQL or PostgreSQL.
5. Security Concerns
    - Improper Firestore security rules can lead to data breaches.
6. Resource Constraints
    - Free Firebase tier has limits on database reads, writes, and storage, which may require paid upgrades for larger institutions.

RESULT:-

This project provides an efficient course allocation system for universities using Firebase & Flask. It ensures real-time updates, user authentication, and dynamic course management.