MENTOR CONNECT

ABSTRACT

Through this effort, the gaps in several areas of constraint between mentors and students are bridged. For mentors and students alike, this system offers an online mentoring system. There is a significant communication gap between mentors and students in conventional systems. The mentor and students must be physically present in the same location under the conventional arrangement. It takes a lot of time, has a lengthy documentation procedure, and allows for little engagement. To overcome all this problems we are developing a web based application providing necessary information management for mentors and students. Students can book to the mentors according to their area of interests to which the mentors can be helpful to improve the personal as well as academics of the students, the students and mentor can communicate through messaging on the website, and can book slot for video call, all the information will be managed digitally.

CHAPTER 1

INTRODUCTION

1.1 OBJECTIVE

Both the institute's faculty and students will benefit greatly from our system's excellent mentoring program. For the kind of students who find it difficult to ask questions or get their doubts answered by faculty members in a one-on-one setting, this will be quite beneficial. Students can use this platform to post their queries and doubts. Message from any location at any time, and the faculty member who has logged into the system can answer any inquiries or concerns. Additionally, since all of the information will be digitally kept and secure, it will aid in the better administration of faculty and student data. The Mentor-Student Booking Application is a platform designed to facilitate seamless interactions between mentors and students for academic or professional guidance. Through this application, mentors can register and create their profiles by providing essential information, including their skills, expertise, and availability. The system allows students to browse through the mentor profiles and send booking requests for a one-on-one session based on the mentor's available time slots. Once a student sends a booking request, mentors have the ability to accept or reject the request based on their availability and preferences. After the booking request is confirmed, students can view the status of their booking, which ensures transparency and keeps both parties informed. In addition, the platform supports real-time communication, allowing students to chat with mentors during their session or at any other time based on the booked calendar slots. The application is designed to enhance the mentoring process, promoting effective communication, collaboration, and scheduling. It helps students easily access guidance from experts, while mentors can manage their sessions efficiently and ensure a structured learning environment. This platform streamlines the mentorship experience, making it more convenient and accessible for both students and mentors.

CHAPTER 2

SYSTEM ANALYSIS

2.1 EXISTING SYSTEM

The existing manual system for mentor-student interactions typically involves students directly contacting mentors through informal channels such as email, phone calls, or in-person requests. This process lacks a structured approach for managing appointments, making it difficult to track available time slots, confirm bookings, or ensure that both parties are on the same page regarding the schedule. In such a system, mentors may not have a clear way to manage multiple student requests, and students often face challenges in getting timely responses or updates regarding their booking status. The absence of an integrated platform means that mentors and students may experience miscommunication, overlapping schedules, or delayed responses. Additionally, mentors may struggle to maintain accurate records of their appointments, and students might find it challenging to follow up on the status of their booking. The manual system's limitations significantly hinder the effectiveness, accessibility, and organization of the mentor-student relationship.

DISADVANTAGES

- Lack of centralized scheduling leads to confusion and errors, as students and mentors must rely on informal channels such as emails or phone calls to coordinate session times.
- The chances of double-booking or missed appointments due to inconsistent tracking of time slots.
- There is often no clear visibility into booking status, which can cause frustration and uncertainty about whether their request has been accepted, rejected, or is pending approval.
- Communication is also a challenge in the manual system as there is no dedicated platform for direct messaging between mentors and students.

2.2 PROPOSED SYSTEM

The proposed Mentor Connect application is designed to streamline the mentorship process by providing a centralized platform for managing bookings, communication, and scheduling. In this system, mentors can register and create detailed profiles by adding their skills and expertise, allowing students to easily identify the right mentor based on their needs. Students can browse through mentor profiles and send booking requests for specific time slots that align with the mentor's availability. Once a request is sent, mentors have the option to accept or reject the booking based on their schedule and preferences, ensuring better control over their time. After the mentor responds, students can view the status of their booking, whether it is confirmed or pending, which adds transparency and reduces confusion. It ensures that both mentors and students have a clear and efficient way to manage their sessions, leading to a more organized, accessible, and productive mentoring experience for all parties involved.

ADVANTAGES

- One of the primary benefits is the centralized scheduling system, which ensures efficient management of booking requests, reducing the risk of scheduling conflicts and missed appointments.
- Students can easily send booking requests to mentors, and mentors can accept or reject requests based on their availability, allowing for better control over appointments.
- Significant advantage is the direct messaging feature, which allows students and mentors to chat with each other based on their scheduled sessions.
- The proposed system improves efficiency, organizes mentor-student interactions, and enhances the overall learning experience.

2.3 FEASIBILITY STUDY

The survey is now developed into a more thorough feasibility assessment, contingent on the findings of the preliminary inquiry. A "FEASIBILITY STUDY" evaluates a system idea based on its viability, organizational impact, capacity to satisfy demands, and efficient use of available resources. It centres on these important queries.

- What observable needs does the user have, and how is a potential system able to address them?
- Which tools are available for the particular candidate system?
- What effects on the Organization is the candidate system expected to have?
- Whether it is worth to solve the problem?

Events and notifications have to be taken into account throughout this project's feasibility investigation. This is accomplished through research and idea generation for a new system.

2.3.1 Economical Feasibility

For most systems, the "Bottom Line" factor is economic justification. A wide range of issues, including cost-benefit analysis, are included in economic justification. This involves weighing the advantages and disadvantages of the potential system, and if it aligns with the organization's primary goal of generating profits, the project moves on to the analysis and design stage. The following estimates are based on the verification of the financial and economic inquiries made during the initial investigation:

- The price of carrying out a thorough system analysis.
- The price of the software and hardware required for the type of application under consideration.
- The advantages in the form of financial savings.
- Since the suggested system would provide up-to-date information, performance will improve, which might lead to higher earnings.
- This feasibility study determines if developing a system with events and alert monitoring is feasible without requiring manual labor. This is economically doable, provided that it is planned judicially. The project's cost is determined by the quantity of man hours needed.

2.3.2 Technical Feasibility

An examination of the resource availability that might impact the development of a workable system. This assessment establishes the availability of the technology required for the suggested system..

- Is it possible to complete the project's work with the current hardware, software, and personnel?
- Should the system be created, is it upgradeable?
- What can be developed if new technology is required?

2.3.3 Operational Feasibility

It mostly has to do with political and human organisation. The things to think about are:

- What modifications will the system bring about?
- Which organizational structures are under jeopardy?
- What fresh abilities will be needed? Do the current employees possess these abilities? If not, will they be able to learn in due course?

The system is operationally feasible as it very easy for the End users to operate it. It only needs basic information about Windows platform Because the end users can run the system with great ease, it is operationally practical. It merely requires the bare minimum of Windows platform knowledge.

CHAPTER 3

SYSTEM SPECIFICATION

3.1 HARDWARE REQUIREMENTS

• Processor	: Dual core processor 2.6.0 GHZ
• RAM	: 4GB
• Hard disk	: 320 GB
• Compact Disk	: 650 Mb
• Keyboard	: Standard keyboard
• Monitor	: 15 inch color monitor

3.2 SOFTWARE REQUIREMENTS

• Operating system	: Windows OS
• Front End	: HTML, CSS, JavaScript
• Back End	: Python
• Database	: MySQL SERVER
• IDLE	: Python 2.7 IDLE

CHAPTER 4

SOFTWARE DESCRIPTION

4.1 Back End: Python

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace. It provides constructs that enable clear programming on both small and large scales. In July 2018, Van Rossum stepped down as the leader in the language community. Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library. Python interpreters are available for many operating systems. CPython, the reference implementation of Python, is open source software and has a community-based development model, as do nearly all of Python's other implementations. Python and CPython are managed by the non-profit Python Software Foundation. Rather than having all of its functionality built into its core, Python was designed to be highly extensible. This compact modularity has made it particularly popular as a means of adding programmable interfaces to existing applications. Van Rossum's vision of a small core language with a large standard library and easily extensible interpreter stemmed from his frustrations with ABC, which espoused the opposite approach. While offering choice in coding methodology, the Python philosophy rejects exuberant syntax (such as that of Perl) in favour of a simpler, less-cluttered grammar. As Alex Martelli put it: "To describe something as 'clever' is not considered a compliment in the Python culture."Python's philosophy rejects the Perl "there is more than one way to do it" approach to language design in favour of "there should be one—and preferably only one—obvious way to do it".

Python's developers strive to avoid premature optimization, and reject patches to non-critical parts of CPython that would offer marginal increases in speed at the cost of clarity.[When speed is important, a Python programmer can move time-critical functions to extension modules written in languages such as C, or use PyPy, a just-in-time compiler. CPython is also available, which translates a Python script into C and makes direct C-level API calls into the Python interpreter. An important goal of Python's developers is keeping it fun to use. This is reflected in the language's name a tribute to the British comedy group Monty Python and in occasionally playful approaches to tutorials and reference materials, such as examples that refer to spam and eggs (from a famous Monty Python sketch) instead of the standard for and bar.

Python's initial development was spearheaded by Guido van Rossum in the late 1980s. Today, it is developed by the Python Software Foundation. Because Python is a multi paradigm language, Python programmers can accomplish their tasks using different styles of programming: object oriented, imperative, functional or reflective. Python can be used in Web development, numeric programming, game development, serial port access and more.

There are two attributes that make development time in Python faster than in other programming languages:

1. Python is an interpreted language, which precludes the need to compile code before executing a program because Python does the compilation in the background. Because Python is a high-level programming language, it abstracts many sophisticated details from the programming code. Python focuses so much on this abstraction that its code can be understood by most novice programmers.
2. Python code tends to be shorter than comparable codes. Although Python offers fast development times, it lags slightly in terms of execution time. Compared to fully compiling languages like C and C++, Python programs execute slower. Of course, with the processing speeds of computers these days, the speed differences are usually only observed in benchmarking tests, not in real-world operations. In most cases, Python is already included in Linux distributions and Mac OS X machines.

4.2 Database: MYSQL SERVER

A database is a structured collection of data. It may be anything from a simple shopping list to a picture gallery or the vast amounts of information in a corporate network. To add, access, and process data stored in a computer database, you need a database management system such as MySQL Server. Since computers are very good at handling large amounts of data, database management systems play a central role in computing, as standalone utilities, or as parts of other applications.

MySQL databases are relational.

A relational database stores data in separate tables rather than putting all the data in one big storeroom. The database structures are organized into physical files optimized for speed. The logical model, with objects such as databases, tables, views, rows, and columns, offers a flexible programming environment. You set up rules governing the relationships between different data fields, such as one-to-one, one-to-many, unique, required or optional, and "pointers" between different tables. The database enforces these rules, so that with a well-designed database, your application never sees inconsistent, duplicate, orphan, out-of-date, or missing data.

The SQL part of "MySQL" stands for "Structured Query Language". SQL is the most common standardized language used to access databases. Depending on your programming environment, you might enter SQL directly (for example, to generate reports), embed SQL statements into code written in another language, or use a language-specific API that hides the SQL syntax.

SQL is defined by the ANSI/ISO SQL Standard. The SQL standard has been evolving since 1986 and several versions exist. In this manual, "SQL-92" refers to the standard released in 1992, "SQL: 1999" refers to the standard released in 1999, and "SQL:2003" refers to the current version of the standard. We use the phrase "the SQL standard" to mean the current version of the SQL Standard at any time.

MySQL software is Open Source.

Open Source means that it is possible for anyone to use and modify the software. Anybody can download the MySQL software from the Internet and use it without paying anything. If you wish, you may study the source code and change it to suit your needs. The MySQL software uses the GPL (GNU General Public License), http://www.fsf.org/licenses/, to define what you may and may not do with the software in different situations. If you feel uncomfortable with the GPL or need to embed MySQL code into a commercial application, you can buy a commercially licensed version from us. See the MySQL Licensing Overview for more information (http://www.mysql.com/company/legal/licensing/).

The MySQL Database Server is very fast, reliable, scalable, and easy to use.

If that is what you are looking for, you should give it a try. MySQL Server can run comfortably on a desktop or laptop, alongside your other applications, web servers, and so on, requiring little or no attention. If you dedicate an entire machine to MySQL, you can adjust the settings to take advantage of all the memory, CPU power, and I/O capacity available. MySQL can also scale up to clusters of machines, networked together. MySQL Server was originally developed to handle large databases much faster than existing solutions and has been successfully used in highly demanding production environments for several years. Although under constant development, MySQL Server today offers a rich and useful set of functions. Its connectivity, speed, and security make MySQL Server highly suited for accessing databases on the Internet.

MySQL Server works in client/server or embedded systems.

The MySQL Database Software is a client/server system that consists of a multithreaded SQL server that supports different back ends, several different client programs and libraries, administrative tools, and a wide range of application programming interfaces (APIs).We also provide MySQL Server as an embedded multithreaded library that you can link into your application to get a smaller, faster, easier-to-manage standalone product.

A large amount of contributed MySQL software is available.

MySQL Server has a practical set of features developed in close cooperation with our users. It is very likely that your favourite application or language supports the MySQL Database Server. The official way to pronounce “MySQL” is “My Ess Que Ell” (not “my sequel”), but we do not mind if you pronounce it as “my sequel” or in some other localized way.

Connectivity

- Clients can connect to MySQL Server using several protocols:
- Clients can connect using TCP/IP sockets on any platform.
- On Windows systems, clients can connect using named pipes if the server is started with the named_pipe system variable enabled. Windows servers also support shared-memory connections if started with the shared_memory system variable enabled. Clients can connect through shared memory by using the --protocol=memory option.
- On Unix systems, clients can connect using Unix domain socket files.
- MySQL client programs can be written in many languages. A client library written in C is available for clients written in C or C++, or for any language that provides C bindings.

- APIs for C, C++, Eiffel, Java, Perl, PHP, Python, Ruby, and Tcl are available, enabling MySQL clients to be written in many languages. See Chapter 29, Connectors and APIs.
- The Connector/ODBC (MyODBC) interface provides MySQL support for client programs that use ODBC (Open Database Connectivity) connections. For example, you can use MS Access to connect to your MySQL server. Clients can be run on Windows or UNIX. Connector/ODBC source is available. All ODBC 2.5 functions are supported, as are many others. See MySQL Connector/ODBC Developer Guide.

CHAPTER 5

PROJECT DESCRIPTION

5.1 PROBLEM DEFINITION

The requirement for in-person sessions, the short interaction time, and the ineffective documentation procedures in traditional mentoring systems sometimes impede good communication between mentors and students. These difficulties lower student involvement, delay feedback, and produce gaps in instruction. Additionally, it could be difficult for students to locate mentors who share their needs for academic or personal growth. This lack of customisation and accessibility has an impact on the mentoring process's overall efficacy. As a result, a digital solution that promotes constant communication, simplifies mentor-mentee matching, and effectively handles all associated data is required. By offering a consolidated platform for scheduling, communication, and progress monitoring, a web-based mentoring system solves these problems and eventually promotes a more effective and convenient mentoring experience for mentors and students.

5.2 OVERVIEW OF THE PROJECT

The goal of this project is to create a web-based mentoring application that fills in the gaps in contact and communication that are frequently present in conventional mentoring systems. The requirement for in-person attendance, time restraints, and laborious paperwork procedures restrict mentor-student connection in traditional arrangements. By providing a digital platform where students may engage with mentors based on their academic and personal interests, this approach removes these obstacles. Regular and productive mentor-student contacts are made possible by the platform's integrated messaging tools, which enable smooth communication. It also offers organized information management, which enables mentors to monitor students' progress and provide customized advice. This technology improves accessibility, speed, and engagement by digitizing the mentoring process, creating a more encouraging and cohesive learning environment.

5.3 MODULES DESCRIPTION

MODULES

MENTOR

- Register
- Login
- Add Skill Details
- View Request
- Accept/ Reject
- Start Chat
- View Slot Request

STUDENTS

- Register
- Login
- View Skill Details
- Request to Mentor
- Start Chat
- Slot Booking

MODULE DESCRIPTION

MENTOR

- **Register**
  In this module allows new mentors to sign up on the platform by providing essential information such as name, email, area of expertise, and contact details. The registration form ensures that all required fields are validated before submission. Upon successful registration, the data is stored in the database and awaits admin verification.

- **Login**
  In this module allows registered mentors to securely access the system using their credentials, such as username and password.
- **Add Skill Details**
  This module allows mentors to input and update their specific skills and areas of expertise on the platform. Mentors can categorize their skills by subject, field, or interest to make it easier for students to find relevant guidance.
- **View Request**
  The View Chat Request and Accept/Reject module allows mentors to view incoming chat requests from students.
- **Accept/ Reject**
  Mentors can review the details of each request, including the student's query and area of interest. The mentor has the option to either accept or reject the request based on availability or relevance.
- **Start Chat**
  This module allows mentors to respond to messages sent by students within the chat interface. Mentors can read the student’s inquiry and send a personalized response. This feature enables real-time, two-way communication, promoting interactive learning.
- **View Slot Request**
  The admin view slot booking request module enables the administrator to monitor and manage all video call slot booking requests made by students. It provides a detailed overview of scheduled, pending, and completed slots between mentors and students. Admin can verify, approve, or modify booking details if necessary to ensure smooth communication flow.

STUDENTS

- **Register**
  In this module allows new students to create an account by providing necessary information such as name, email, phone number, and areas of interest.
- **Login**
  In this module enables registered students to access their accounts by entering their username and password.

- **View Skill Details**

  This module allows students to browse and view the skills and areas of expertise of available mentors. This module displays a comprehensive list of skills, categorized by subjects or fields of interest, to help students identify mentors who match their requirements.
- **Request to Mentor**

  In this module enables students to send a communication request to a mentor for a chat session. Students can choose mentors based on their skills and expertise, and send a request to initiate a conversation.
- **Start Chat**

  In this module enables students to initiate a conversation with an accepted mentor. Once the mentor accepts the chat request, the student can send messages to the mentor through the platform. The system allows real-time communication, fostering interaction and discussion.
- **Slot Booking**

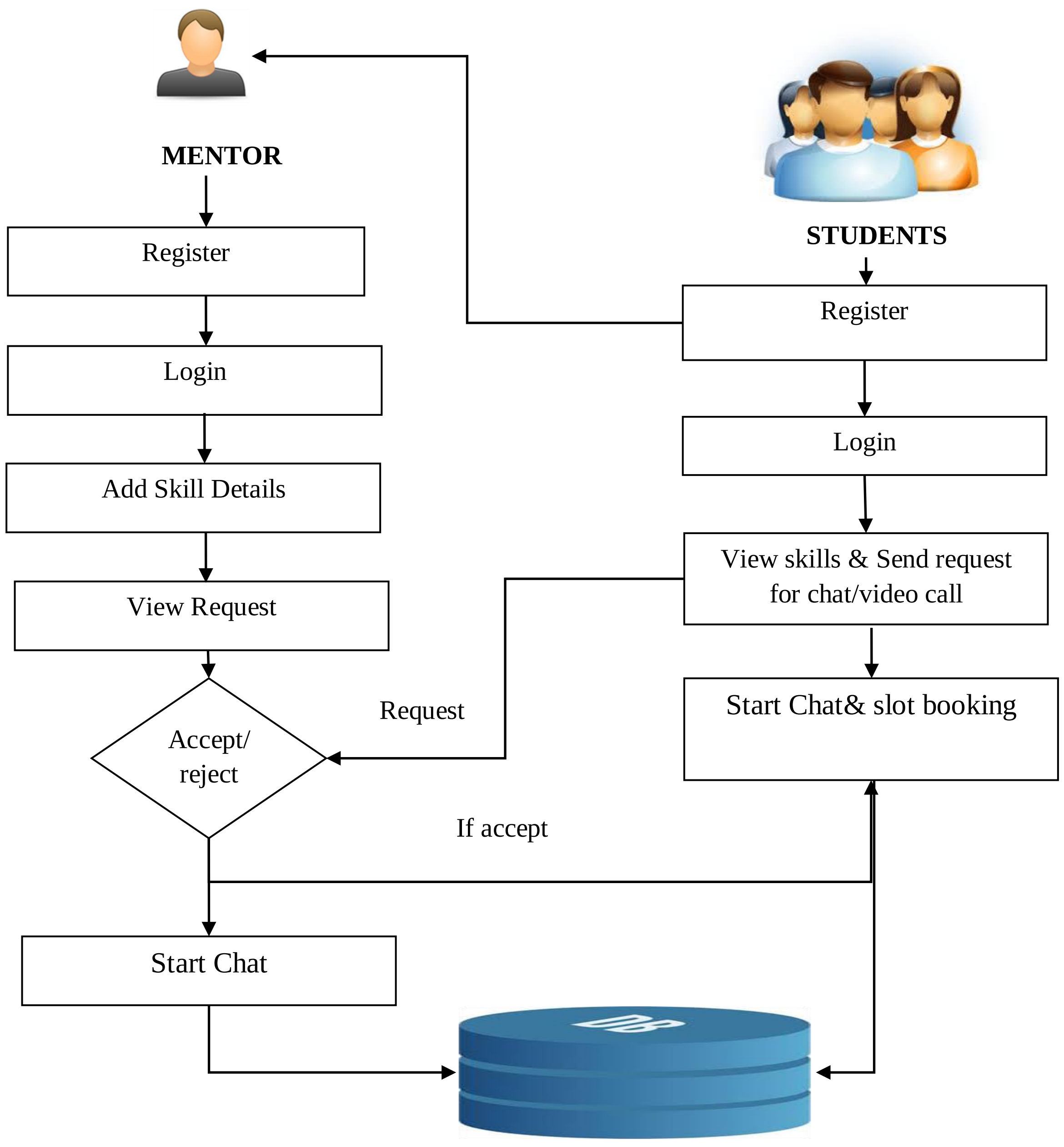
  The video call slot booking module allows students to schedule one-on-one virtual meetings with their assigned mentors based on mutual availability. Students can view available time slots and book a preferred time for a video call, ensuring organized and timely interactions. It also reduces scheduling conflicts and promotes structured guidance sessions. Notifications are sent to both mentor and student upon successful booking.

5.4 SYSTEM ARCHITECTURE

A system architecture or systems architecture is the conceptual model that defines the structure, behaviour, and more views of a system. An architecture description is a formal description and representation of a system, organized in a way that supports reasoning about the structures and behaviours of the system. System architecture can comprise system components, the externally visible properties of those components, the relationships (e.g. the behaviour) between them. It can provide a plan from which products can be procured, and systems developed, that will work together to implement the overall system. There have been efforts to formalize languages to describe system architecture, collectively these are called architecture description languages (ADLs).

Various organizations define systems architecture in different ways, including:

- An allocated arrangement of physical elements which provides the design solution for a consumer product or life-cycle process intended to satisfy the requirements of the functional architecture and the requirements baseline.
- Architecture comprises the most important, pervasive, top-level, strategic inventions, decisions, and their associated rationales about the overall structure (i.e., essential elements and their relationships) and associated characteristics and behavior.
- If documented, it may include information such as a detailed inventory of current hardware, software and networking capabilities; a description of long-range plans and priorities for future purchases, and a plan for upgrading and/or replacing dated equipment and software
- The composite of the design architectures for products and their life-cycle processes.


MENTOR
Register
Login
Add Skill Details
View Request
Accept/ reject
Start Chat
STUDENTS
Register
Login
View skills & Send request for chat/video call
Start Chat& slot booking
Request
If accept
DB

CHAPTER 6

SYSTEM TESTING

6.1 UNIT TESTING

The first test in the development process is the unit test. The source code is normally divided into modules, which in turn are divided into smaller units called units. These units have specific behaviour. The test done on these units of code is called unit test. Unit test depends upon the language on which the project is developed. Unit tests ensure that each unique path of the project performs accurately to the documented specifications and contains clearly defined inputs and expected results.

6.2 INTEGRATION TESTING

In integration testing modules are combined and tested as a group. Modules are typically code modules, individual applications, source and destination applications on a network, etc. Integration Testing follows unit testing and precedes system testing. Testing after the product is code complete. Betas are often widely distributed or even distributed to the public at large in hopes that they will buy the final product when it is released.

6.3 VALIDATION TESTING

The process of evaluating software during the development process or at the end of the development process to determine whether it satisfies specified business requirements. Validation Testing ensures that the product actually meets the client's needs. It can also be defined as to demonstrate that the product fulfills its intended use when deployed on appropriate environment.

6.4 SYSTEM TESTING

System testing is defined as testing of a complete and fully integrated software product. This testing falls in black-box testing wherein knowledge of the inner design of the code is not a pre-requisite and is done by the testing team. It is the final test to verify that the product to be delivered meets the specifications mentioned in the requirement document. It should investigate both functional and non-functional requirements.

CHAPTER 7

SYSTEM IMPLEMENTATION

Implementation is the stage in the project where the theoretical design is turned into a working system. The most critical stage is achieving a successful system and in giving confidence on the new system for the users, what it will work efficient and effectively. It involves careful planning, investing of the current system, and its constraints on implementation, design of methods to achieve the change over methods. The implementation process begins with preparing a plan for the implementation of the system. According to this plan, the activities are to be carried out in these plans; discussion has been made regarding the equipment, resources and how to test activities. The coding step translates a detail design representation into a programming language

Realization. Programming languages are vehicles for communication between human and computers programming language characteristics and coding style can profoundly affect software quality and maintainability. The coding is done with the following characteristics in mind.

- Ease of design to code translation.
- Code efficiency.
- Memory efficiency.
- Maintainability.

Implementation is the stage of the project when the theoretical design is turned out into a working system. Thus it can be considered to be the most critical stage in achieving a successful new system and in giving the user, confidence that the new system will work and be effective.

CHAPTER 8

CONCLUSION & FUTURE ENHANCEMENTS

8.1 CONCLUSION

In conclusion, the Mentor-Student Booking Application offers a streamlined and efficient solution for managing mentorship sessions. By enabling mentors to register, create detailed profiles with their skills, and easily manage booking requests, the system ensures that both students and mentors have a clear and structured approach to scheduling and communication. Students benefit from the ability to send booking requests, view real-time booking statuses, and engage in direct chats with mentors based on scheduled time slots, enhancing the overall experience. This application not only reduces the risks of missed appointments and scheduling conflicts but also fosters better communication, ensuring that students can always reach out to mentors when needed. With its centralized platform, the system brings organization, transparency, and efficiency to the mentor-student relationship, making the mentoring process more accessible and productive.

8.2 FUTURE ENHANCEMENTS

The future scope of this online mentoring system holds immense potential for further enhancement and scalability. In the coming years, the system can integrate advanced features such as AI-driven matching algorithms to recommend mentors based on students' learning styles, goals, and academic needs. Incorporating video chat functionality would enhance the mentoring experience, enabling face-to-face interactions, which can lead to more effective communication.

CHAPTER 9

APPENDIX

9.1 SAMPLE SOURCE CODE

```
from flask import Flask, render_template, flash, request, session, send_file, jsonify
from flask import render_template, redirect, url_for, request
import os
import mysql.connector
import random
import numpy as np
import pickle
import json
from flask import Flask, render_template, request
import nltk
from keras.models import load_model
from nltk.stem import WordNetLemmatizer
import datetime
import time
lemmatizer = WordNetLemmatizer()

# chat initialization
model = load_model("chatbot_model.h5")
intents = json.loads(open("intents.json").read())
words = pickle.load(open("words.pkl", "rb"))
classes = pickle.load(open("classes.pkl", "rb"))

app = Flask(__name__)
app.config['DEBUG']
app.config['SECRET_KEY'] = '7d441f27d441f27567d441f2b6176a'

@app.route("/")
def homepage():
    return render_template('index.html')
```

```
@app.route("/AdminLogin")
def AdminLogin():
   return render_template('AdminLogin.html')

@app.route("/NewMentor")
def NewMentor():
   return render_template('NewMentor.html')
@app.route('/MentorLogin')
def MentorLogin():
   return render_template('MentorLogin.html')
@app.route("/NewStudent")
def NewStudent():
   return render_template('NewStudent.html')
@app.route("/StudentLogin")
def StudentLogin():
   return render_template('StudentLogin.html')
@app.route("/adminlogin", methods=['GET', 'POST'])
def adminlogin():
   if request.method == 'POST':
      if request.form['uname'] == 'admin' and request.form['Password'] == 'admin':
         conn = mysql.connector.connect(user='root', password='', host='localhost',
database='1collegementordb')
         cur = conn.cursor()
         cur.execute("SELECT * FROM mentortb")
         data = cur.fetchall()
         return render_template('AdminHome.html', data=data)
      else:
         flash("UserName or Password Incorrect!")
         return render_template('AdminLogin.html')


@app.route("/AdminHome")
def AdminHome():
   conn = mysql.connector.connect(user='root', password='', host='localhost',
database='1collegementordb')
   cur = conn.cursor()
   cur.execute("SELECT * FROM mentortb ")
   data = cur.fetchall()
   return render_template('AdminHome.html', data=data)

@app.route("/AStudentInfo")
def AStudentInfo():
   conn = mysql.connector.connect(user='root', password='', host='localhost',
database='1collegementordb')
   cur = conn.cursor()
   cur.execute("SELECT * FROM regtb ")
   data = cur.fetchall()
   return render_template('AStudentInfo.html', data=data)

@app.route("/newmentor", methods=['GET', 'POST'])
def newuser():
   if request.method == 'POST':
      name = request.form['name']
      age = request.form['age']
      mobile = request.form['mobile']
      email = request.form['email']
      address = request.form['address']
      Subject = request.form['Subject']
      username = request.form['username']
      Password = request.form['Password']
      conn = mysql.connector.connect(user='root', password='', host='localhost',
database='1collegementordb')
      cursor = conn.cursor()
      cursor.execute("SELECT * from mentortb where username='" + username + "' ")
      data = cursor.fetchone()
      if data is None:
         conn = mysql.connector.connect(user='root', password='', host='localhost',
database='1collegementordb')
```

```
        cursor = conn.cursor()
        cursor.execute(
            "insert into mentortb values('','" + name + "','" + age + "','" + mobile + "','" + email
+ "','" + address + "','" + Subject + "','" +
            username + "','" + Password + "')")
        conn.commit()
        conn.close()
        return render_template('MentorLogin.html')
    else:
        flash('Already Register Username')
        return render_template('NewMentor.html')


@app.route("/mentorlogin", methods=['GET', 'POST'])
def mentorlogin():
    if request.method == 'POST':
        username = request.form['uname']
        password = request.form['Password']
        session['mname'] = request.form['uname']
        conn = mysql.connector.connect(user='root', password='', host='localhost',
database='1collegementordb')
        cursor = conn.cursor()
        cursor.execute("SELECT * from mentortb where username='" + username + "' and
Password='" + password + "'")
        data = cursor.fetchone()
        if data is None:
            flash('Username or Password is wrong')
            return render_template('MentorLogin.html')
        else:
            session['subject'] = data[6]
            conn = mysql.connector.connect(user='root', password='', host='localhost',
database='1collegementordb')
            cur = conn.cursor()
            cur.execute("SELECT * FROM mentortb where UserName='" + session['mname'] +
"' ")
```

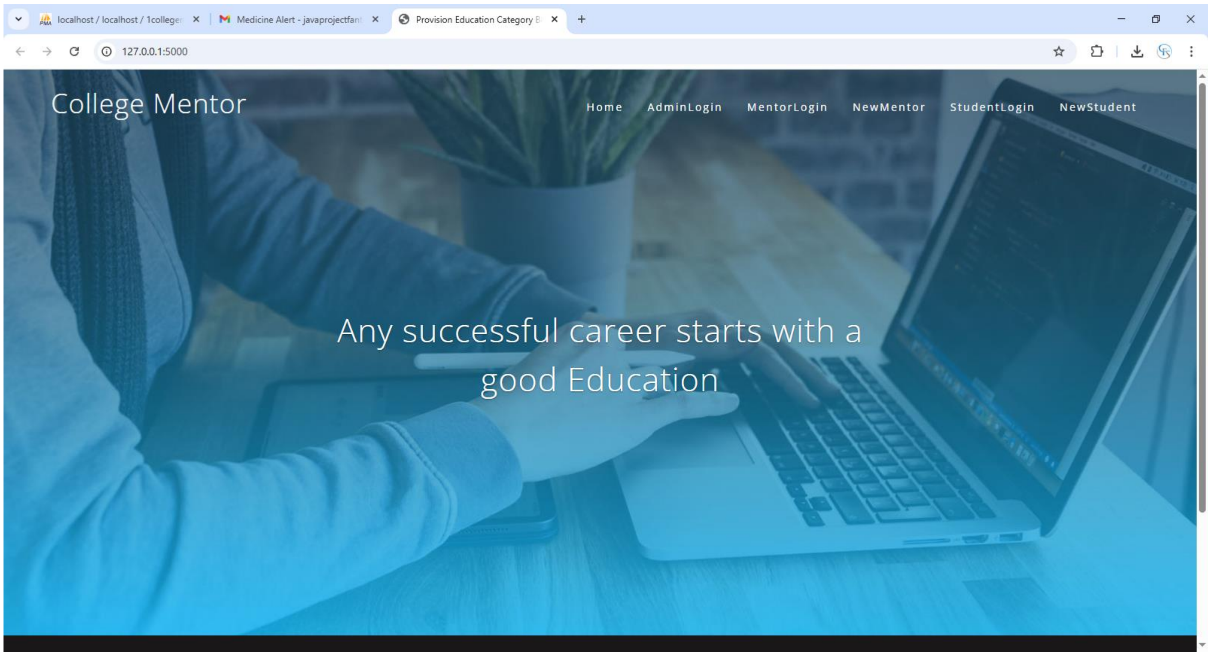
```
            data = cur.fetchall()
            return render_template('MentorHome.html', data=data)

@app.route("/MentorHome")
def MentorHome():
    conn = mysql.connector.connect(user='root', password='', host='localhost',
database='1collegementordb')
    cur = conn.cursor()
    cur.execute("SELECT * FROM mentortb where UserName='" + session['mname'] + "' ")
    data = cur.fetchall()
    return render_template('MentorHome.html', data=data)

@app.route('/ShareNotes')
def ShareNotes():
    conn = mysql.connector.connect(user='root', password='', host='localhost',
database='1collegementordb')
    cur = conn.cursor()
    cur.execute("SELECT * FROM sharetb where MName='" + session['mname'] + "' ")
    data = cur.fetchall()
    return render_template('ShareNotes.html', data=data, subject=session['subject'])

@app.route("/newsharenote", methods=['GET', 'POST'])
def newsharenote():
    if request.method == 'POST':
        name = request.form['name']
        Info = request.form['Info']
        Batch = request.form['Batch']
        file = request.files['file']
        fnew = random.randint(1111, 9999)
        savename = str(fnew) + file.filename
        file.save("static/upload/" + savename)
        ts = time.time()
        date = datetime.datetime.fromtimestamp(ts).strftime('%Y-%m-%d')
        conn = mysql.connector.connect(user='root', password='', host='localhost',
database='1collegementordb')
        cursor = conn.cursor()
        cursor.execute("SELECT * from regtb where Batch='" + Batch + "' ")
        data = cursor.fetchone()
        if data:
            conn = mysql.connector.connect(user='root', password='', host='localhost',
database='1collegementordb')
            cursor = conn.cursor()
            cursor.execute(
                "insert into sharetb values('','" + session[
                    'mname'] + "','" + name + "','" + savename + "','" + Info + "','" + Batch + "','" +
date + "')")
            conn.commit()
            conn.close()
            flash('Record Saved..!')
            conn = mysql.connector.connect(user='root', password='', host='localhost',
database='1collegementordb')
            cur = conn.cursor()
            cur.execute("SELECT * FROM sharetb where MName='" + session['mname'] + "' ")
            data = cur.fetchall()
            return render_template('ShareNotes.html', data=data, subject=session['subject'])
        else:
            flash('Student Not Found This Year')
            conn = mysql.connector.connect(user='root', password='', host='localhost',
database='1collegementordb')
            cur = conn.cursor()
            cur.execute("SELECT * FROM sharetb where MName='" + session['mname'] + "' ")
            data = cur.fetchall()
            return render_template('ShareNotes.html', data=data, subject=session['subject'])


@app.route("/MRemove")
def MRemove():
    id = request.args.get('id')
    conn = mysql.connector.connect(user='root', password='', host='localhost',
database='1collegementordb')
    cursor = conn.cursor()
    cursor.execute("delete from  sharetb  where id='" + id + "' ")
    conn.commit()
    conn.close()
    flash('Notes Removed..!')
    conn = mysql.connector.connect(user='root', password='', host='localhost',
database='1collegementordb')
    cur = conn.cursor()
    cur.execute("SELECT * FROM sharetb where MName='" + session['mname'] + "' ")
    data = cur.fetchall()
    return render_template('ShareNotes.html', data=data, subject=session['subject'])

@app.route("/MQueryInfo")
def MQueryInfo():
    conn = mysql.connector.connect(user='root', password='', host='localhost',
database='1collegementordb')
    cur = conn.cursor()
    cur.execute("SELECT * FROM Querytb where Answer='waiting' and Mname='" +
session['mname'] + "'  ")
    data = cur.fetchall()
    conn = mysql.connector.connect(user='root', password='', host='localhost',
database='1collegementordb')
    cur = conn.cursor()
    cur.execute("SELECT * FROM Querytb where Answer!='waiting' and Mname='" +
session['mname'] + "'  ")
    data1 = cur.fetchall()
    return render_template('MQueryInfo.html', data=data, data1=data1)

@app.route("/Forward")
def Forward():
    id = request.args.get('id')
    session['qid'] = id
    conn = mysql.connector.connect(user='root', password='', host='localhost',
database='1collegementordb')
    cur = conn.cursor()
    cur.execute("SELECT * FROM mentortb   ")
    data = cur.fetchall()
    return render_template('Forward.html', data=data)

@app.route("/Answer")
def Answer():
    id = request.args.get('id')
    session['qid'] = id
    return render_template('Answer.html')

@app.route("/answer", methods=['GET', 'POST'])
def answer():
    if request.method == 'POST':
        Answer = request.form['Answer']
        conn = mysql.connector.connect(user='root', password='', host='localhost',
database='1collegementordb')
        cursor = conn.cursor()
        cursor.execute(
            "update   Querytb set Answer='" + Answer + "'  where id='" + session['qid'] + "' ")
        conn.commit()
        conn.close()
        flash("Record Saved!")
        return MQueryInfo()

@app.route("/forwardq", methods=['GET', 'POST'])
def forwardq():
    if request.method == 'POST':
        ment = request.form['Batch']
        conn = mysql.connector.connect(user='root', password='', host='localhost',
database='1collegementordb')
        cursor = conn.cursor()
        cursor.execute(
            "update   Querytb set Mname='" + ment + "'  where id='" + session['qid'] + "' ")
        conn.commit()
        conn.close()
        flash("Forward to Mentor Saved!")
        return MQueryInfo()

@app.route("/newstudent", methods=['GET', 'POST'])
def newstudent():
    if request.method == 'POST':
        name = request.form['name']
        regno = request.form['regno']
        mobile = request.form['mobile']
        email = request.form['email']
        address = request.form['address']
        Subject = request.form['Batch']
        username = request.form['username']
        Password = request.form['Password']
        conn = mysql.connector.connect(user='root', password='', host='localhost',
database='1collegementordb')
        cursor = conn.cursor()
        cursor.execute("SELECT * from regtb where username='" + username + "' ")
        data = cursor.fetchone()
        if data is None:
            conn = mysql.connector.connect(user='root', password='', host='localhost',
database='1collegementordb')
            cursor = conn.cursor()
            cursor.execute(
                "insert into regtb values('','" + name + "','" + regno + "','" + mobile + "','" + email +
"','" + address + "','" + Subject + "','" +
                username + "','" + Password + "')")
            conn.commit()
            conn.close()
            return render_template('StudentLogin.html')
        else:
            flash('Already Register Username')
            return render_template('NewStudent.html')

@app.route("/studentlogin", methods=['GET', 'POST'])
def studentlogin():
    if request.method == 'POST':
        username = request.form['uname']
        password = request.form['Password']
        session['sname'] = request.form['uname']
        conn = mysql.connector.connect(user='root', password='', host='localhost',
database='1collegementordb')
        cursor = conn.cursor()
        cursor.execute("SELECT * from regtb where username='" + username + "' and
Password='" + password + "'")
        data = cursor.fetchone()
        if data is None:
            flash('Username or Password is wrong')
            return render_template('StudentLogin.html')
        else:
            session['regno'] = data[2]
            session['batch'] = data[6]
            conn = mysql.connector.connect(user='root', password='', host='localhost',
database='1collegementordb')
            cur = conn.cursor()
            cur.execute("SELECT * FROM regtb where UserName='" + session['sname'] + "' ")
            data = cur.fetchall()
            return render_template('StudentHome.html', data=data)

@app.route("/StudentHome")
def StudentHome():
    conn = mysql.connector.connect(user='root', password='', host='localhost',
database='1collegementordb')
    cur = conn.cursor()
    cur.execute("SELECT * FROM regtb where UserName='" + session['sname'] + "' ")
    data = cur.fetchall()
    return render_template('StudentHome.html', data=data)

@app.route("/SNotes")
def SNotes():
    conn = mysql.connector.connect(user='root', password='', host='localhost',
database='1collegementordb')
    cur = conn.cursor()
    cur.execute("SELECT * FROM sharetb where Batch='" + session['batch'] + "' ")
    data = cur.fetchall()
    return render_template('SNotes.html', data=data)

@app.route("/search", methods=['GET', 'POST'])
def search():
    if request.method == 'POST':
        Subject = request.form['Subject']
        conn = mysql.connector.connect(user='root', password='', host='localhost',
database='1collegementordb')
        cursor = conn.cursor()
        cursor.execute("SELECT * from sharetb where Subject='" + Subject + "' and Batch='" +
session['batch'] + "'")
        data = cursor.fetchone()
        if data is None:
            conn = mysql.connector.connect(user='root', password='', host='localhost',
database='1collegementordb')
            cur = conn.cursor()
            cur.execute("SELECT * FROM sharetb where Batch='" + session['batch'] + "' ")
            data = cur.fetchall()
            flash('Record Not Found..!')
            return render_template('SNotes.html', data=data)
        else:
            conn = mysql.connector.connect(user='root', password='', host='localhost',
database='1collegementordb')
            cur = conn.cursor()
            cur.execute("SELECT * from sharetb where Subject='" + Subject + "' and Batch='" +
session['batch'] + "'")
            data = cur.fetchall()
            return render_template('SNotes.html', data=data)

@app.route("/Download")
def Download():
    id = request.args.get('id')
    conn = mysql.connector.connect(user='root', password='', host='localhost',
database='1collegementordb')
    cursor = conn.cursor()
    cursor.execute("SELECT * from sharetb where id ='" + str(id) + "'   ")
    data = cursor.fetchone()
    if data is None:
        return 'material Not Upload'
    else:
        filename = "static/upload/" + data[3]
        return send_file(filename, as_attachment=True)

@app.route("/NewQuery")
def NewQuery():
    conn = mysql.connector.connect(user='root', password='', host='localhost',
database='1collegementordb')
    cursor = conn.cursor()
    cursor.execute("SELECT * from accepttb where Sname='" + session['sname'] + "' ")
    reg = cursor.fetchone()
    if reg:
        mname = reg[1]
    conn = mysql.connector.connect(user='root', password='', host='localhost',
database='1collegementordb')
    cur = conn.cursor()
    cur.execute("SELECT * FROM Mentortb  where UserName='"+ mname +"' and
Status='Accepted'")
    data = cur.fetchall()
    return render_template('NewQuery.html', data=data)

@app.route("/msearch", methods=['GET', 'POST'])
def msearch():
    if request.method == 'POST':
        Subject = request.form['Subject']
        conn = mysql.connector.connect(user='root', password='', host='localhost',
database='1collegementordb')
        cursor = conn.cursor()
        cursor.execute("SELECT * from Mentortb where Subject='" + Subject + "' ")
        data = cursor.fetchone()
        if data is None:
            conn = mysql.connector.connect(user='root', password='', host='localhost',
database='1collegementordb')
            cur = conn.cursor()
            cur.execute("SELECT * FROM Mentortb  ")
            data = cur.fetchall()
            flash('Record Not Found..!')
            return render_template('NewQuery.html', data=data)
        else:
            conn = mysql.connector.connect(user='root', password='', host='localhost',
database='1collegementordb')
            cur = conn.cursor()
            cur.execute("SELECT * from Mentortb where Subject='" + Subject + "' ")
            data = cur.fetchall()
            return render_template('NewQuery.html', data=data)

@app.route("/newq")
def newq():
    sub = request.args.get('sub')
    mname = request.args.get('mname')
    return render_template('NewQuerys.html', sub=sub, mname=mname)

@app.route("/newquery", methods=['GET', 'POST'])
```

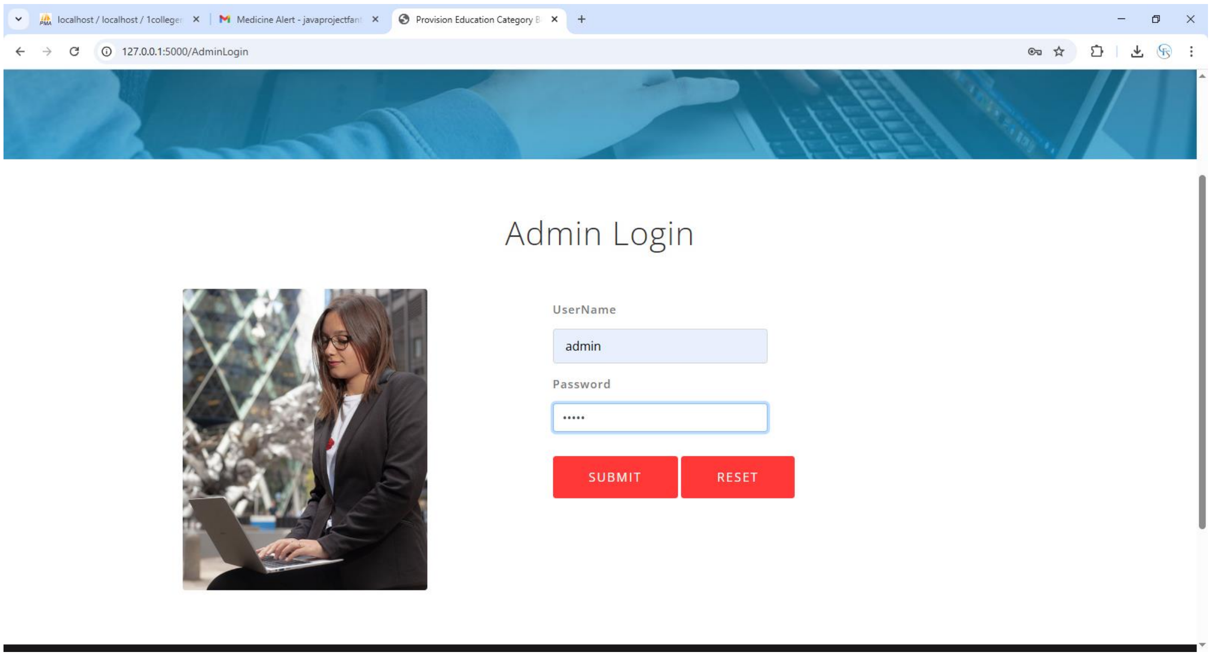
```
def newquery():
    if request.method == 'POST':
        mname = request.form['mname']
        sub = request.form['sub']
        Query = request.form['Query']
        ts = time.time()
        date = datetime.datetime.fromtimestamp(ts).strftime('%Y-%m-%d')
        conn = mysql.connector.connect(user='root', password='', host='localhost',
database='1collegementordb')
        cursor = conn.cursor()
        cursor.execute(
            "insert into Querytb values('','" + mname + "','" + sub + "','" + Query + "','waiting','" +
date + "','" +
            session['sname'] + "')")
        conn.commit()
        conn.close()
        flash('Record Saved..!')
        return NewQuery()


@app.route("/SAnswer")
def SAnswer():
    conn = mysql.connector.connect(user='root', password='', host='localhost',
database='1collegementordb')
    cur = conn.cursor()
    cur.execute("SELECT * FROM Querytb where Sname='" + session['sname'] + "'  ")
    data = cur.fetchall()
    return render_template('SAnswer.html', data=data)

@app.route("/Qsearch", methods=['GET', 'POST'])
def Qsearch():
    if request.method == 'POST':
        Subject = request.form['Subject']
        conn = mysql.connector.connect(user='root', password='', host='localhost',
database='1collegementordb')
        cursor = conn.cursor()
        cursor.execute("SELECT * from Querytb where Subject='" + Subject + "' ")
        data = cursor.fetchone()
        if data is None:
            conn = mysql.connector.connect(user='root', password='', host='localhost',
database='1collegementordb')
            cur = conn.cursor()
            cur.execute("SELECT * FROM Querytb where Sname='" + session['sname'] + "'  ")
            data = cur.fetchall()
            flash('Record Not Found..!')
            return render_template('SAnswer.html', data=data)
        else:
            conn = mysql.connector.connect(user='root', password='', host='localhost',
database='1collegementordb')
            cur = conn.cursor()
            cur.execute("SELECT * FROM Querytb where Sname='" + session['sname'] + "'  ")
            data = cur.fetchall()
            return render_template('SAnswer.html', data=data)

@app.route('/Chat')
def Chat():
    return render_template('Chat.html')

@app.route("/ask", methods=['GET', 'POST'])
def ask():
    message = str(request.form['messageText'])
# msg = request.form["msg"]
msg = message
    print(msg)
    if msg.startswith('my name is'):
        name = msg[11:]
        ints = predict_class(msg, model)
        res1 = getResponse(ints, intents)
        res = res1.replace("{n}", name)
    elif msg.startswith('hi my name is'):
        name = msg[14:]
        ints = predict_class(msg, model)
        res1 = getResponse(ints, intents)
        res = res1.replace("{n}", name)
    elif msg == "Mentor" or msg == "mentor" or msg == "MENTOR":
        conn1 = mysql.connector.connect(user='root', password='', host='localhost',
database='1collegementordb')
# search Result
cur1 = conn1.cursor()
        cur1.execute(
            "SELECT *  from mentortb  ")
        data1 = cur1.fetchall()
        for item1 in data1:
            ss1 = '<a href="http://127.0.0.1:5000/newq?id='
            ss11 = item1[0] + "&sub" + item1[6] + "&mname" + item1[7] +
'">NewQuery</a><br>'
            price1 = '<p class="price">SubjectName ' + item1[6] + '</p><br>'
            bot_response1 = ss1 + ss11 + price1
            if (bott1 == ""):
                bott1 = bot_response1
            else:
                bott1 = bott1 + bot_response1
            print(bott1)
        res = bott1
    else:
        ints = predict_class(msg, model)
        res = getResponse(ints, intents)
# return res
return jsonify({'status': 'OK', 'answer': res})

@app.route("/get", methods=["POST"])
def chatbot_response():
    msg = request.form["msg"]
    bott1 = ''
    if msg.startswith('my name is'):
        name = msg[11:]
        ints = predict_class(msg, model)
        res1 = getResponse(ints, intents)
        res = res1.replace("{n}", name)
    elif msg.startswith('hi my name is'):
        name = msg[14:]
        ints = predict_class(msg, model)
        res1 = getResponse(ints, intents)
        res = res1.replace("{n}", name)
    elif msg == "Mentor" or msg == "mentor" or msg == "MENTOR":
        conn1 = mysql.connector.connect(user='root', password='', host='localhost',
database='1collegementordb')
# search Result
cur1 = conn1.cursor()
        cur1.execute(
            "SELECT *  from mentortb  ")
        data1 = cur1.fetchall()
        for item1 in data1:
            ss1 = '<a href="http://127.0.0.1:5000/newq?id='
            ss11 = str(item1[0]) + "&sub=" + item1[6] + "&mname=" + item1[7] +
'">NewQuery</a><br>'
            price2 = '<p class="price">MentorName ' + item1[7] + '</p><br>'
            price1 = '<p class="price">SubjectName ' + item1[6] + '</p><br>'
            bot_response1 = price2 + price1 +ss1 + ss11
            if (bott1 == ""):
                bott1 = bot_response1
            else:
                bott1 = bott1 + bot_response1
            print(bott1)
        res = bott1
    else:
        ints = predict_class(msg, model)
        res = getResponse(ints, intents)
    return res

# chat functionalities
def clean_up_sentence(sentence):
    sentence_words = nltk.word_tokenize(sentence)
    sentence_words = [lemmatizer.lemmatize(word.lower()) for word in sentence_words]
    return sentence_words

# return bag of words array: 0 or 1 for each word in the bag that exists in the sentence
def bow(sentence, words, show_details=True):
# tokenize the pattern
sentence_words = clean_up_sentence(sentence)
# bag of words - matrix of N words, vocabulary matrix
bag = [0] * len(words)
    for s in sentence_words:
        for i, w in enumerate(words):
            if w == s:
# assign 1 if current word is in the vocabulary position
bag[i] = 1
                if show_details:
                    print("found in bag: %s" % w)
    return np.array(bag)
def predict_class(sentence, model):
# filter out predictions below a threshold
p = bow(sentence, words, show_details=False)
    res = model.predict(np.array([p]))[0]
    ERROR_THRESHOLD = 0.25
    results = [[i, r] for i, r in enumerate(res) if r > ERROR_THRESHOLD]
# sort by strength of probability
results.sort(key=lambda x: x[1], reverse=True)
    return_list = []
    for r in results:
        return_list.append({"intent": classes[r[0]], "probability": str(r[1])})
    return return_list

import random
def getResponse(ints, intents_json):
# Check if the 'ints' list is empty
if not ints:
        return "I'm sorry, I didn't understand that."
# Extract the predicted tag from the first element
tag = ints[0].get("intent")
# Extract the list of intents from the JSON data
list_of_intents = intents_json.get("intents", [])
# Search for the matching intent and return a random response
for intent in list_of_intents:
        if intent["tag"] == tag:
            return random.choice(intent["responses"])
    return "I'm sorry, I don't have a response for that."

@app.route("/NewAccept")
def NewAccept():
    conn = mysql.connector.connect(user='root', password='', host='localhost',
database='1collegementordb')
    cur = conn.cursor()
    cur.execute("SELECT * FROM Mentortb")
    data = cur.fetchall()
    return render_template('NewAccept.html', data=data)

@app.route("/Request")
def Request():
    id = request.args.get('id')
    uname = session['sname']
    conn = mysql.connector.connect(user='root', password='', host='localhost',
database='1collegementordb')
    cur = conn.cursor()
    cur.execute("SELECT * FROM Mentortb where id='" + id + "'")
    job = cur.fetchone()
    if job:
        mname = job[7]
    conn = mysql.connector.connect(user='root', password='', host='localhost',
database='1collegementordb')
    cur = conn.cursor()
    cur.execute("SELECT * FROM accepttb where Mname='" + mname + "' and Sname='"+
uname +"'")
    alr = cur.fetchone()
    if alr:
        flash('Already Requested')
        conn = mysql.connector.connect(user='root', password='', host='localhost',
database='1collegementordb')
        cur = conn.cursor()
        cur.execute("SELECT * FROM Mentortb  ")
        data = cur.fetchall()
        return render_template('NewAccept.html', data=data)
    else:
        conn = mysql.connector.connect(user='root', password='', host='localhost',
database='1collegementordb')
        cursor = conn.cursor()
        cursor.execute(
            "INSERT INTO accepttb VALUES ('','" + mname + "','" + uname + "','Waiting')")
        conn.commit()
        conn.close()
        flash('Mentor Apply successful')
        conn = mysql.connector.connect(user='root', password='', host='localhost',
database='1collegementordb')
        cur = conn.cursor()
        cur.execute("SELECT * FROM Mentortb  ")
        data = cur.fetchall()
        return render_template('NewAccept.html', data=data)
```

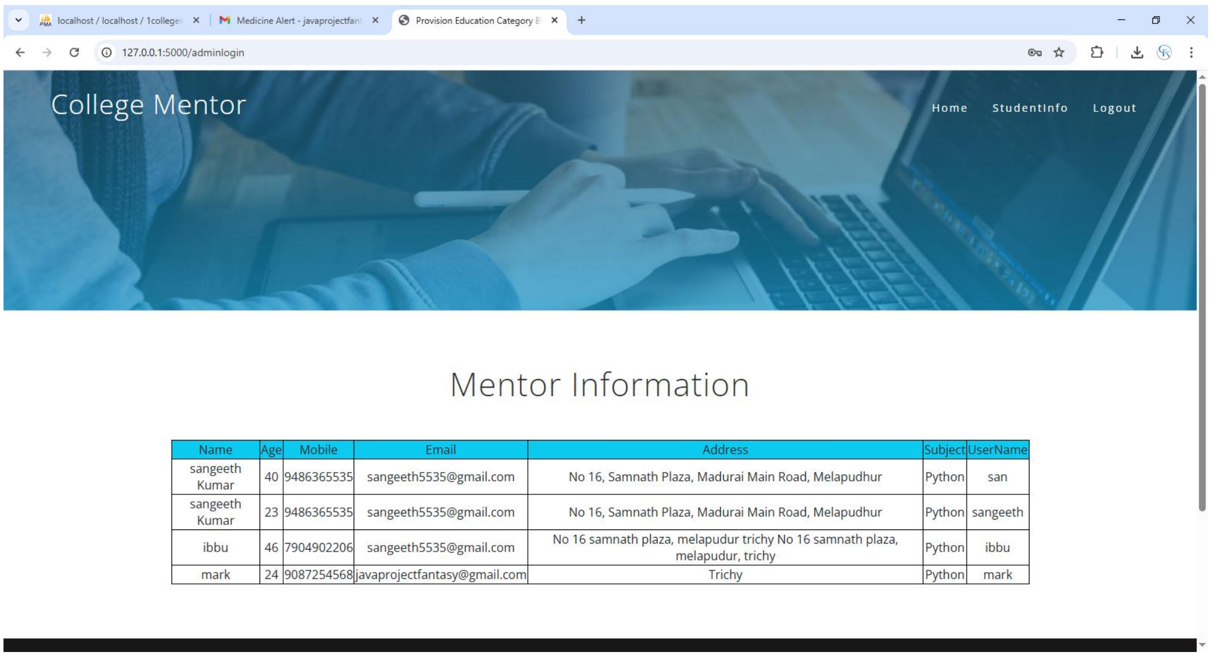
```
@app.route("/MRequest")
def MRequest():
    conn = mysql.connector.connect(user='root', password='', host='localhost',
database='1collegementordb')
    cur = conn.cursor()
    cur.execute("SELECT * FROM accepttb where Status='Waiting' and Mname='"+
session['mname'] +"' ")
    data = cur.fetchall()
    return render_template('MRequest.html', data=data)

@app.route("/Reject")
def Reject():
    id = request.args.get('id')
    conn = mysql.connector.connect(user='root', password='', host='localhost',
database='1collegementordb')
    cursor = conn.cursor()
    cursor.execute(
        "update accepttb set Status='Rejected' where id='" + id + "'")
    conn.commit()
    conn.close()
    flash('Mentor Request Rejected...!')
    conn = mysql.connector.connect(user='root', password='', host='localhost',
database='1collegementordb')
    cur = conn.cursor()
    cur.execute("SELECT * FROM accepttb where Status='Waiting' and Sname='" +
session['mname'] + "' ")
    data = cur.fetchall()
    return render_template('MRequest.html', data=data)

@app.route("/Accept")
def Accept():
    id = request.args.get('id')
    conn = mysql.connector.connect(user='root', password='', host='localhost',
database='1collegementordb')
```

```
    cursor = conn.cursor()
    cursor.execute(
        "update accepttb set Status='Accepted' where id='" + id + "'")
    conn.commit()
    conn.close()
    flash('Mentor Request Accepted...!')
    conn = mysql.connector.connect(user='root', password='', host='localhost',
database='1collegementordb')
    cur = conn.cursor()
    cur.execute("SELECT * FROM accepttb where Status='Waiting' and Sname='" +
session['mname'] + "' ")
    data = cur.fetchall()
    return render_template('MRequest.html', data=data)
if __name__ == '__main__':
    app.run(debug=True, use_reloader=True)
```

9.2 SCREEN SHOTS

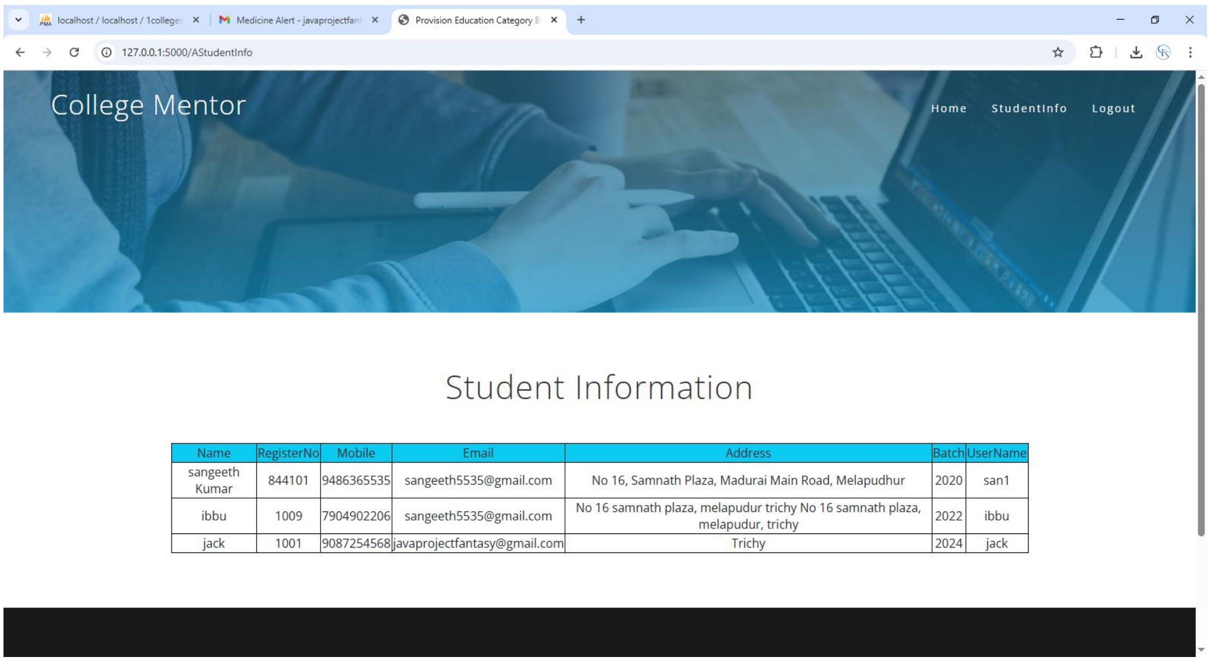

localhost / localhost / 1college
Medicine Alert - javaprojectfan
Provision Education Category
127.0.0.1:5000
College Mentor
Home
AdminLogin
MentorLogin
NewMentor
StudentLogin
NewStudent
Any successful career starts with a good Education


localhost / localhost / 1college
Medicine Alert - javaprojectfan
Provision Education Category
127.0.0.1:5000/AdminLogin
Admin Login
UserName
admin
Password
•••••
SUBMIT
RESET


localhost / localhost / 1college
Medicine Alert - javaprojectfant
Provision Education Category B
127.0.0.1:5000/adminlogin
College Mentor
Home
StudentInfo
Logout

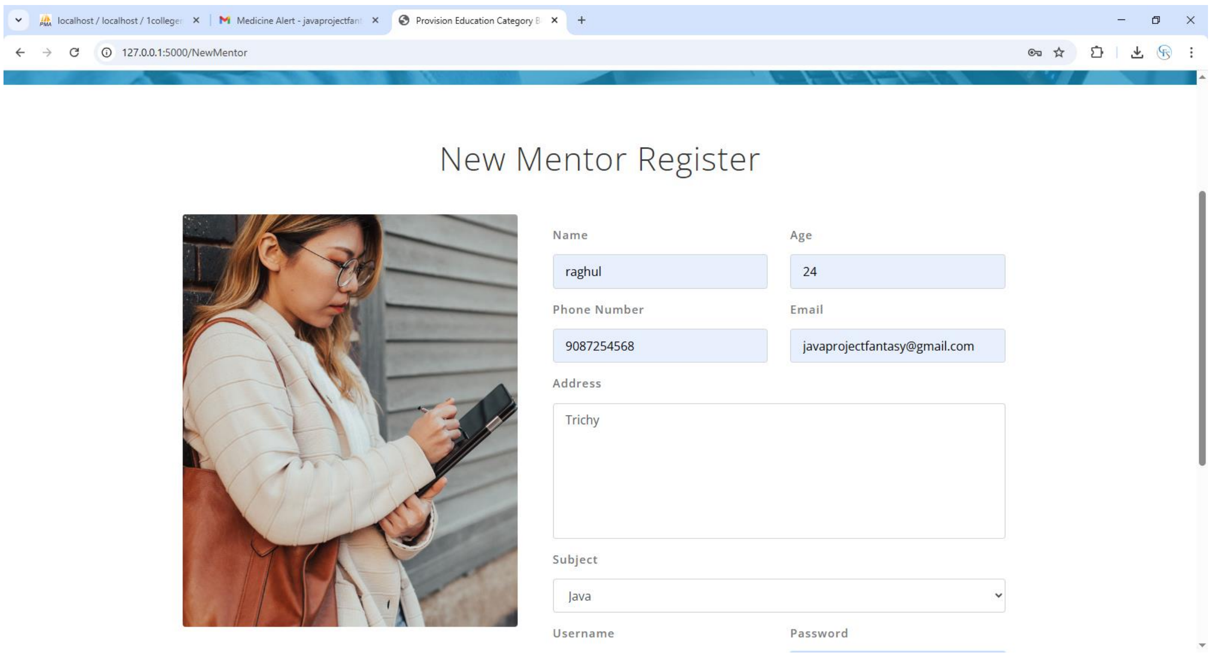
Mentor Information

Name	Age	Mobile	Email	Address	Subject	UserName
sangeeth Kumar	40	9486365535	sangeeth5535@gmail.com	No 16, Samnath Plaza, Madurai Main Road, Melapudhur	Python	san
sangeeth Kumar	23	9486365535	sangeeth5535@gmail.com	No 16, Samnath Plaza, Madurai Main Road, Melapudhur	Python	sangeeth
ibbu	46	7904902206	sangeeth5535@gmail.com	No 16 samnath plaza, melapudur trichy No 16 samnath plaza, melapudur, trichy	Python	ibbu
mark	24	9087254568	javaprojectfantasy@gmail.com	Trichy	Python	mark


localhost / localhost / 1college
Medicine Alert - javaprojectfant
Provision Education Category B
127.0.0.1:5000/AStudentInfo
College Mentor
Home
StudentInfo
Logout

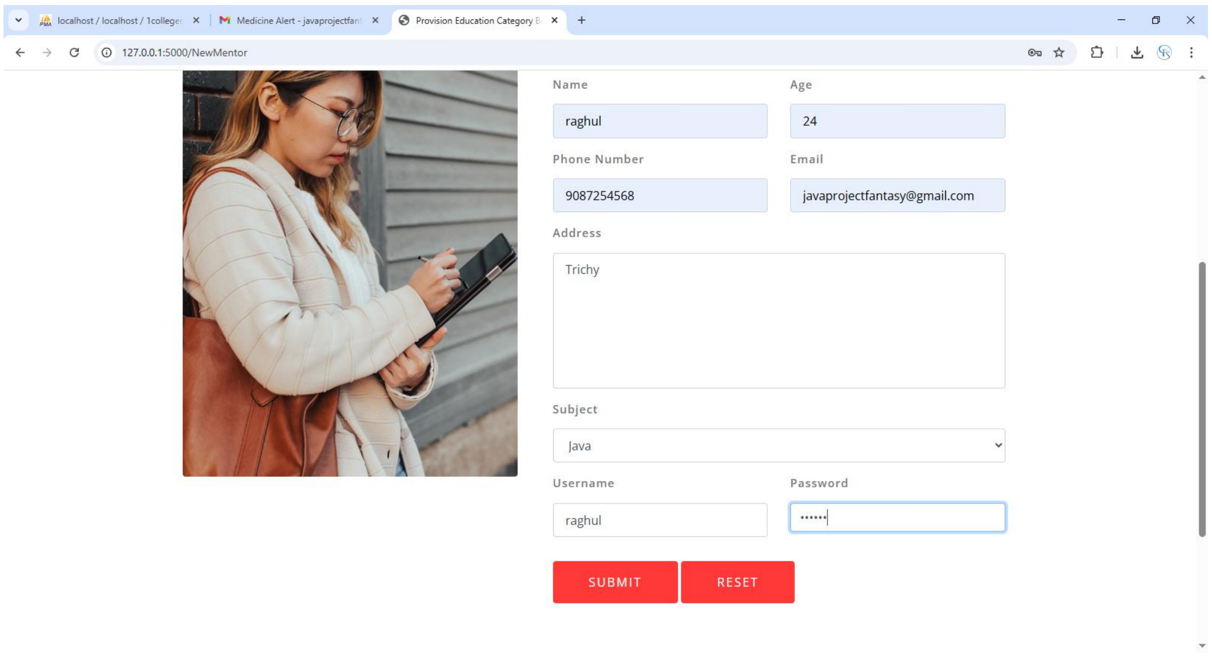
Student Information

Name	RegisterNo	Mobile	Email	Address	Batch	UserName
sangeeth Kumar	844101	9486365535	sangeeth5535@gmail.com	No 16, Samnath Plaza, Madurai Main Road, Melapudhur	2020	san1
ibbu	1009	7904902206	sangeeth5535@gmail.com	No 16 samnath plaza, melapudur trichy No 16 samnath plaza, melapudur, trichy	2022	ibbu
jack	1001	9087254568	javaprojectfantasy@gmail.com	Trichy	2024	jack

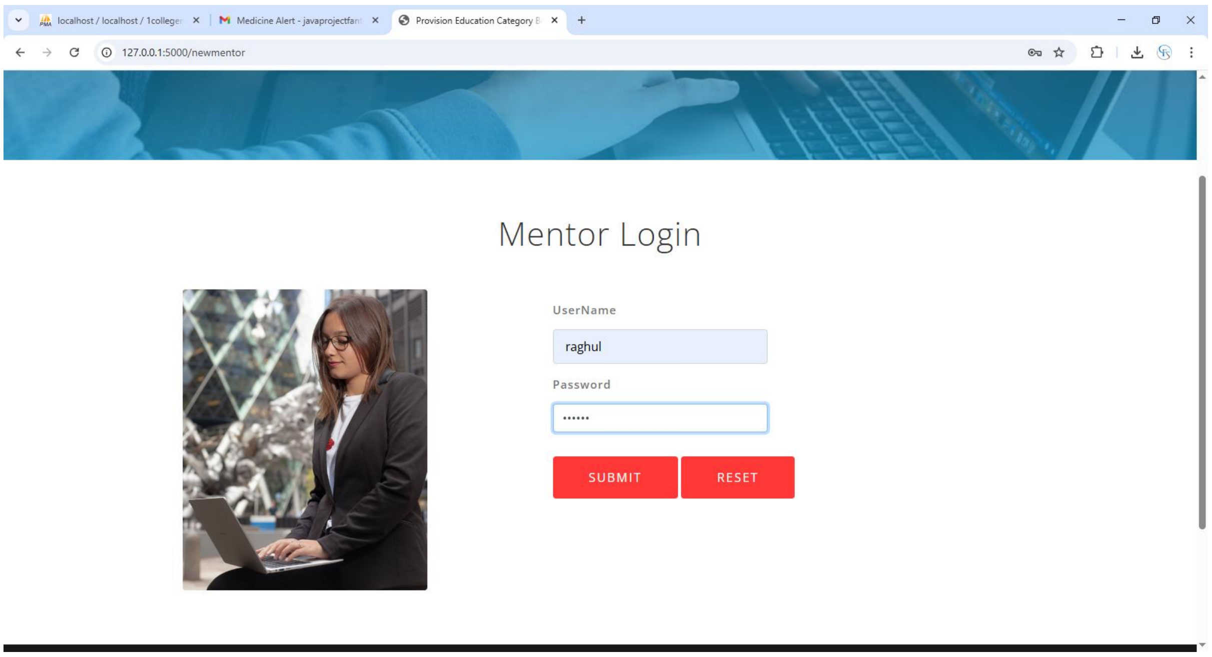

localhost / localhost / 1college
Medicine Alert - javaprojectfan
Provision Education Category
127.0.0.1:5000/NewMentor

New Mentor Register

Name: raghul
Age: 24
Phone Number: 9087254568
Email: javaprojectfantasy@gmail.com
Address: Trichy
Subject: Java
Username
Password


localhost / localhost / 1college
Medicine Alert - javaprojectfan
Provision Education Category
127.0.0.1:5000/NewMentor

Name: raghul
Age: 24
Phone Number: 9087254568
Email: javaprojectfantasy@gmail.com
Address: Trichy
Subject: Java
Username: raghul
Password: ••••••

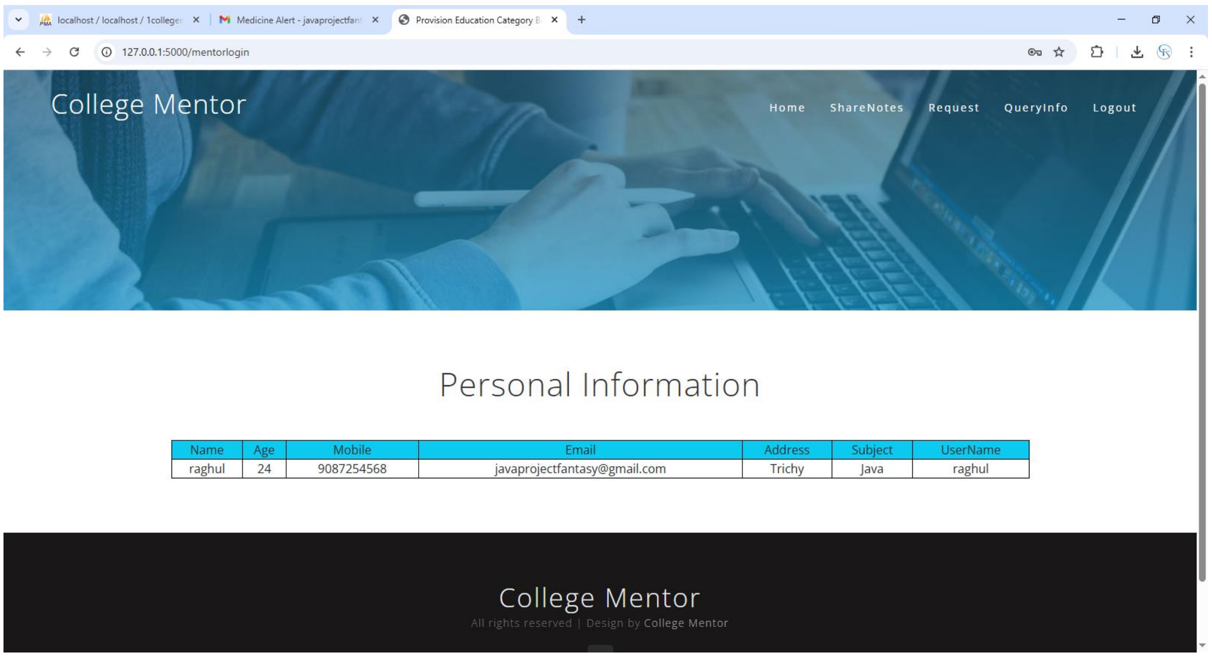

localhost / localhost / 1college
Medicine Alert - javaprojectfan
Provision Education Category
127.0.0.1:5000/newmentor

Mentor Login

UserName: raghul

Password: ••••••

SUBMIT RESET

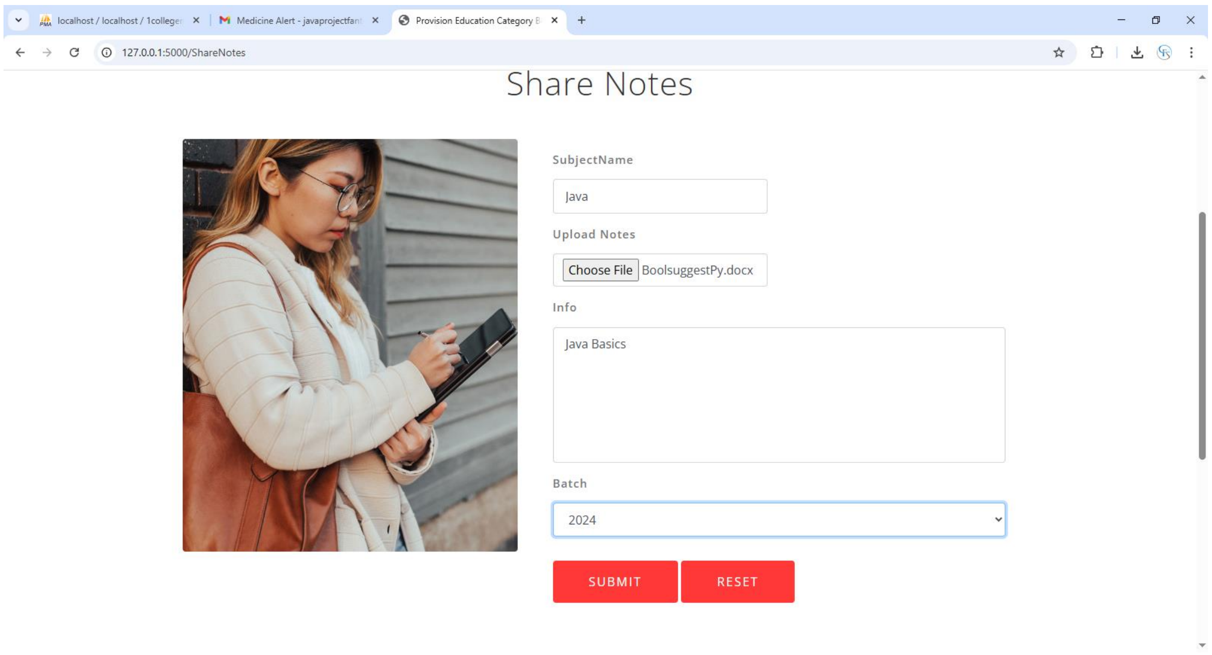

localhost / localhost / 1college
Medicine Alert - javaprojectfan
Provision Education Category
127.0.0.1:5000/mentorlogin
College Mentor
Home
ShareNotes
Request
QueryInfo
Logout

Personal Information

Name	Age	Mobile	Email	Address	Subject	UserName
raghul	24	9087254568	javaprojectfantasy@gmail.com	Trichy	Java	raghul

College Mentor

All rights reserved | Design by College Mentor


localhost / localhost / 1college
Medicine Alert - javaprojectfan
Provision Education Category
127.0.0.1:5000/ShareNotes

Share Notes

SubjectName

Java

Upload Notes

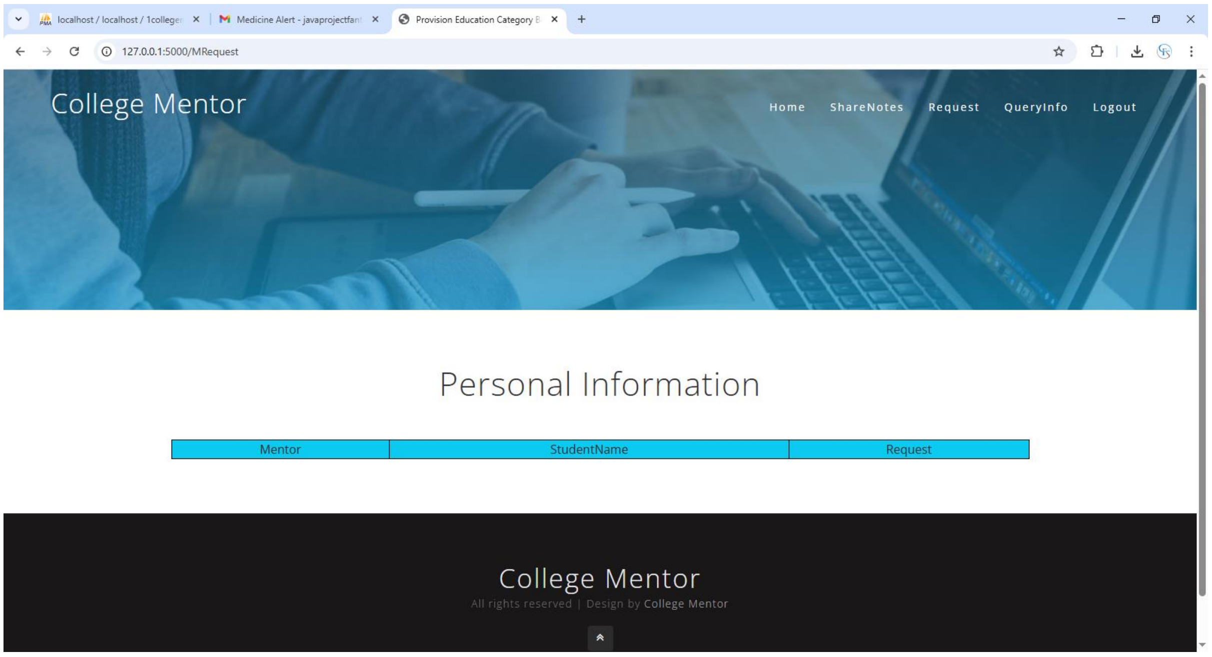
Choose File BoolsuggestPy.docx

Info

Java Basics

Batch

2024

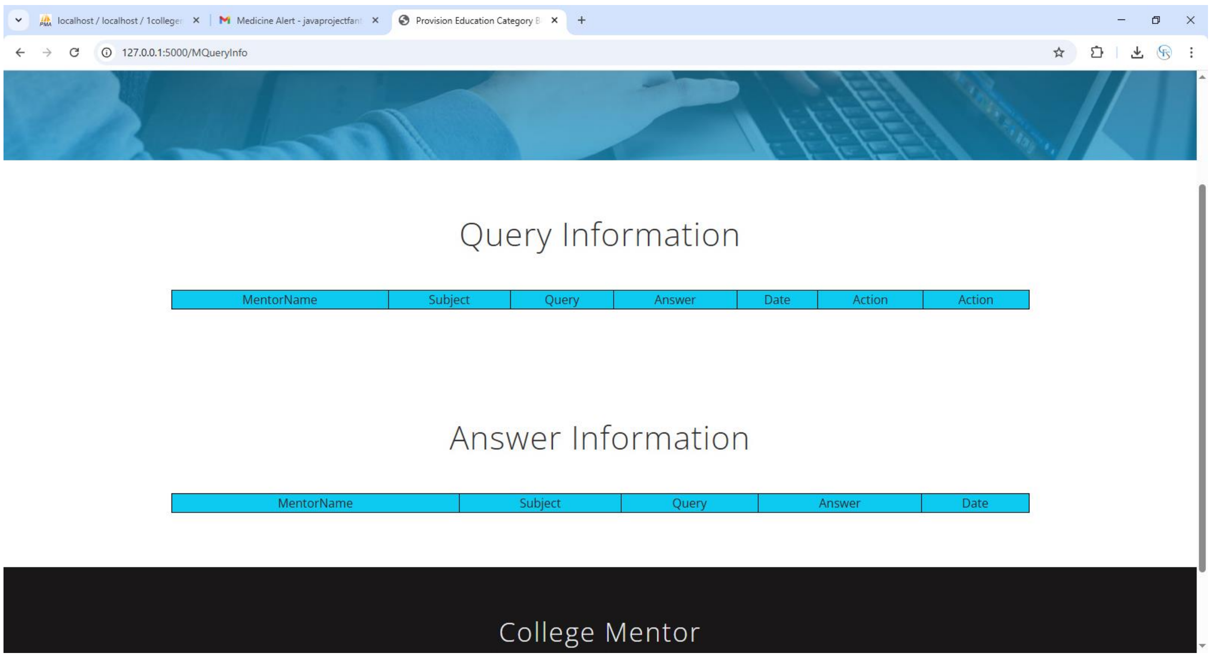

localhost / localhost / 1college
Medicine Alert - javaprojectfan
Provision Education Category
127.0.0.1:5000/MRequest
College Mentor
Home
ShareNotes
Request
QueryInfo
Logout

Personal Information

Mentor	StudentName	Request

College Mentor

All rights reserved | Design by College Mentor



localhost / localhost / 1college | Medicine Alert - javaprojectfan | Provision Education Category

127.0.0.1:5000/MQueryInfo

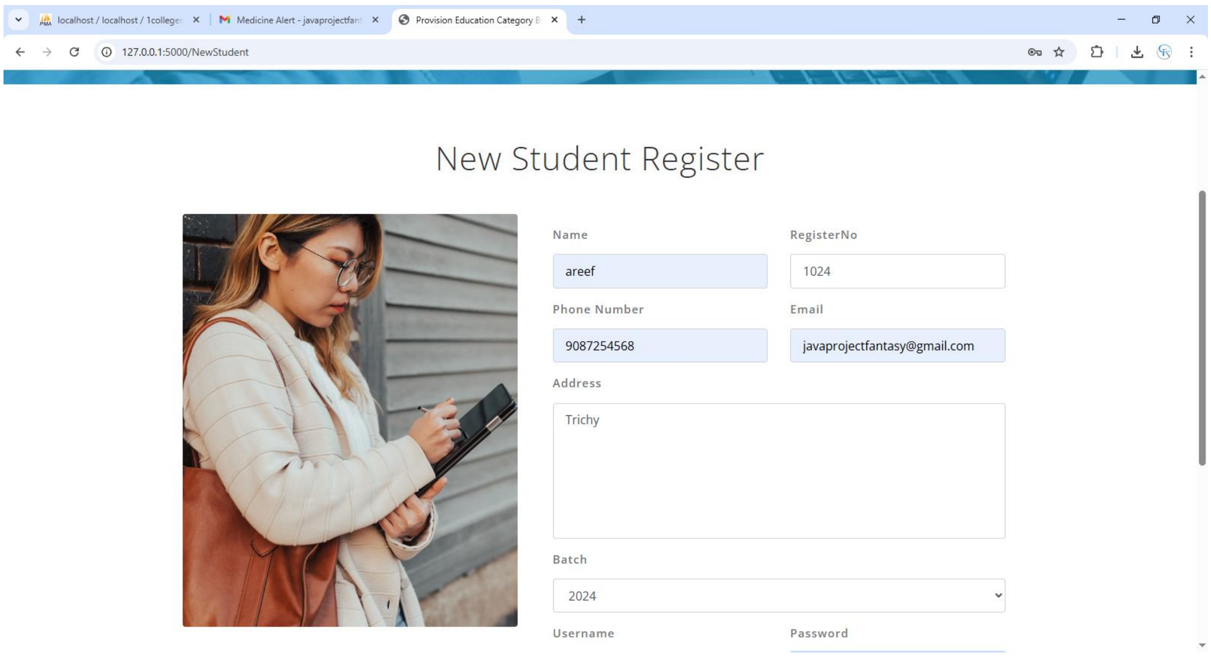
Query Information

MentorName	Subject	Query	Answer	Date	Action	Action

Answer Information

MentorName	Subject	Query	Answer	Date

College Mentor



localhost / localhost / 1college | Medicine Alert - javaprojectfan | Provision Education Category

127.0.0.1:5000/NewStudent

New Student Register

Name	RegisterNo
areef	1024
Phone Number	Email
9087254568	javaprojectfantasy@gmail.com

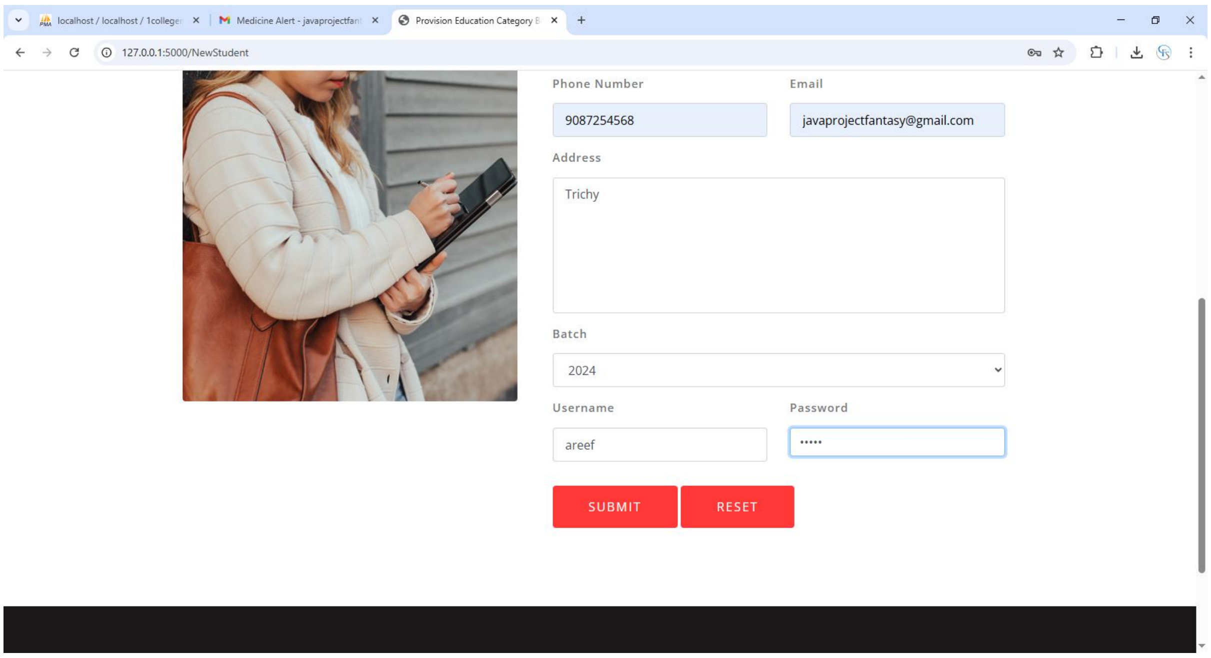
Address

Trichy

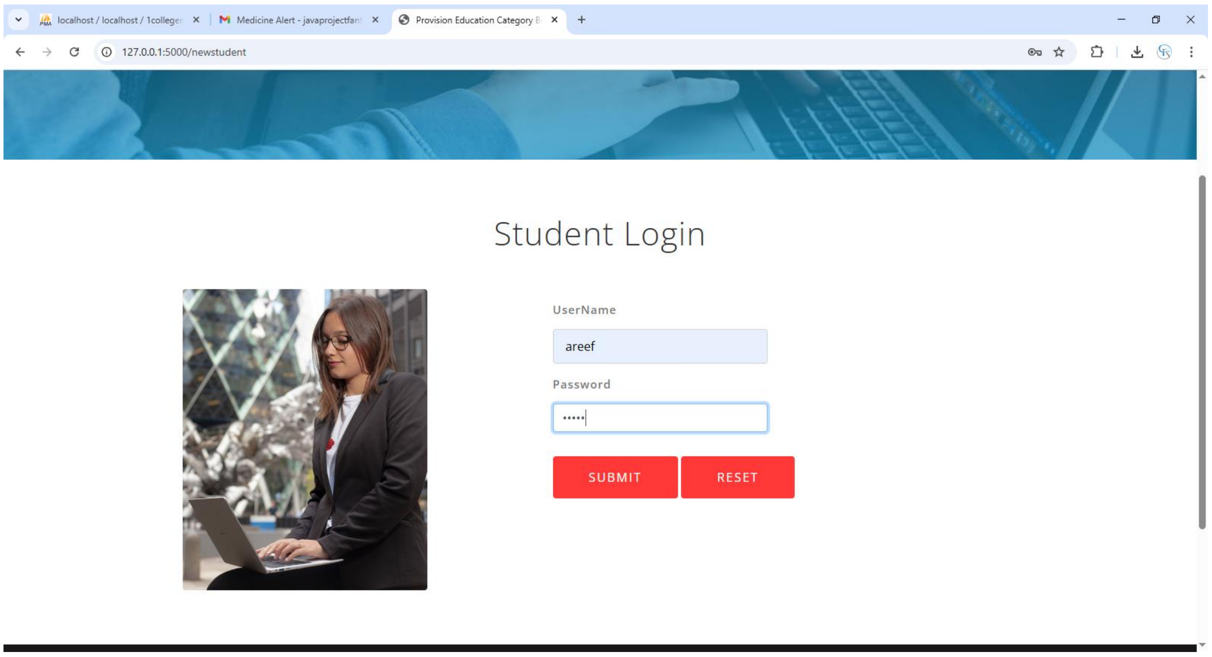
Batch

2024

Username	Password

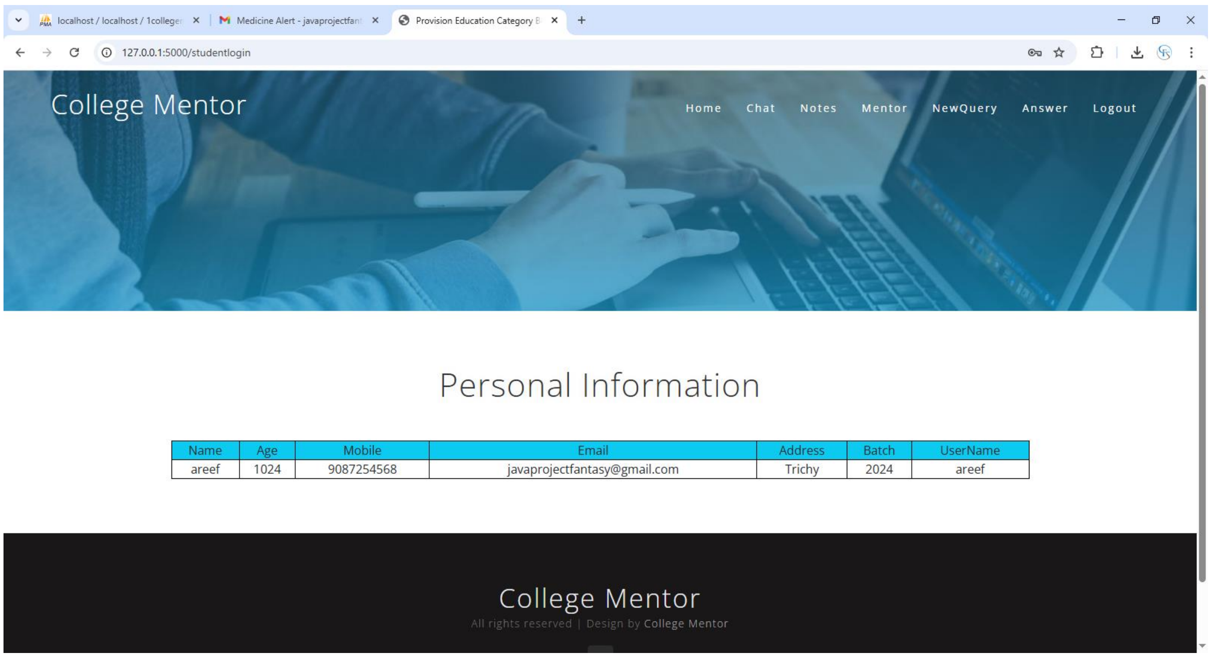

localhost / localhost / 1college
Medicine Alert - javaprojectfant
Provision Education Category B
127.0.0.1:5000/NewStudent

Phone Number: 9087254568
Email: javaprojectfantasy@gmail.com
Address: Trichy
Batch: 2024
Username: areef
Password: •••••


localhost / localhost / 1college
Medicine Alert - javaprojectfant
Provision Education Category B
127.0.0.1:5000/newstudent

Student Login

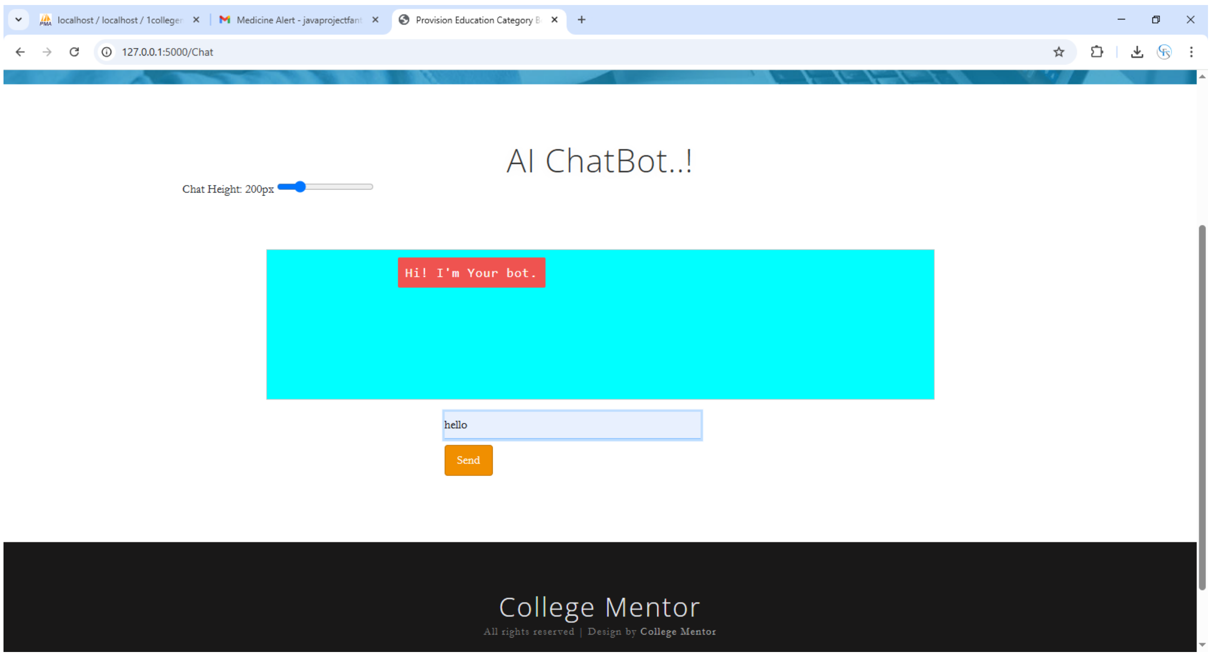
UserName: areef
Password: •••••


localhost / localhost / 1college
Medicine Alert - javaprojectfan
Provision Education Category
127.0.0.1:5000/studentlogin
College Mentor
Home
Chat
Notes
Mentor
NewQuery
Answer
Logout

Personal Information

Name	Age	Mobile	Email	Address	Batch	UserName
areef	1024	9087254568	javaprojectfantasy@gmail.com	Trichy	2024	areef

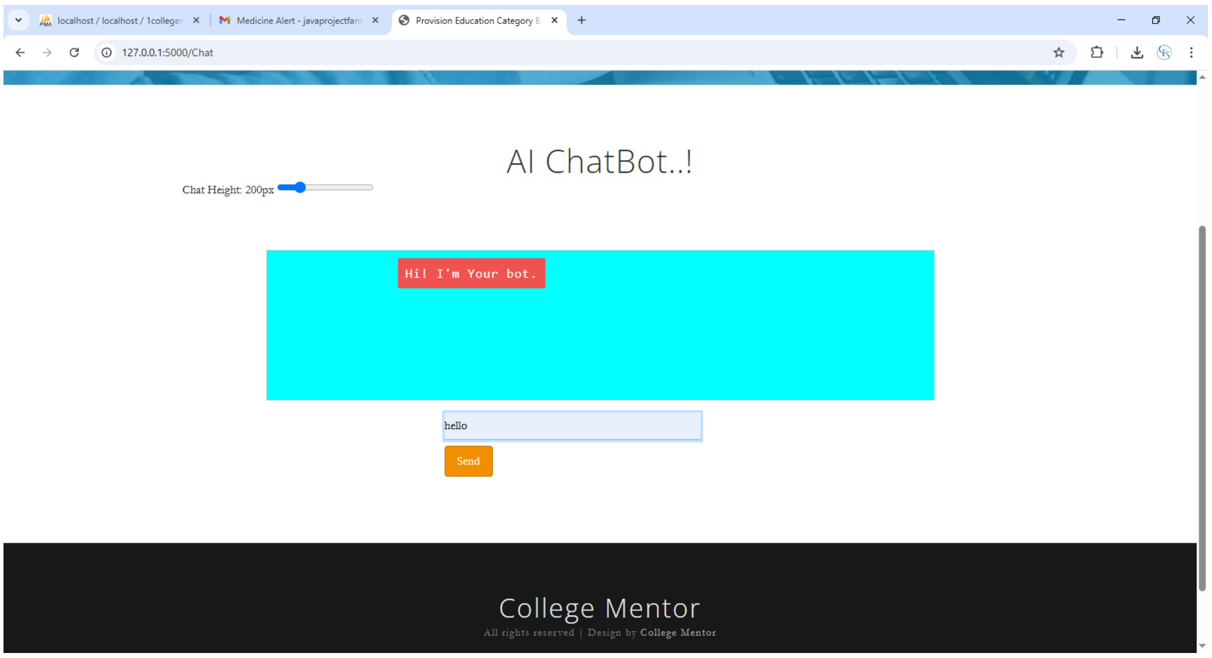
College Mentor
All rights reserved | Design by College Mentor


localhost / localhost / 1college
Medicine Alert - javaprojectfan
Provision Education Category
127.0.0.1:5000/Chat
Chat Height: 200px
hello
Send

AI ChatBot..!

Hi! I'm Your bot.

College Mentor
All rights reserved | Design by College Mentor


localhost / localhost / 1college
Medicine Alert - javaprojectfan
Provision Education Category
127.0.0.1:5000/Chat

AI ChatBot..!

Chat Height: 200px

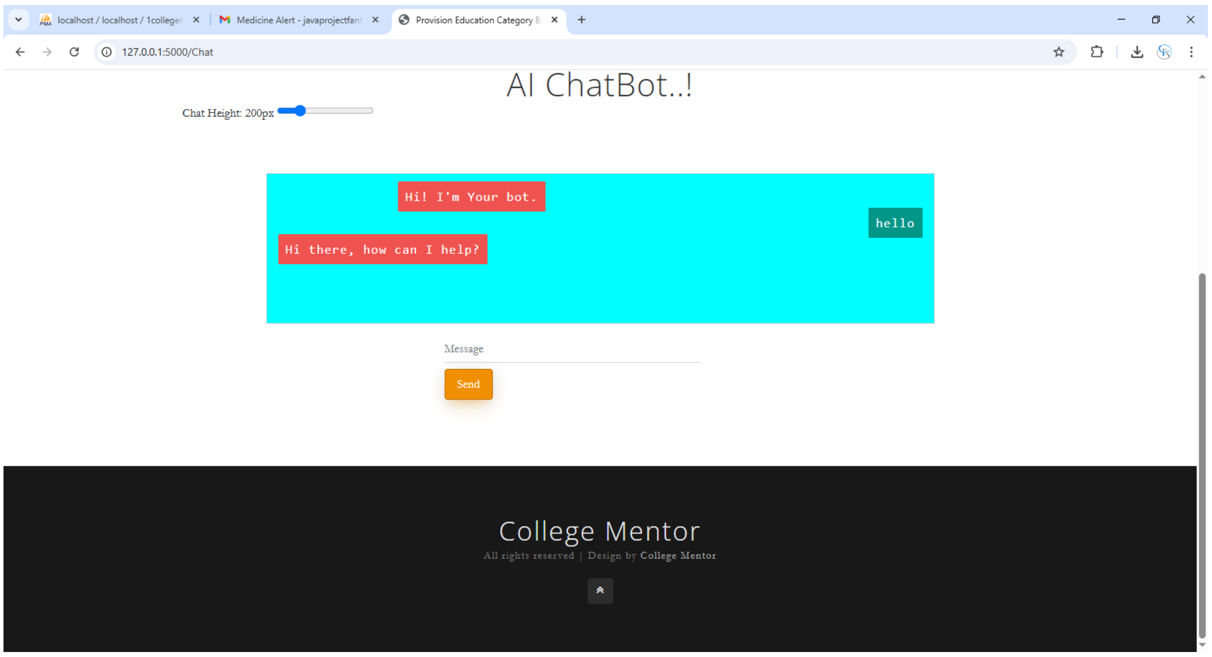
Hi! I'm Your bot.

hello

Send

College Mentor

All rights reserved | Design by College Mentor


localhost / localhost / 1college
Medicine Alert - javaprojectfan
Provision Education Category
127.0.0.1:5000/Chat

AI ChatBot..!

Chat Height: 200px

Hi! I'm Your bot.

hello

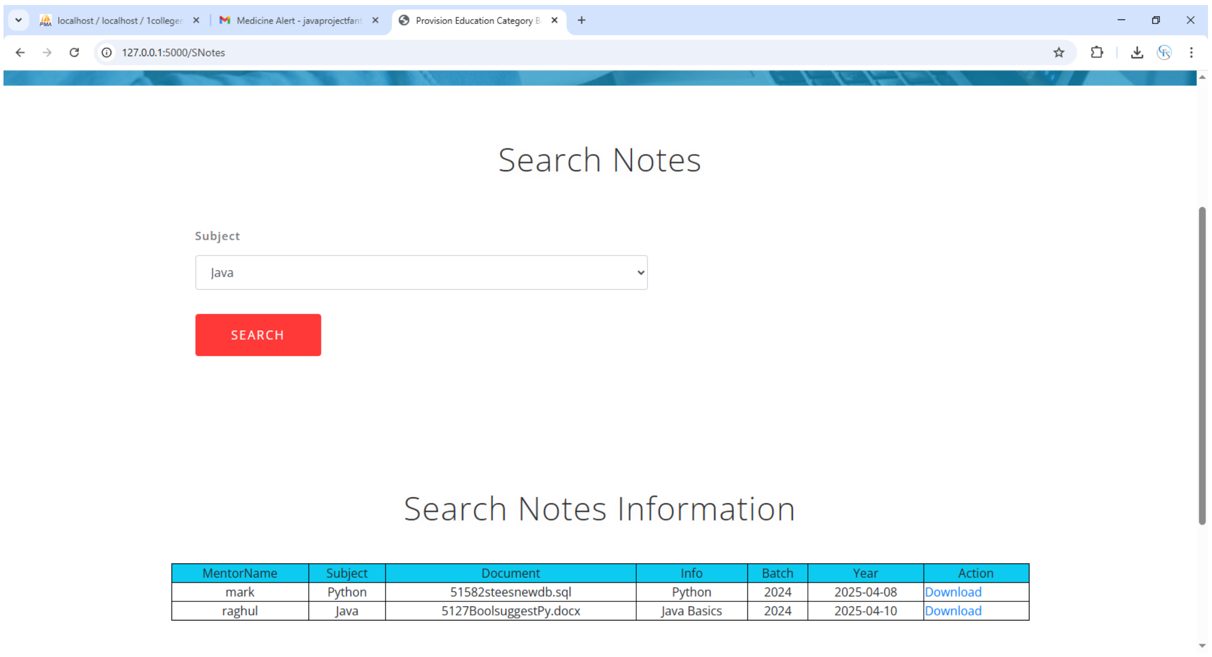
Hi there, how can I help?

Message

Send

College Mentor

All rights reserved | Design by College Mentor


localhost / localhost / 1college
Medicine Alert - javaprojectfan
Provision Education Category
127.0.0.1:5000/SNotes

Search Notes

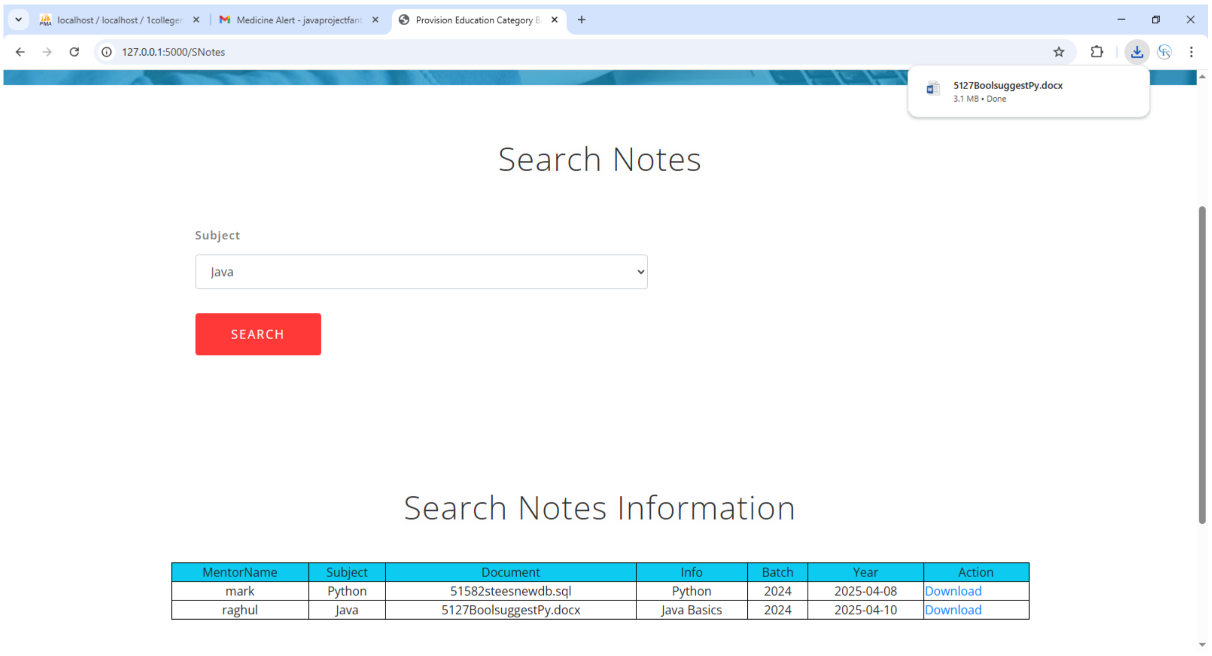
Subject

Java

SEARCH

Search Notes Information

MentorName	Subject	Document	Info	Batch	Year	Action
mark	Python	51582steesnewdb.sql	Python	2024	2025-04-08	Download
raghul	Java	5127BoolsuggestPy.docx	Java Basics	2024	2025-04-10	Download


localhost / localhost / 1college
Medicine Alert - javaprojectfan
Provision Education Category
127.0.0.1:5000/SNotes
5127BoolsuggestPy.docx
3.1 MB • Done

Search Notes

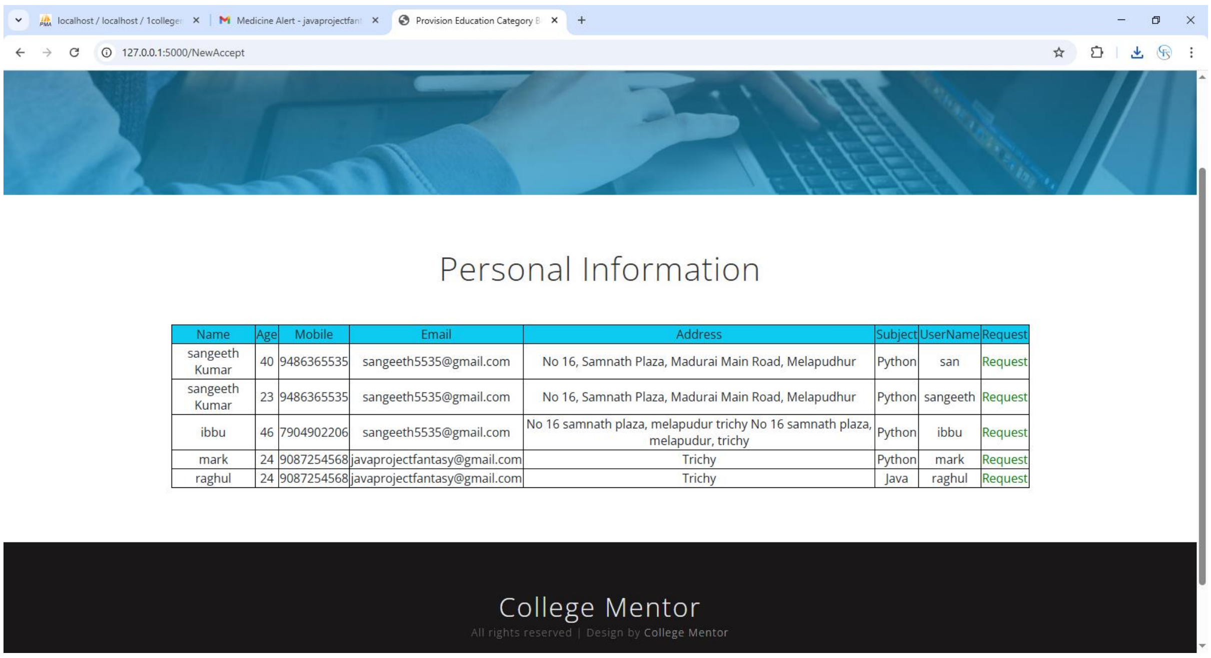
Subject

Java

SEARCH

Search Notes Information

MentorName	Subject	Document	Info	Batch	Year	Action
mark	Python	51582steesnewdb.sql	Python	2024	2025-04-08	Download
raghul	Java	5127BoolsuggestPy.docx	Java Basics	2024	2025-04-10	Download

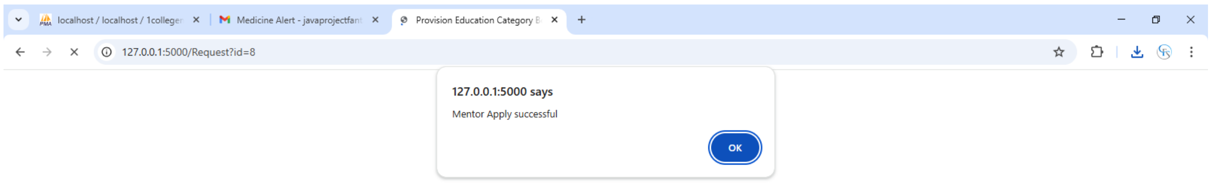

localhost / localhost / 1college
Medicine Alert - javaprojectfan
Provision Education Category
127.0.0.1:5000/NewAccept

Personal Information

Name	Age	Mobile	Email	Address	Subject	UserName	Request
sangeeth Kumar	40	9486365535	sangeeth5535@gmail.com	No 16, Samnath Plaza, Madurai Main Road, Melapudhur	Python	san	Request
sangeeth Kumar	23	9486365535	sangeeth5535@gmail.com	No 16, Samnath Plaza, Madurai Main Road, Melapudhur	Python	sangeeth	Request
ibbu	46	7904902206	sangeeth5535@gmail.com	No 16 samnath plaza, melapudur trichy No 16 samnath plaza, melapudur, trichy	Python	ibbu	Request
mark	24	9087254568	javaprojectfantasy@gmail.com	Trichy	Python	mark	Request
raghul	24	9087254568	javaprojectfantasy@gmail.com	Trichy	Java	raghul	Request

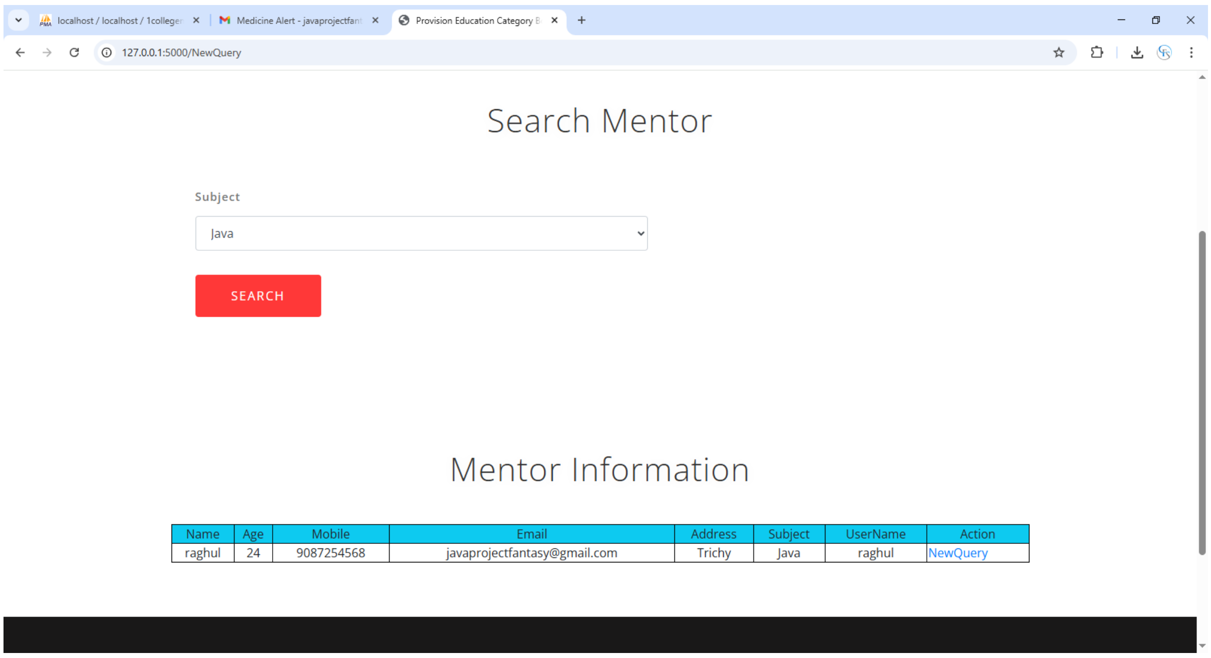
College Mentor

All rights reserved | Design by College Mentor


localhost / localhost / 1college
Medicine Alert - javaprojectfan
Provision Education Category
127.0.0.1:5000/Request?id=8
OK

127.0.0.1:5000 says

Mentor Apply successful



localhost / localhost / 1college
Medicine Alert - javaprojectfan
Provision Education Category
127.0.0.1:5000/NewQuery

Search Mentor

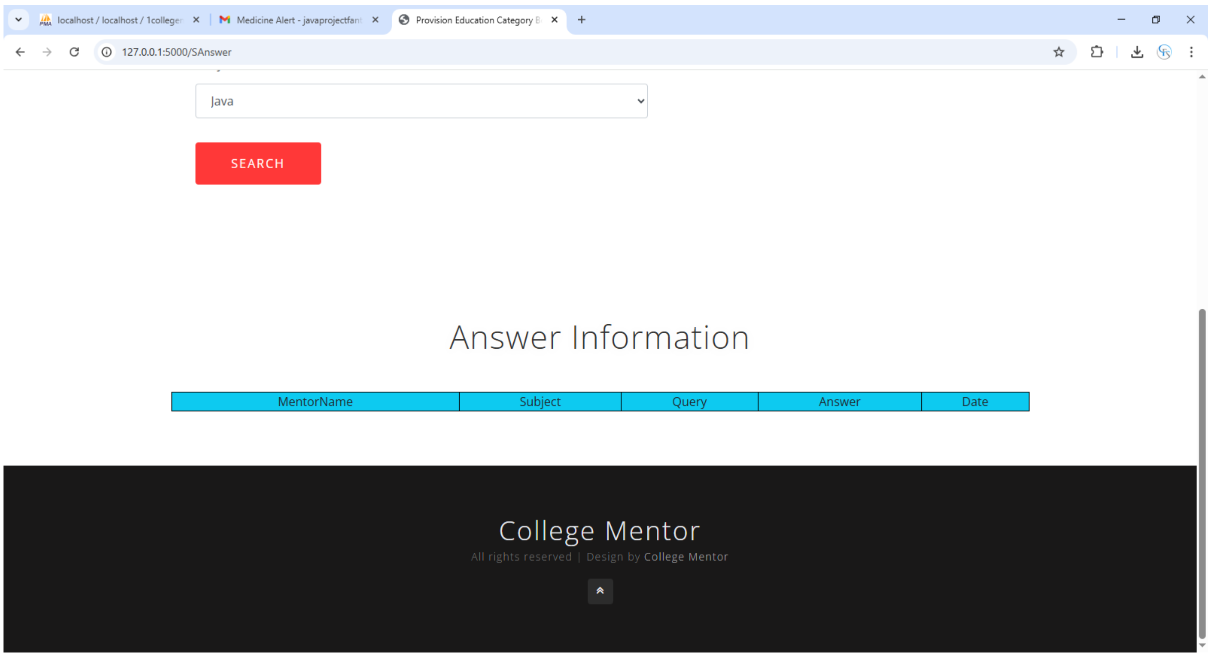
Subject

Java

SEARCH

Mentor Information

Name	Age	Mobile	Email	Address	Subject	UserName	Action
raghul	24	9087254568	javaprojectfantasy@gmail.com	Trichy	Java	raghul	NewQuery



localhost / localhost / 1college
Medicine Alert - javaprojectfan
Provision Education Category
127.0.0.1:5000/SAnswer

Java

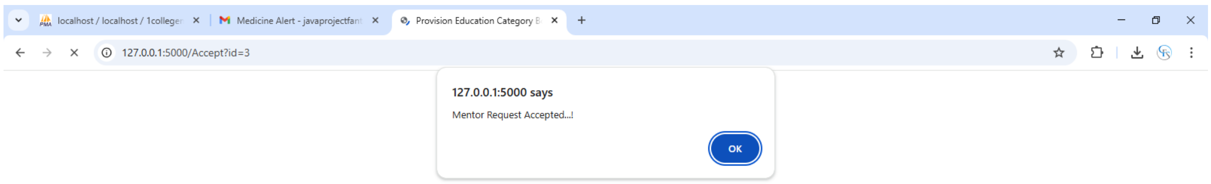
SEARCH

Answer Information

MentorName	Subject	Query	Answer	Date

College Mentor

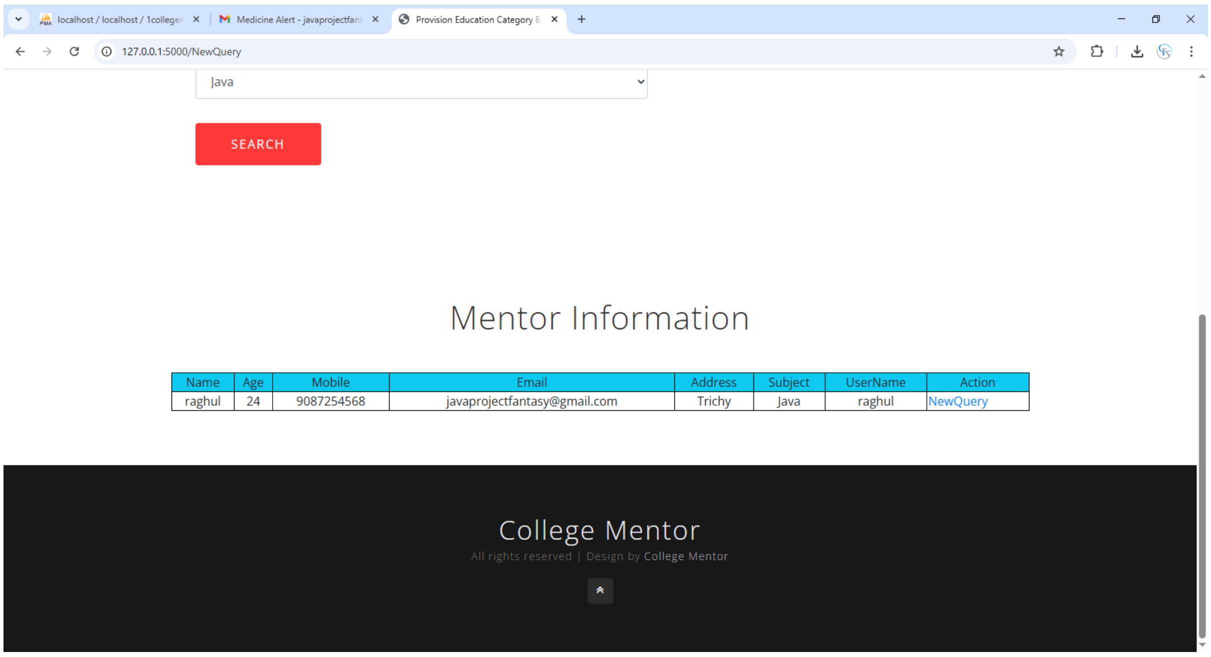
All rights reserved | Design by College Mentor



127.0.0.1:5000/Accept?id=3

127.0.0.1:5000 says

Mentor Request Accepted...!



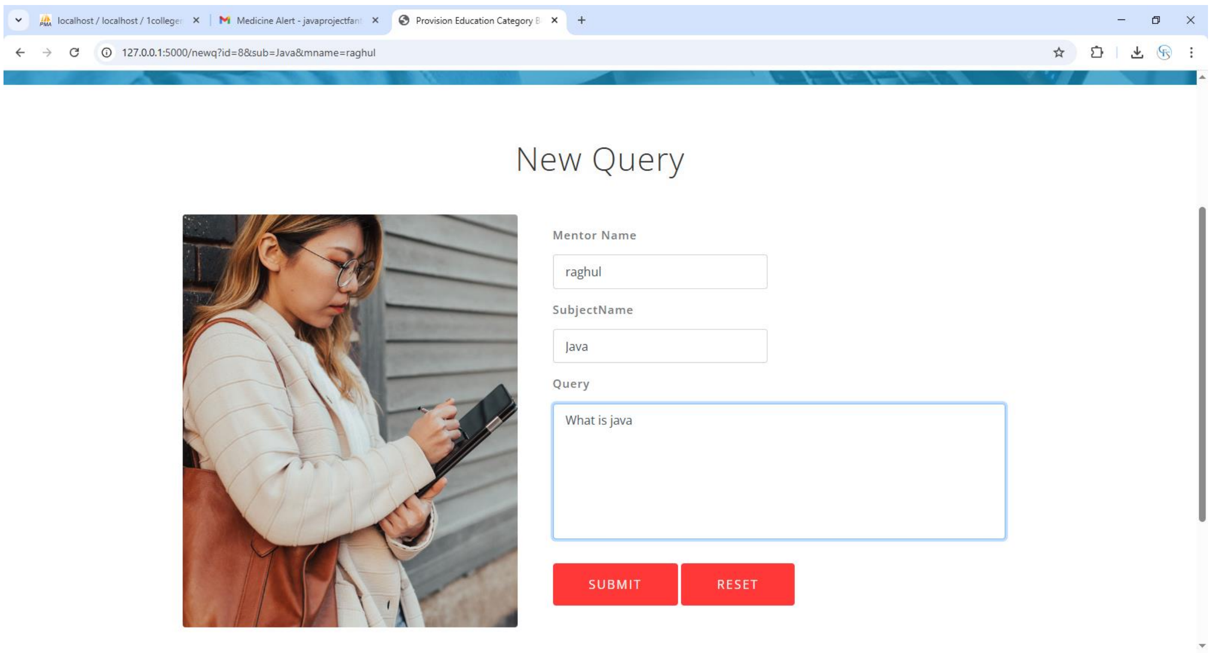
127.0.0.1:5000/NewQuery

Mentor Information

Name	Age	Mobile	Email	Address	Subject	UserName	Action
raghul	24	9087254568	javaprojectfantasy@gmail.com	Trichy	Java	raghul	NewQuery

College Mentor

All rights reserved | Design by College Mentor


localhost / localhost / 1college
Medicine Alert - javaprojectfan
Provision Education Category
127.0.0.1:5000/newq?id=8&sub=Java&mname=raghul

New Query

Mentor Name

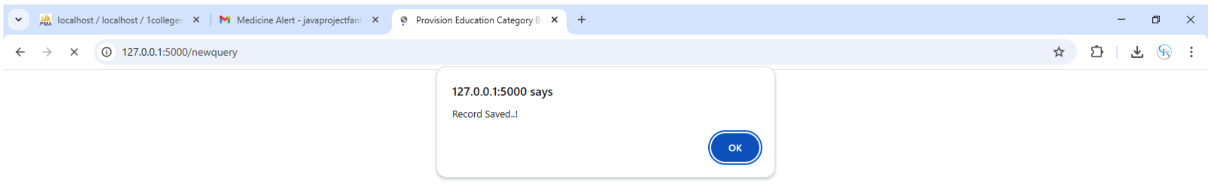
raghul

SubjectName

Java

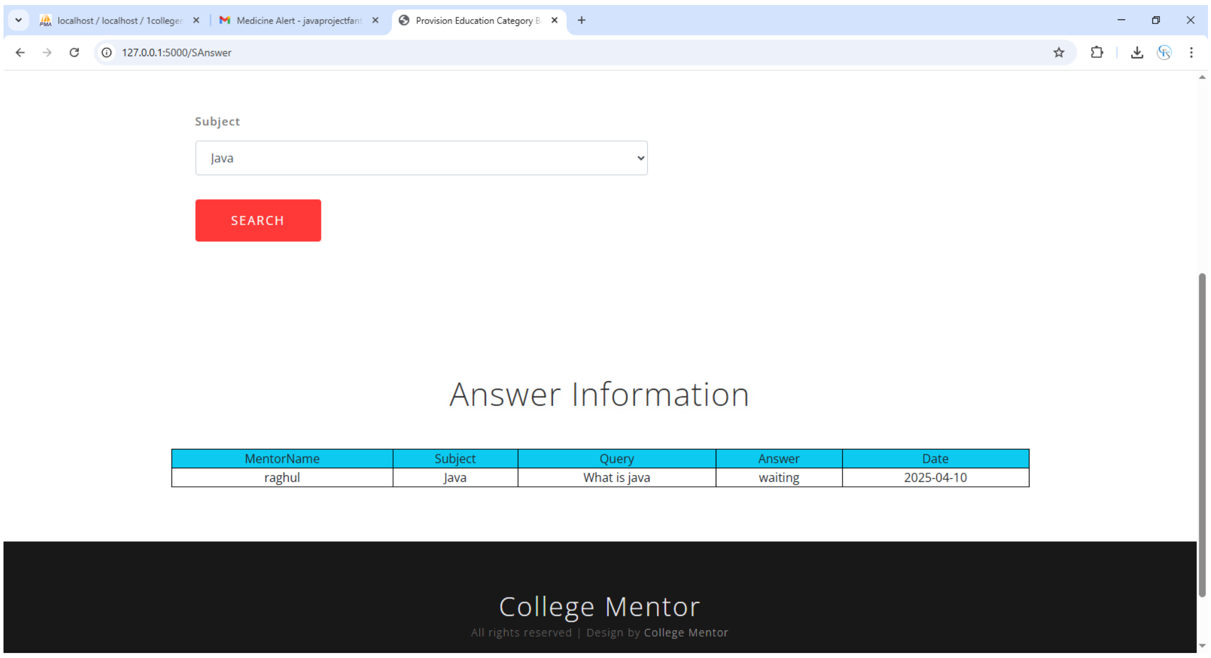
Query

What is java


localhost / localhost / 1college
Medicine Alert - javaprojectfan
Provision Education Category
127.0.0.1:5000/newquery

127.0.0.1:5000 says

Record Saved..!


localhost / localhost / 1college
Medicine Alert - javaprojectfan
Provision Education Category
127.0.0.1:5000/SAnswer
SEARCH

Subject

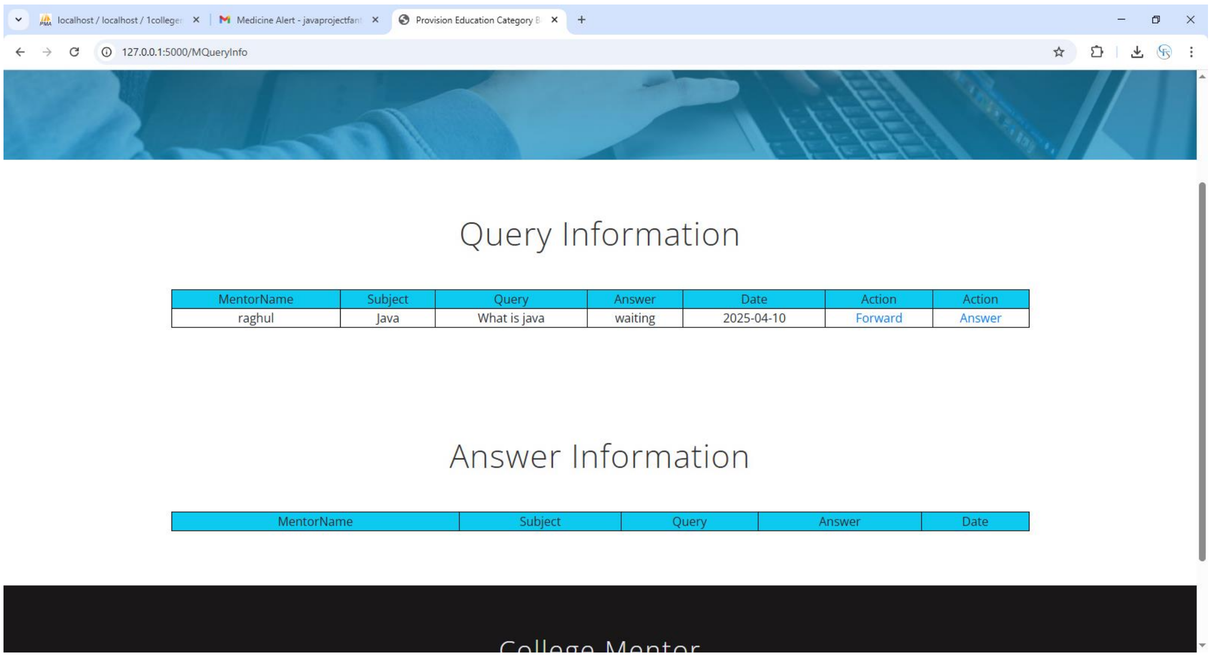
Java

Answer Information

MentorName	Subject	Query	Answer	Date
raghul	Java	What is java	waiting	2025-04-10

College Mentor

All rights reserved | Design by College Mentor


localhost / localhost / 1college
Medicine Alert - javaprojectfan
Provision Education Category
127.0.0.1:5000/MQueryInfo

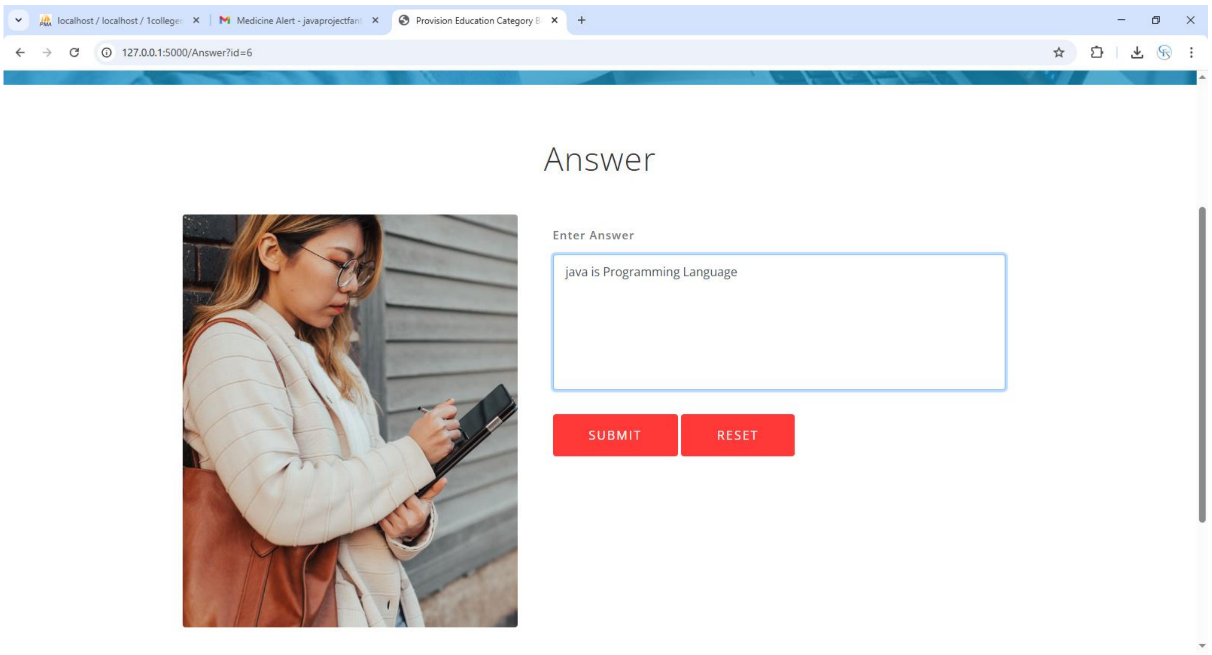
Query Information

MentorName	Subject	Query	Answer	Date	Action	Action
raghul	Java	What is java	waiting	2025-04-10	Forward	Answer

Answer Information

MentorName	Subject	Query	Answer	Date

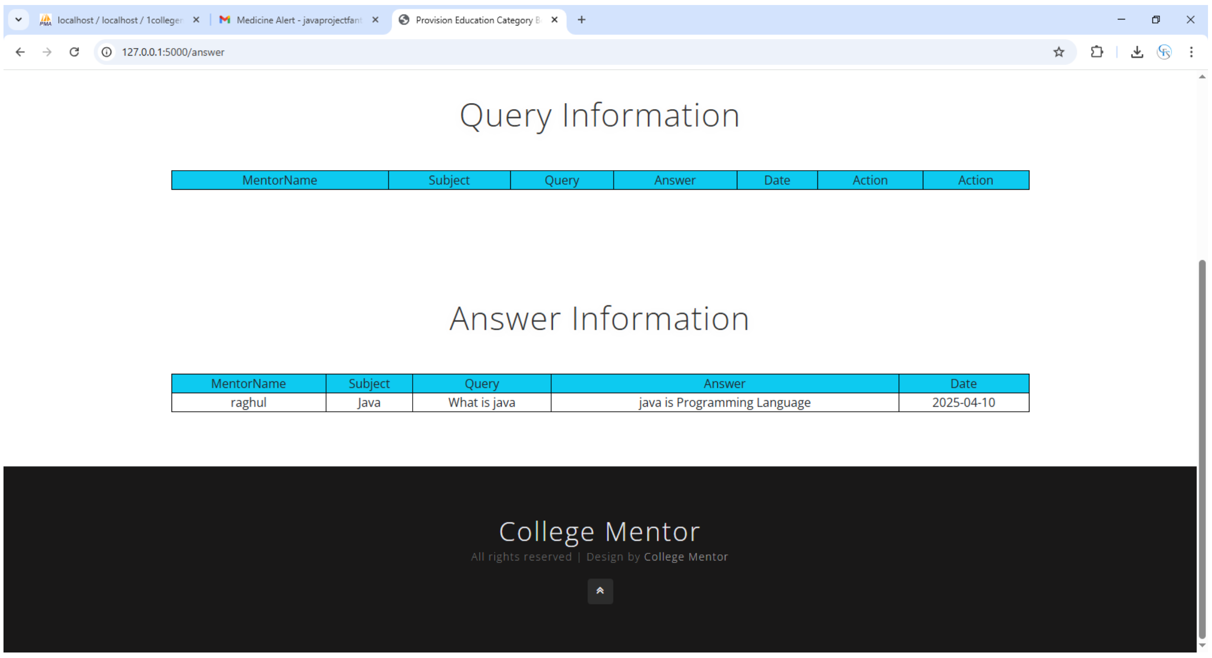
College Mentor


localhost / localhost / 1college
Medicine Alert - javaprojectfan
Provision Education Category
127.0.0.1:5000/Answer?id=6

Answer

Enter Answer

java is Programming Language


localhost / localhost / 1college
Medicine Alert - javaprojectfan
Provision Education Category
127.0.0.1:5000/answer

Query Information

MentorName	Subject	Query	Answer	Date	Action	Action

Answer Information

MentorName	Subject	Query	Answer	Date
raghul	Java	What is java	java is Programming Language	2025-04-10

College Mentor

All rights reserved | Design by College Mentor

CHAPTER 10

REFERENCES

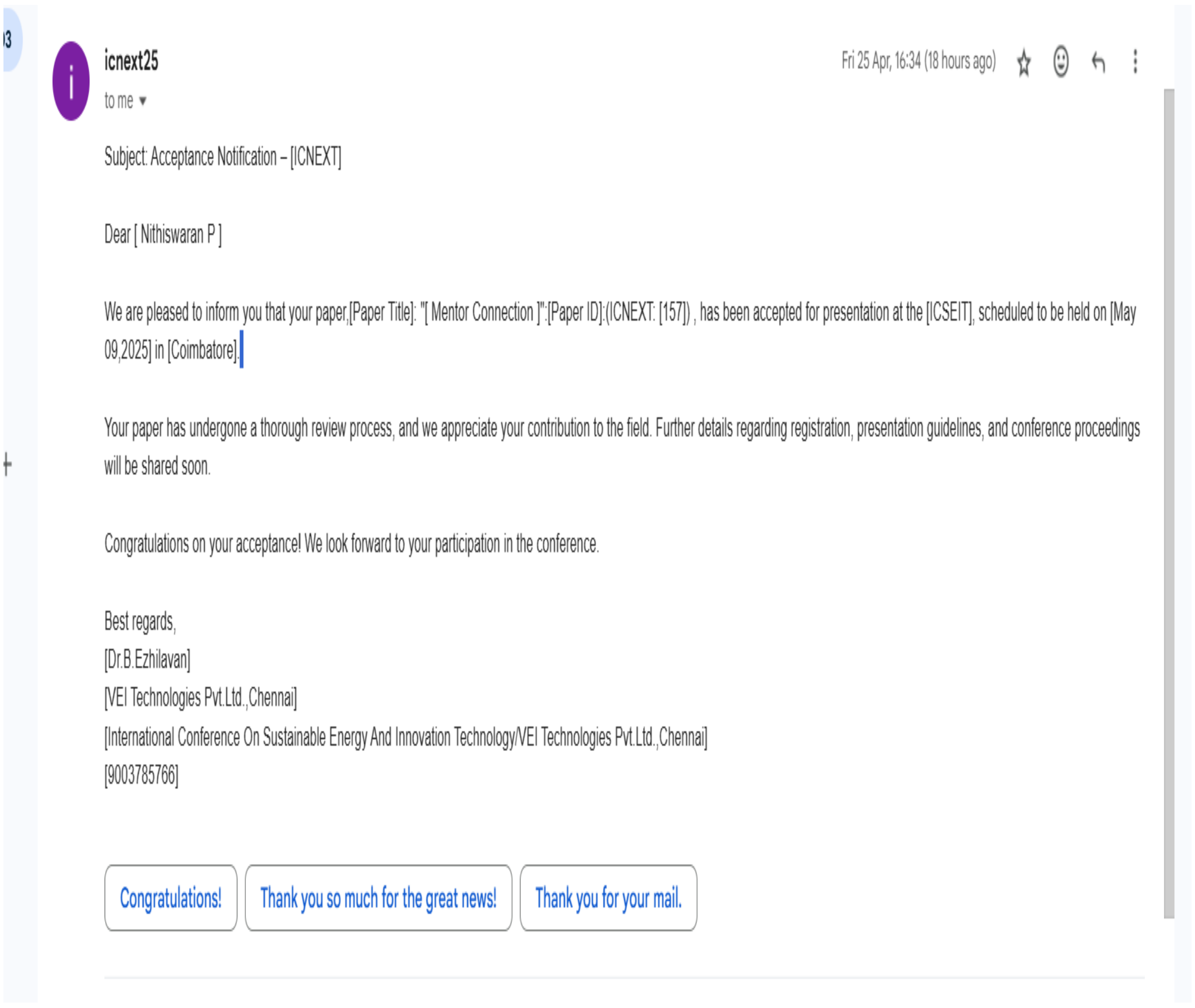
BOOK REFERENCES

1. Heinold, Brian. "A practical introduction to Python programming." (2021).
2. Kneusel, Ronald T. Practical deep learning: A Python-based introduction. No Starch Press, 2021.
3. Dhruv, Akshit J., Reema Patel, and Nishant Doshi. "Python: the most advanced programming language for computer science applications." Science and Technology Publications, Lda (2021): 292-299.
4. Sundnes, Joakim. Introduction to scientific programming with Python. Springer Nature, 2020.
5. Hill, Christian. Learning scientific programming with Python. Cambridge University Press, 2020.

WEBSITE REFERENCES

1. https://medium.com/javarevisited/10-free-python-tutorials-and-courses-from-google-microsoft-and-coursera-for-beginners-96b9ad20b4e6
2. https://www.bestcolleges.com/bootcamps/guides/learn-python-free/
3. https://www.programiz.com/python-programming
4. https://realpython.com/
5. https://www.codecademy.com/learn/learn-python

CONFERENCE ACCEPTED


icnext25

to me

Fri 25 Apr, 16:34 (18 hours ago)

Subject: Acceptance Notification – [ICNEXT]

Dear [Nithiswaran P]

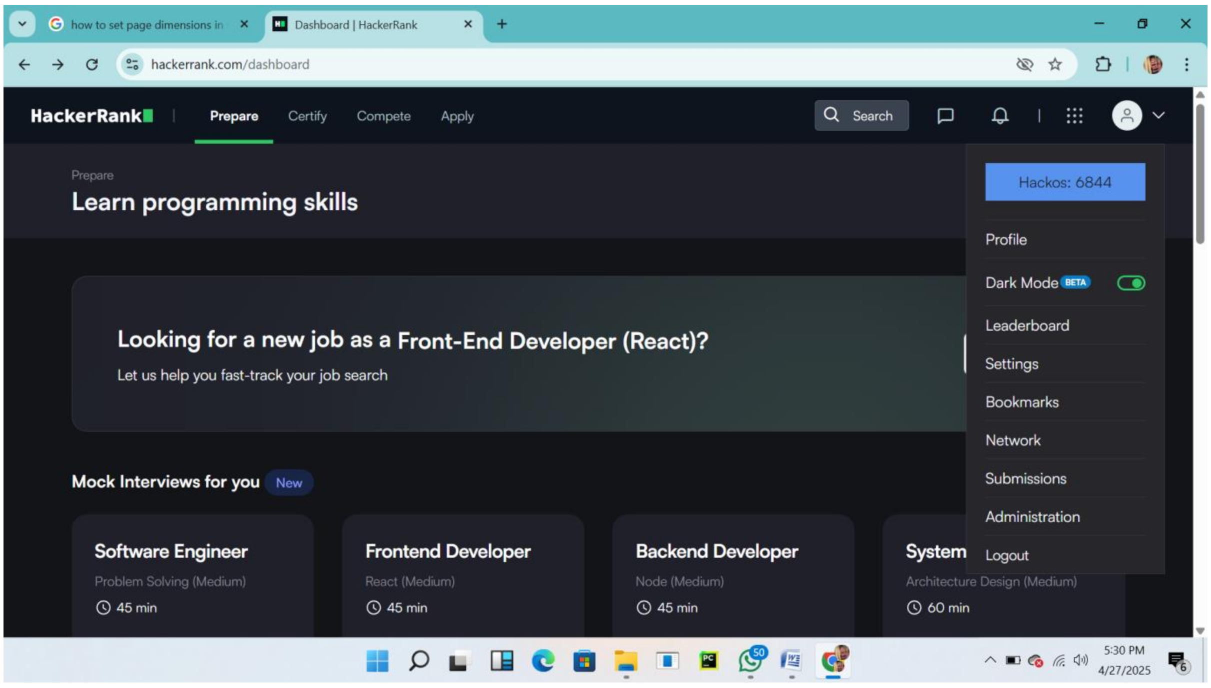
We are pleased to inform you that your paper,[Paper Title]: "[Mentor Connection]":[Paper ID]:(ICNEXT: [157]) , has been accepted for presentation at the [ICSEIT], scheduled to be held on [May 09,2025] in [Coimbatore].

Your paper has undergone a thorough review process, and we appreciate your contribution to the field. Further details regarding registration, presentation guidelines, and conference proceedings will be shared soon.

Congratulations on your acceptance! We look forward to your participation in the conference.

Best regards,
[Dr.B.Ezhilavan]
[VEI Technologies Pvt.Ltd.,Chennai]
[International Conference On Sustainable Energy And Innovation Technology/VEI Technologies Pvt.Ltd.,Chennai]
[9003785766]

HACHERRANK SCORE



Prepare

Learn programming skills

Looking for a new job as a Front-End Developer (React)?

Let us help you fast-track your job search

Mock Interviews for you New

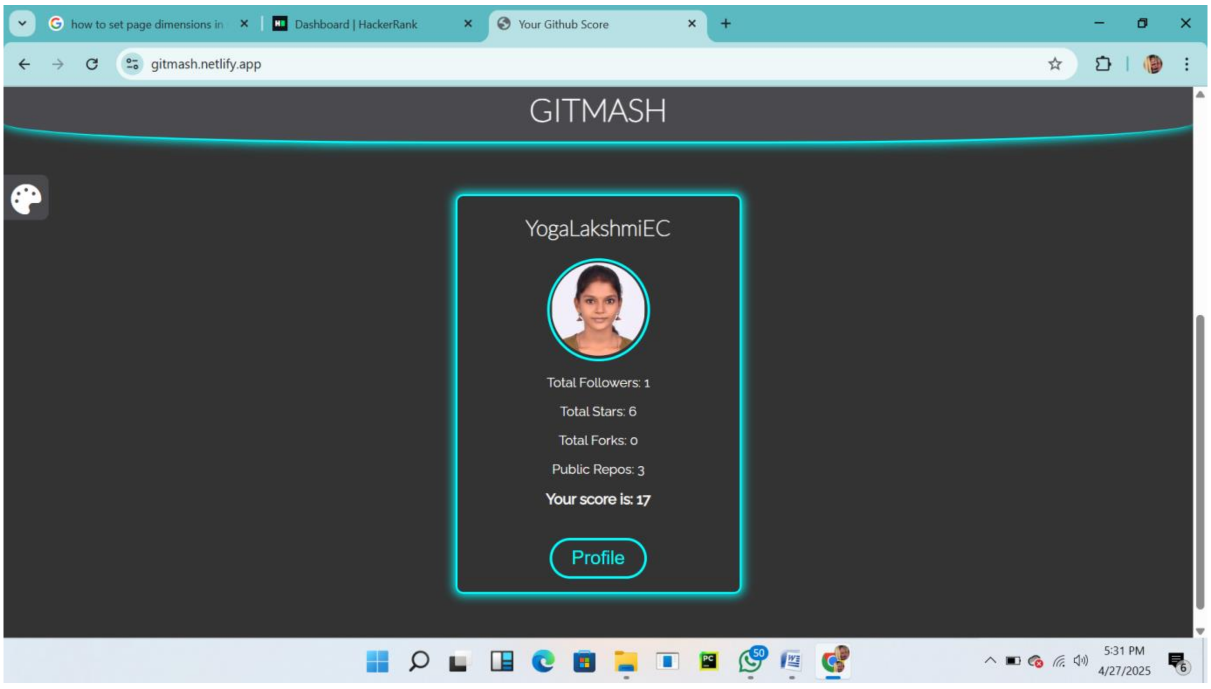
Software Engineer
Problem Solving (Medium)
45 min

Frontend Developer
React (Medium)
45 min

Backend Developer
Node (Medium)
45 min

System
Architecture Design (Medium)
60 min

GITHUB SCORE



YogaLakshmiEC

Total Followers: 1

Total Stars: 6

Total Forks: 0

Public Repos: 3

Your score is: 17