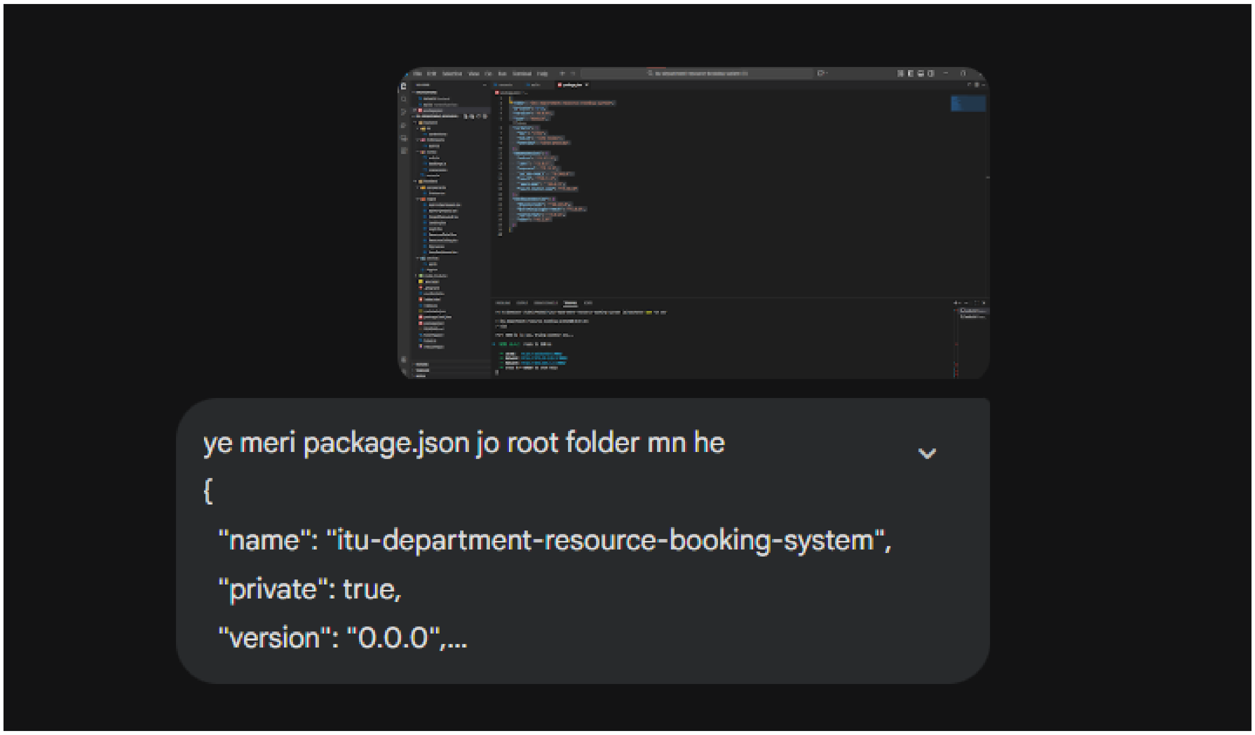
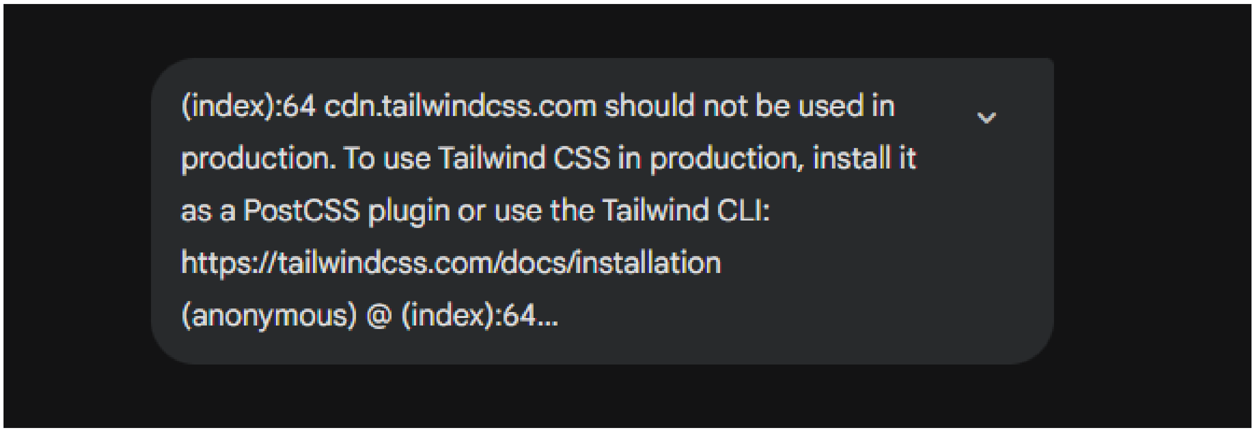
PROMPTS LOG

Code:

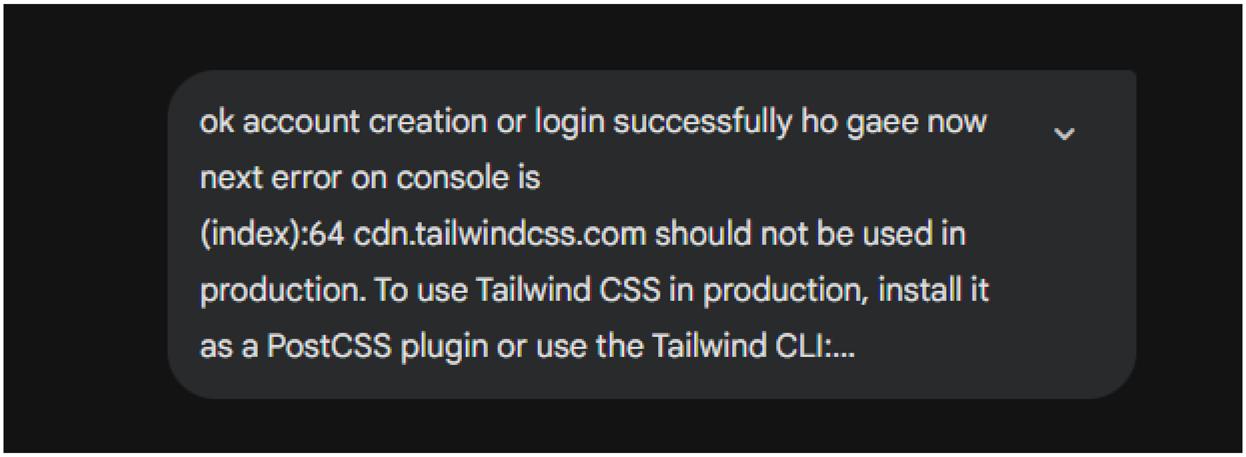


ye meri package.json jo root folder mn he

```
{
  "name": "itu-department-resource-booking-system",
  "private": true,
  "version": "0.0.0",...
```

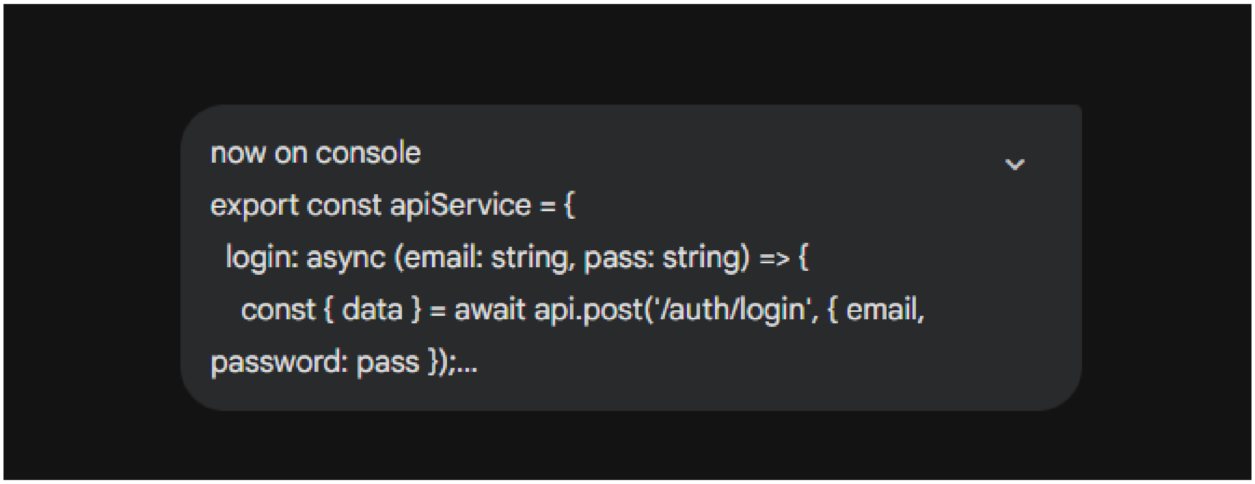


```
(index):64 cdn.tailwindcss.com should not be used in production. To use Tailwind CSS in production, install it as a PostCSS plugin or use the Tailwind CLI: https://tailwindcss.com/docs/installation
(anonymous) @ (index):64...
```



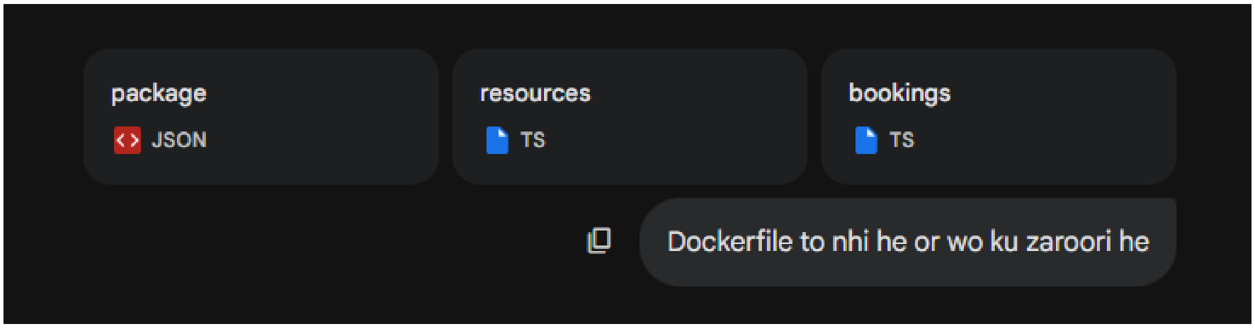
ok account creation or login successfully ho gaee now next error on console is

(index):64 cdn.tailwindcss.com should not be used in production. To use Tailwind CSS in production, install it as a PostCSS plugin or use the Tailwind CLI:...

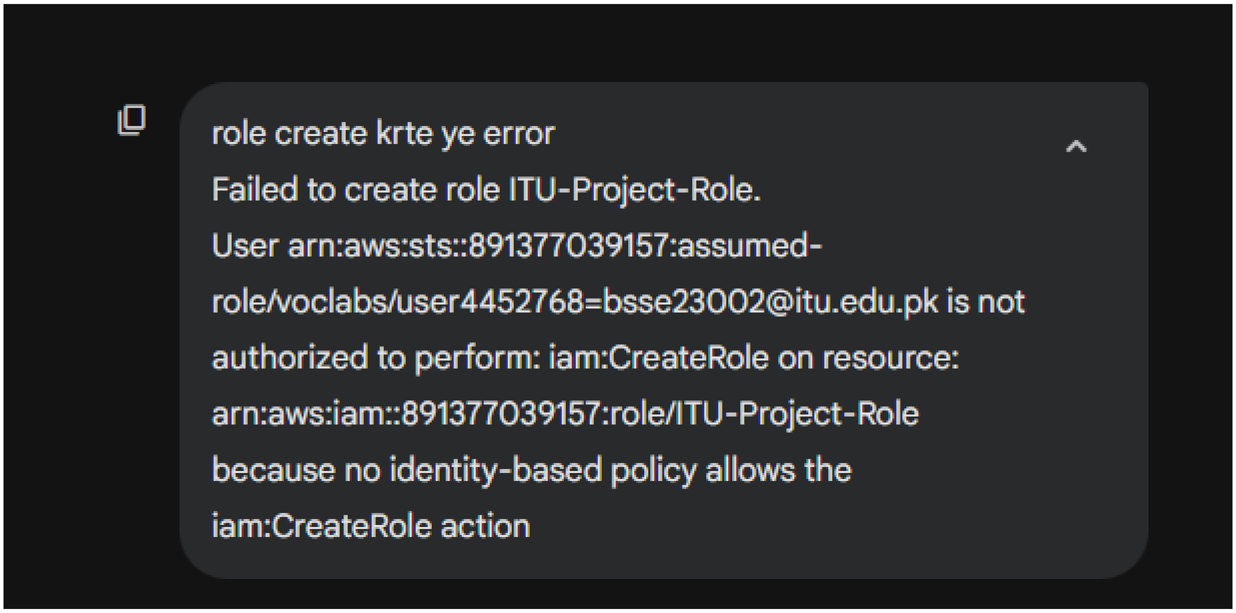


now on console

```
export const apiService = {
  login: async (email: string, pass: string) => {
    const { data } = await api.post('/auth/login', { email, password: pass });...
```

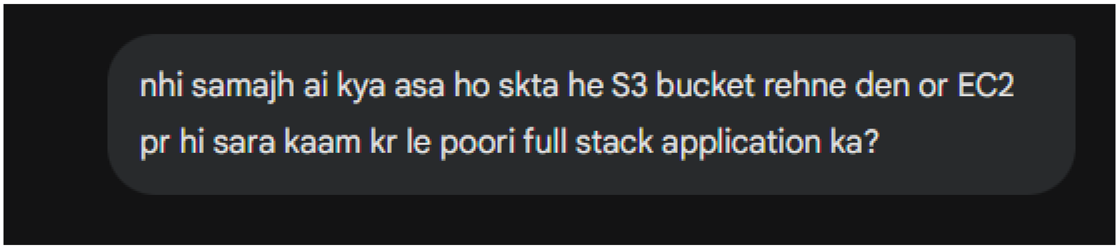


Dockerfile to nhi he or wo ku zaroori he

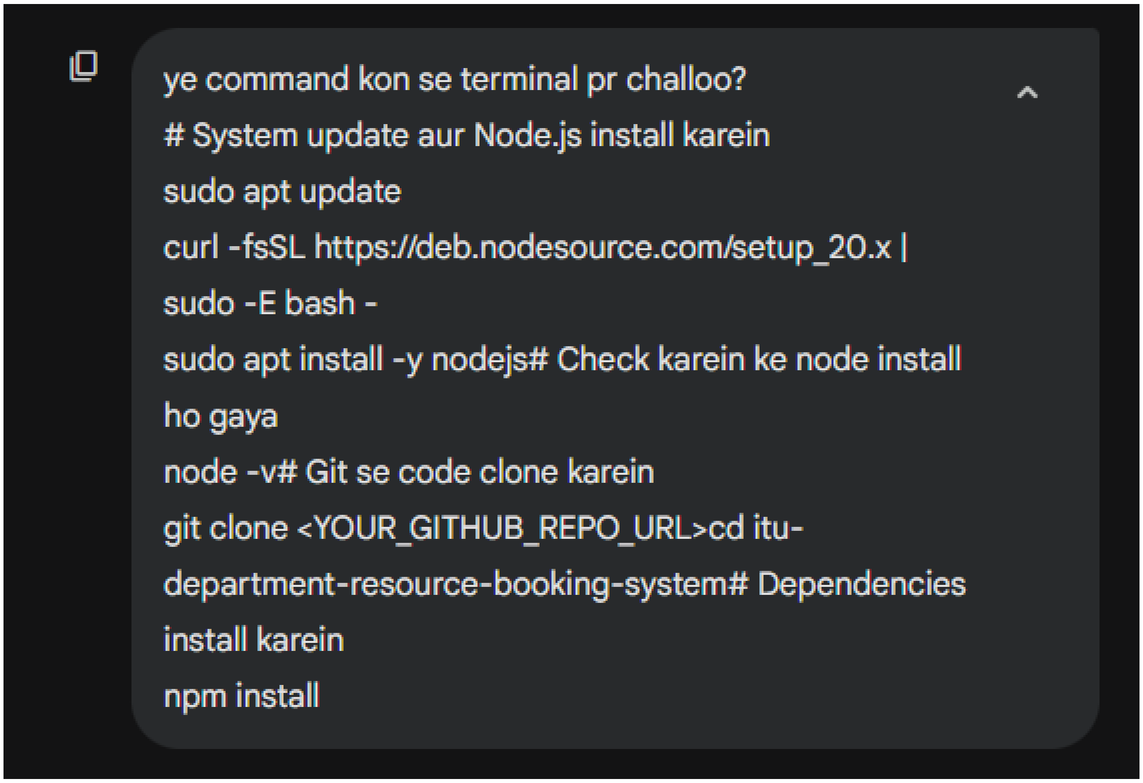


role create krte ye error

```
Failed to create role ITU-Project-Role.
User arn:aws:sts::891377039157:assumed-
role/voclabs/user4452768=bsse23002@itu.edu.pk is not
authorized to perform: iam:CreateRole on resource:
arn:aws:iam::891377039157:role/ITU-Project-Role
because no identity-based policy allows the
iam:CreateRole action
```

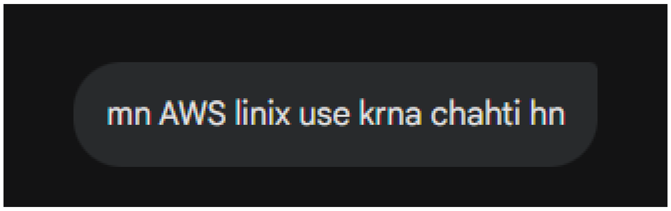


nhi samajh ai kya asa ho skta he S3 bucket rehne den or EC2 pr hi sara kaam kr le poori full stack application ka?

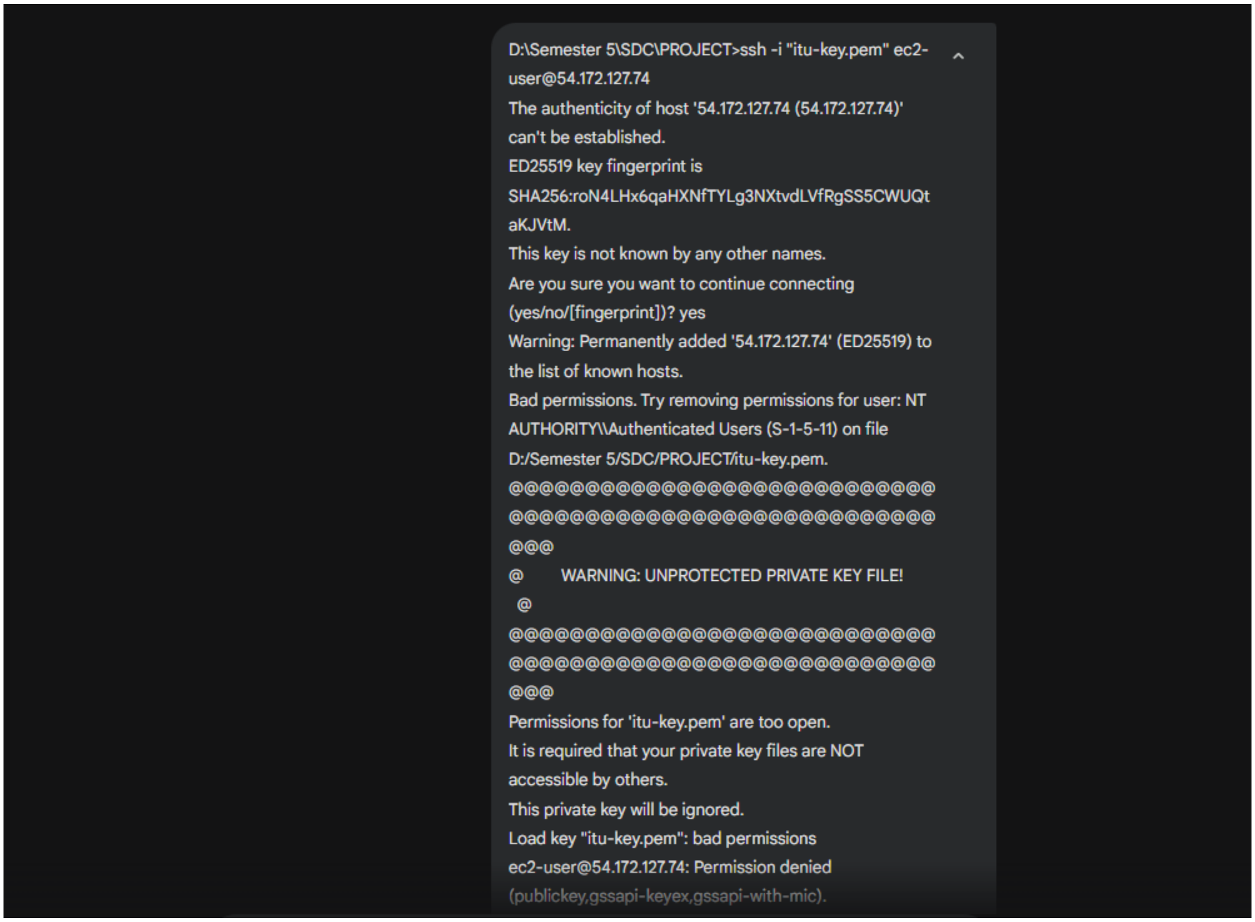


ye command kon se terminal pr challoo?

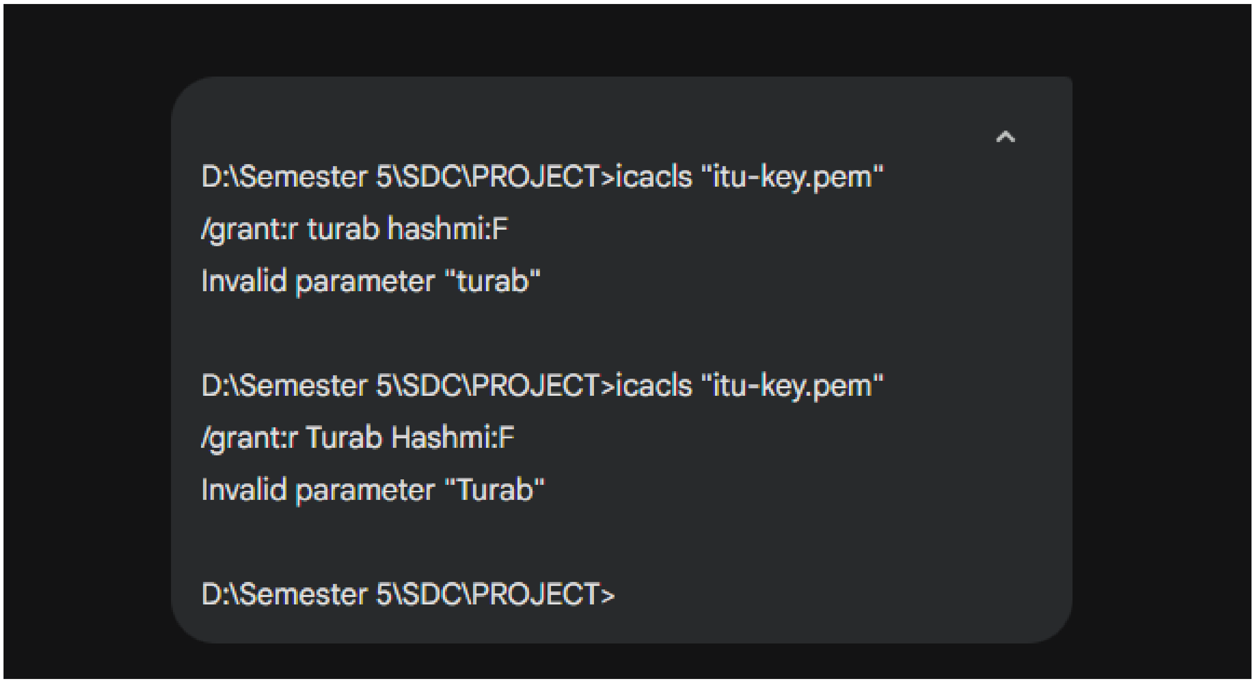
```
# System update aur Node.js install karein
sudo apt update
curl -fsSL https://deb.nodesource.com/setup_20.x |
sudo -E bash -
sudo apt install -y nodejs# Check karein ke node install
ho gaya
node -v# Git se code clone karein
git clone <YOUR_GITHUB_REPO_URL>cd itu-
department-resource-booking-system# Dependencies
install karein
npm install
```



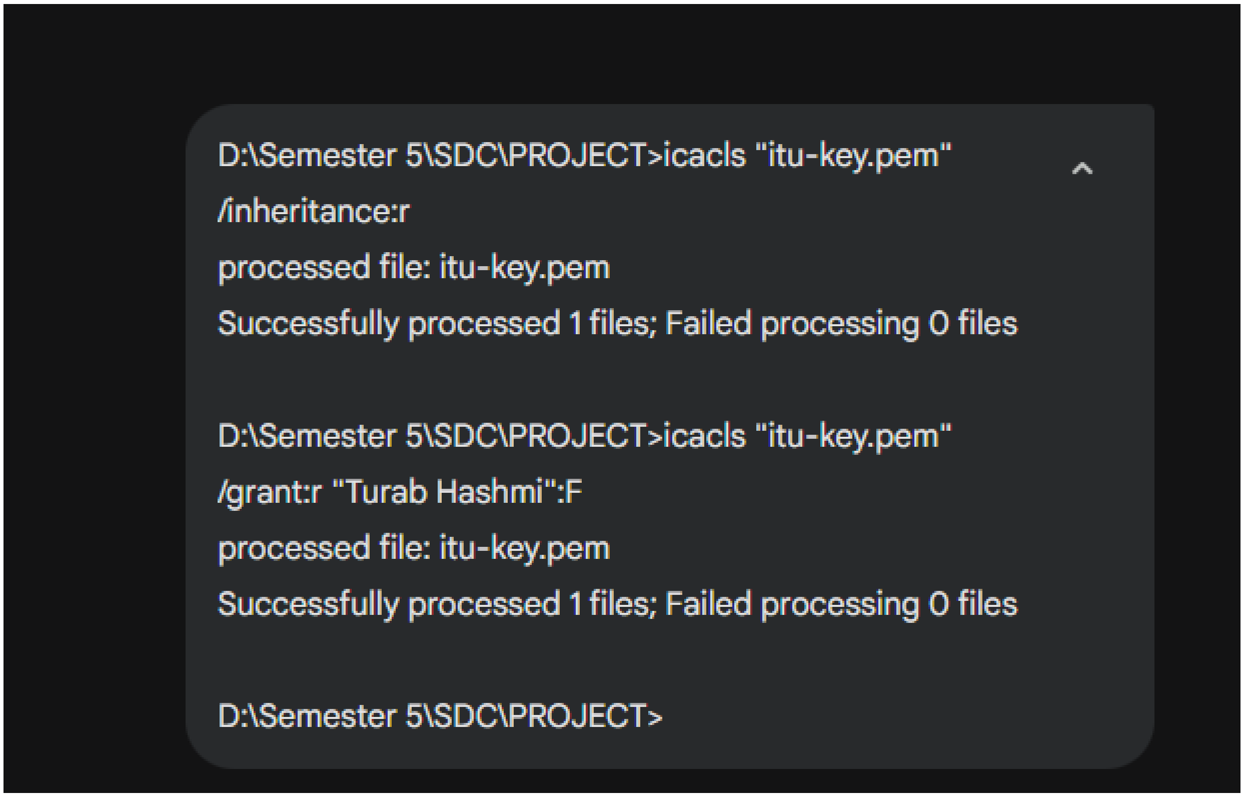
mn AWS linix use krna chahti hn



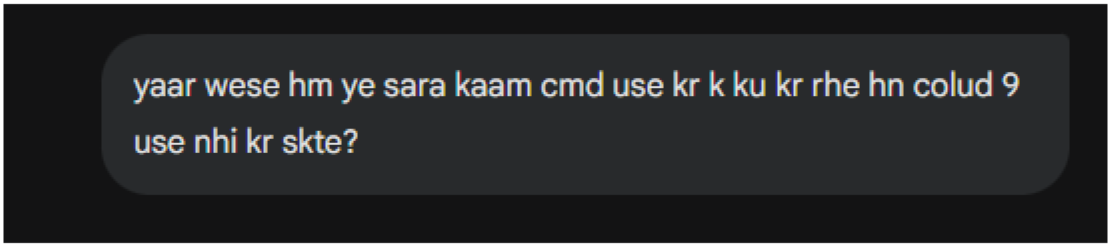
```
D:\Semester 5\SDC\PROJECT>ssh -i "itu-key.pem" ec2-
user@54.172.127.74
The authenticity of host '54.172.127.74 (54.172.127.74)'
can't be established.
ED25519 key fingerprint is
SHA256:roN4LHx6qaHXNfTYLg3NXtvdLVfRgSS5CWUQt
aKJVtM.
This key is not known by any other names.
Are you sure you want to continue connecting
(yes/no/[fingerprint])? yes
Warning: Permanently added '54.172.127.74' (ED25519) to
the list of known hosts.
Bad permissions. Try removing permissions for user: NT
AUTHORITY\\Authenticated Users (S-1-5-11) on file
D:/Semester 5/SDC/PROJECT/itu-key.pem.
@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@
@       WARNING: UNPROTECTED PRIVATE KEY FILE!
 @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@
Permissions for 'itu-key.pem' are too open.
It is required that your private key files are NOT
accessible by others.
This private key will be ignored.
Load key "itu-key.pem": bad permissions
ec2-user@54.172.127.74: Permission denied
(publickey,gssapi-keyex,gssapi-with-mic).
```



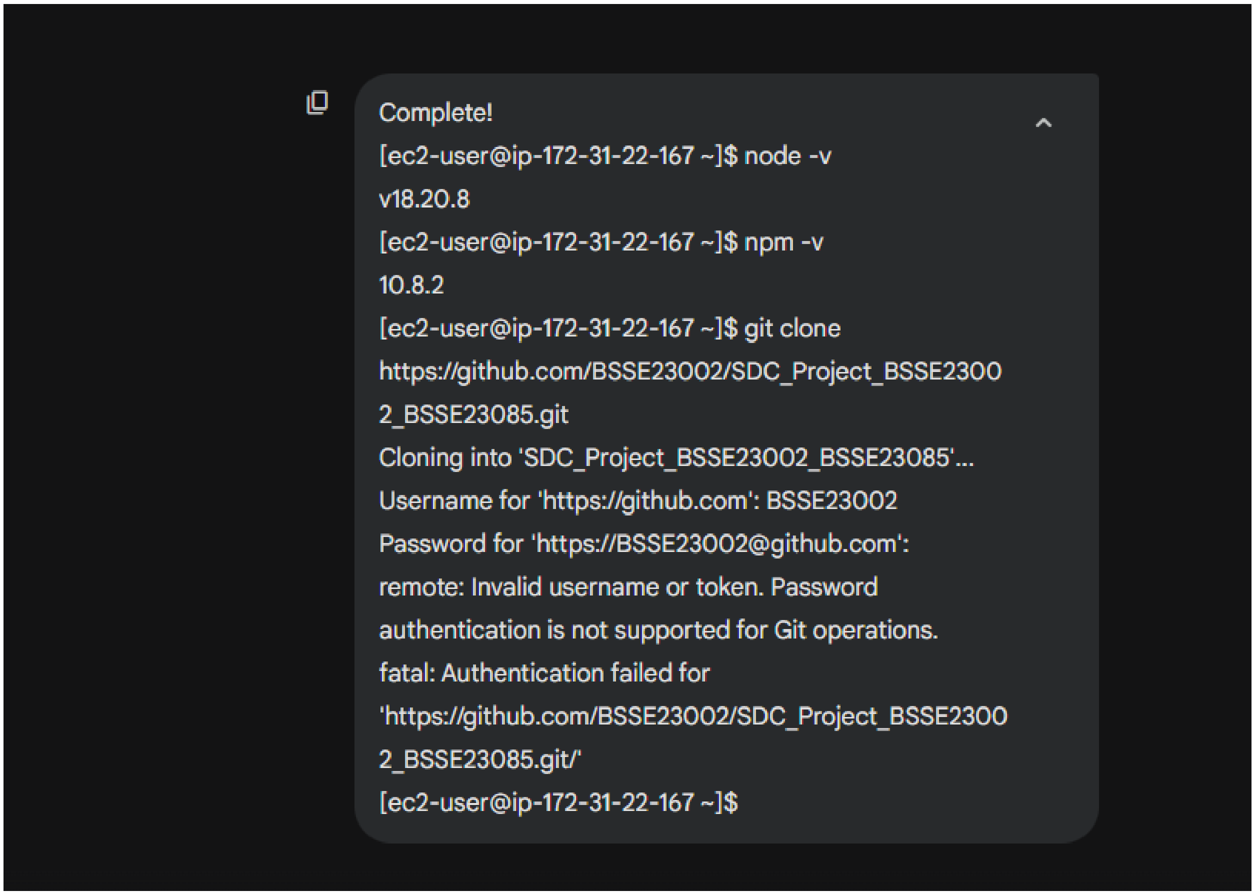
```
D:\Semester 5\SDC\PROJECT>icacls "itu-key.pem"
/grant:r turab hashmi:F
Invalid parameter "turab"

D:\Semester 5\SDC\PROJECT>icacls "itu-key.pem"
/grant:r Turab Hashmi:F
Invalid parameter "Turab"

D:\Semester 5\SDC\PROJECT>
```



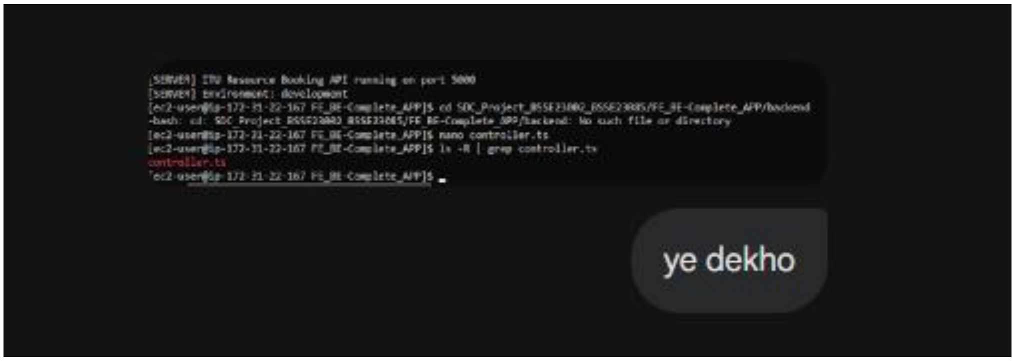
```
D:\Semester 5\SDC\PROJECT>icacls "itu-key.pem"
/inheritance:r
processed file: itu-key.pem
Successfully processed 1 files; Failed processing 0 files

D:\Semester 5\SDC\PROJECT>icacls "itu-key.pem"
/grant:r "Turab Hashmi":F
processed file: itu-key.pem
Successfully processed 1 files; Failed processing 0 files

D:\Semester 5\SDC\PROJECT>
```



yaar wese hm ye sara kaam cmd use kr k ku kr rhe hn colud 9 use nhi kr skte?

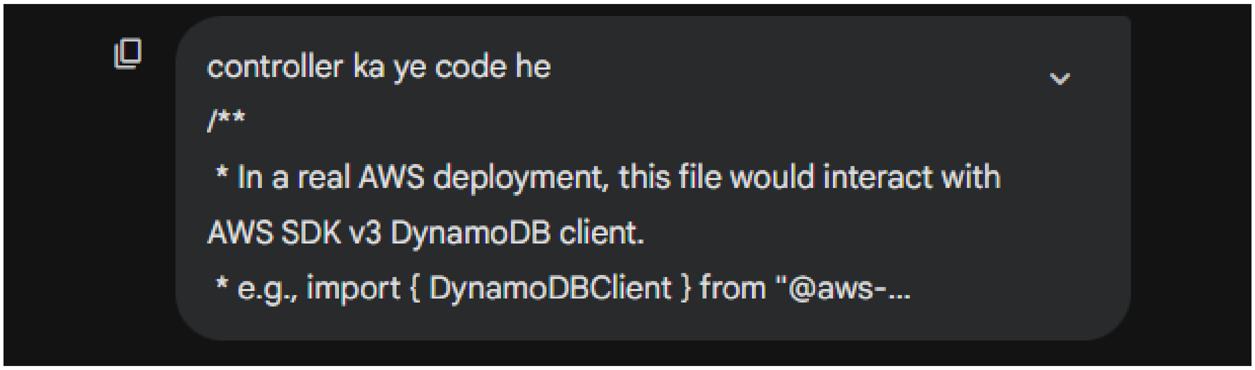


```
Complete!
[ec2-user@ip-172-31-22-167 ~]$ node -v
v18.20.8
[ec2-user@ip-172-31-22-167 ~]$ npm -v
10.8.2
[ec2-user@ip-172-31-22-167 ~]$ git clone
https://github.com/BSSE23002/SDC_Project_BSSE2300
2_BSSE23085.git
Cloning into 'SDC_Project_BSSE23002_BSSE23085'...
Username for 'https://github.com': BSSE23002
Password for 'https://BSSE23002@github.com':
remote: Invalid username or token. Password
authentication is not supported for Git operations.
fatal: Authentication failed for
'https://github.com/BSSE23002/SDC_Project_BSSE2300
2_BSSE23085.git/'
[ec2-user@ip-172-31-22-167 ~]$
```



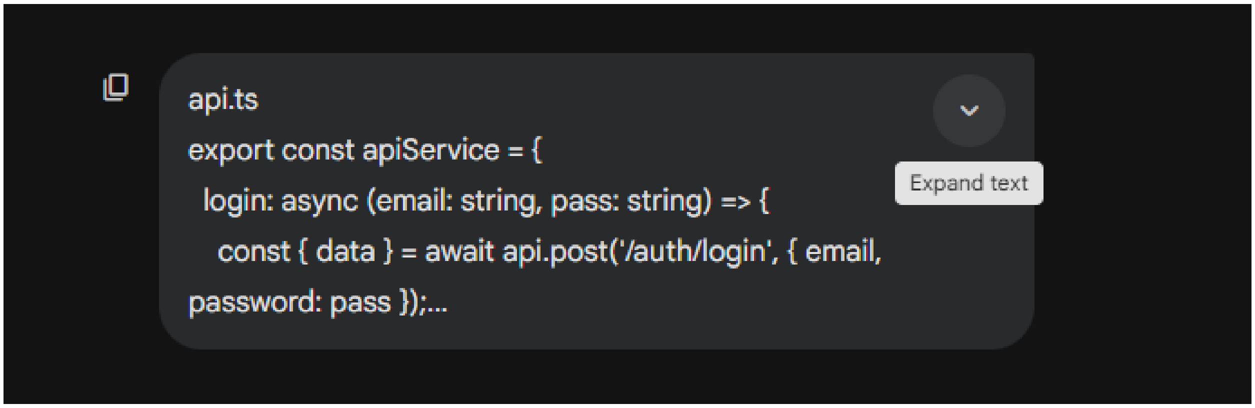
```
[SERVER] ITU Resource Booking API running on port 5000
[SERVER] Environment: development
[ec2-user@ip-172-31-22-167 FE_BE-Complete_APP]$ cd SDC_Project_BSSE23002_BSSE23085/FE_BE-Complete_APP/backend
-bash: cd: SDC_Project_BSSE23002_BSSE23085/FE_BE-Complete_APP/backend: No such file or directory
[ec2-user@ip-172-31-22-167 FE_BE-Complete_APP]$ nano controller.ts
[ec2-user@ip-172-31-22-167 FE_BE-Complete_APP]$ ls -R | grep controller.ts
controller.ts
[ec2-user@ip-172-31-22-167 FE_BE-Complete_APP]$ 
```

ye dekho



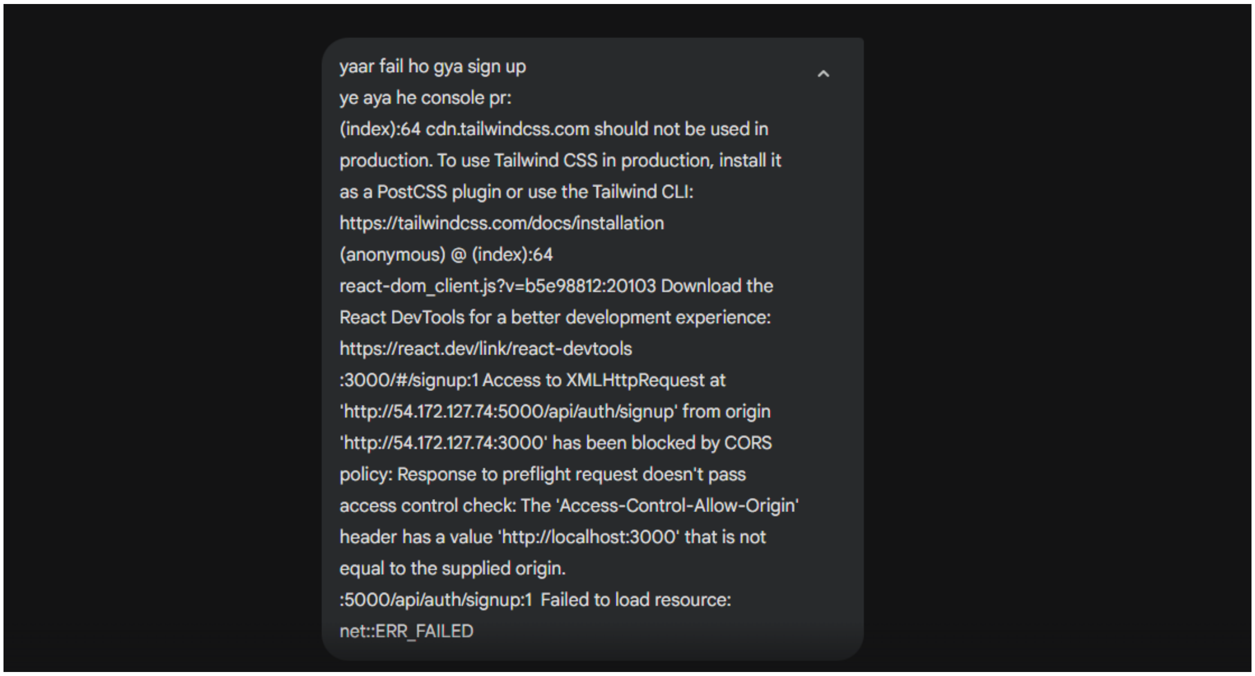
controller ka ye code he

```
/**
 * In a real AWS deployment, this file would interact with
AWS SDK v3 DynamoDB client.
 * e.g., import { DynamoDBClient } from "@aws-...
```



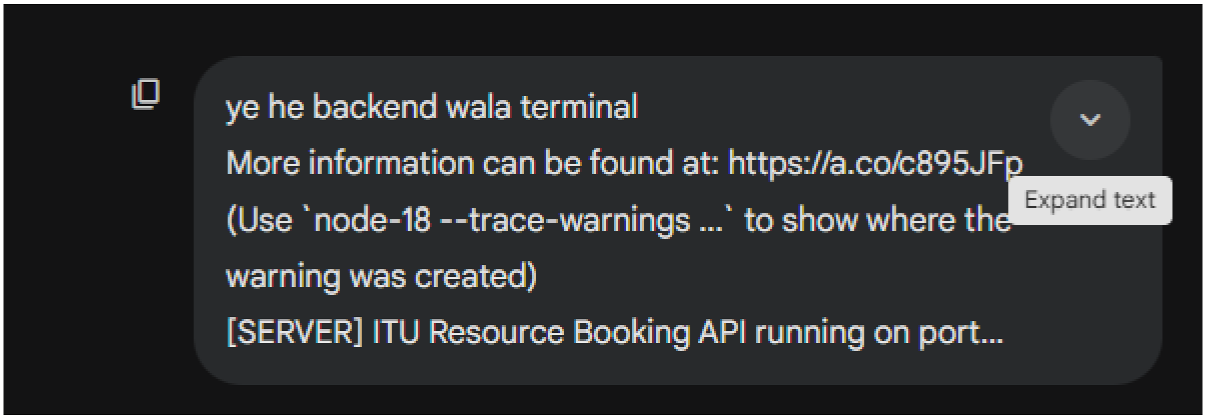
api.ts

```
export const apiService = {
  login: async (email: string, pass: string) => {
    const { data } = await api.post('/auth/login', { email,
password: pass });...
```



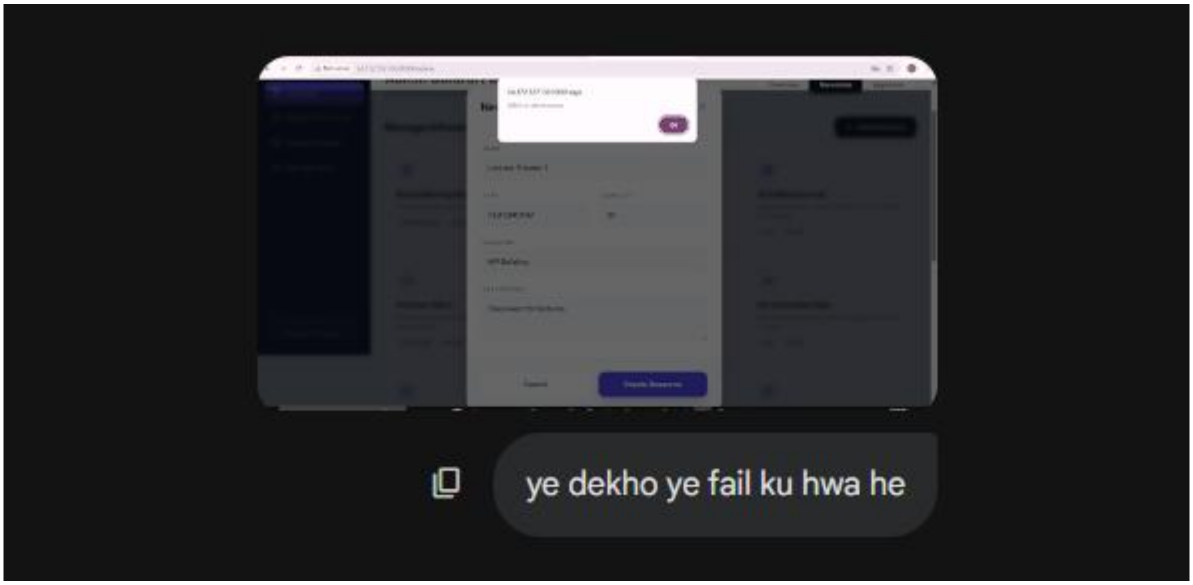
yaar fail ho gya sign up
ye aya he console pr:

```
(index):64 cdn.tailwindcss.com should not be used in
production. To use Tailwind CSS in production, install it
as a PostCSS plugin or use the Tailwind CLI:
https://tailwindcss.com/docs/installation
(anonymous) @ (index):64
react-dom_client.js?v=b5e98812:20103 Download the
React DevTools for a better development experience:
https://react.dev/link/react-devtools
:3000/#/signup:1 Access to XMLHttpRequest at
'http://54.172.127.74:5000/api/auth/signup' from origin
'http://54.172.127.74:3000' has been blocked by CORS
policy: Response to preflight request doesn't pass
access control check: The 'Access-Control-Allow-Origin'
header has a value 'http://localhost:3000' that is not
equal to the supplied origin.
:5000/api/auth/signup:1  Failed to load resource:
net::ERR_FAILED
```

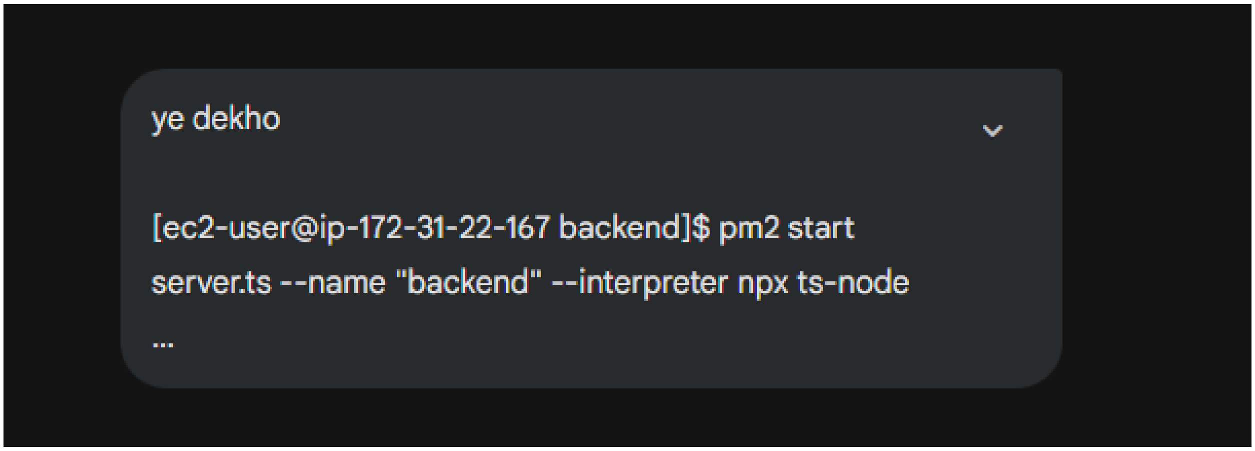


ye he backend wala terminal

```
More information can be found at: https://a.co/c895JFp
(Use `node-18 --trace-warnings ...` to show where the
warning was created)
[SERVER] ITU Resource Booking API running on port...
```

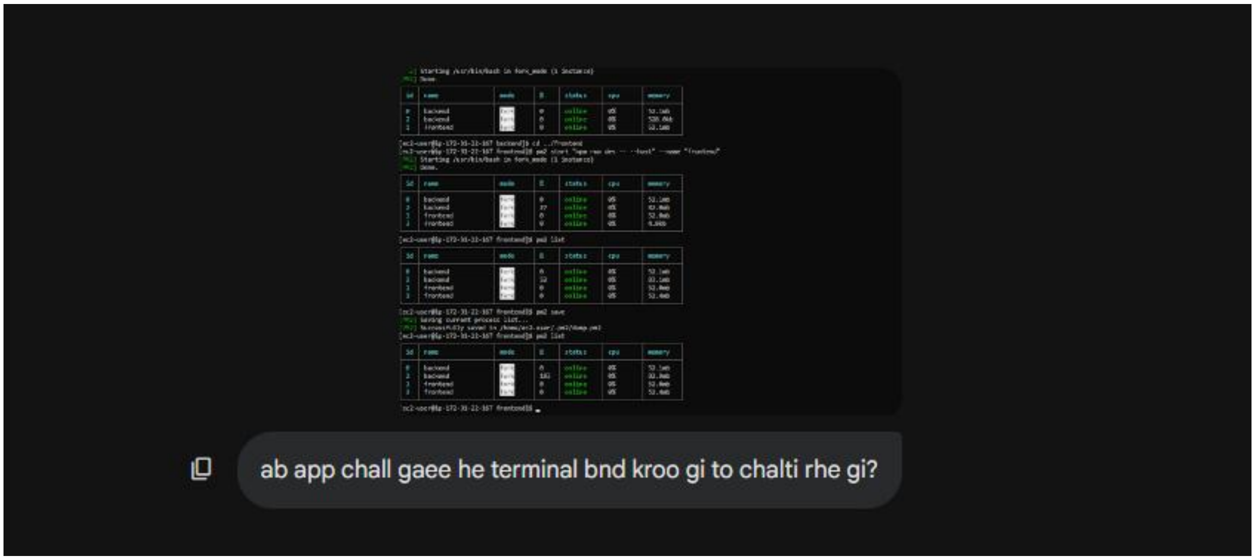


ye dekho ye fail ku hwa he

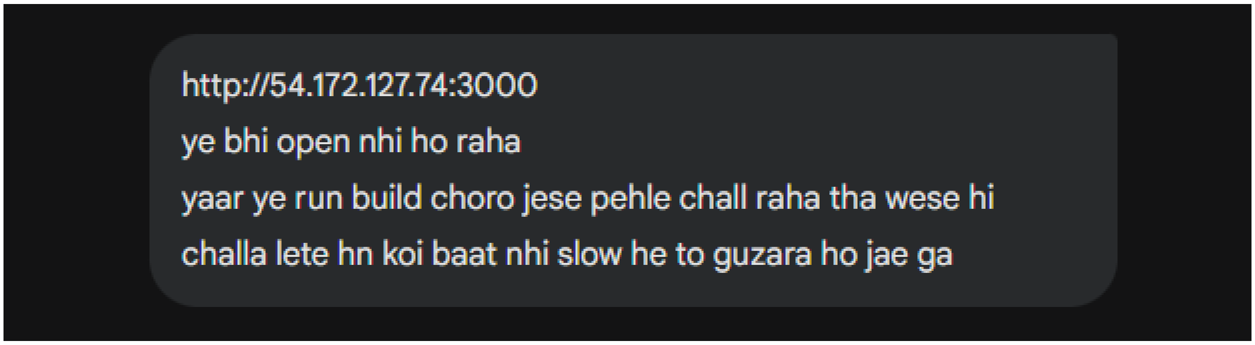


ye dekho

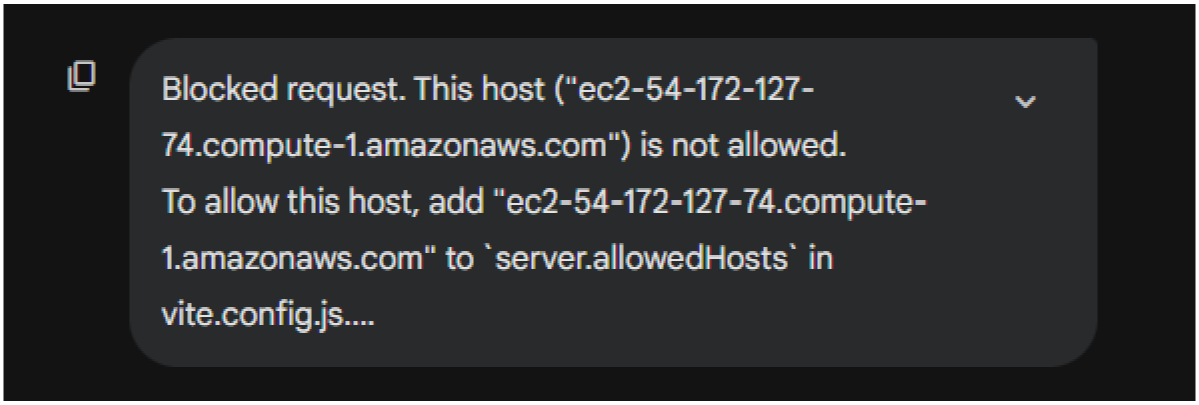
```
[ec2-user@ip-172-31-22-167 backend]$ pm2 start
server.ts --name "backend" --interpreter npx ts-node
...
```



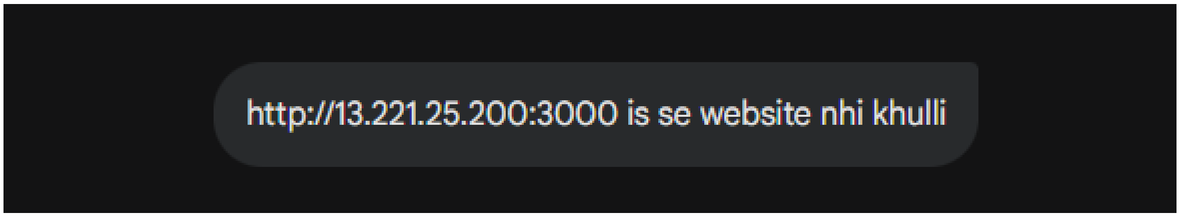
ab app chall gaee he terminal bnd kroo gi to chalti rhe gi?



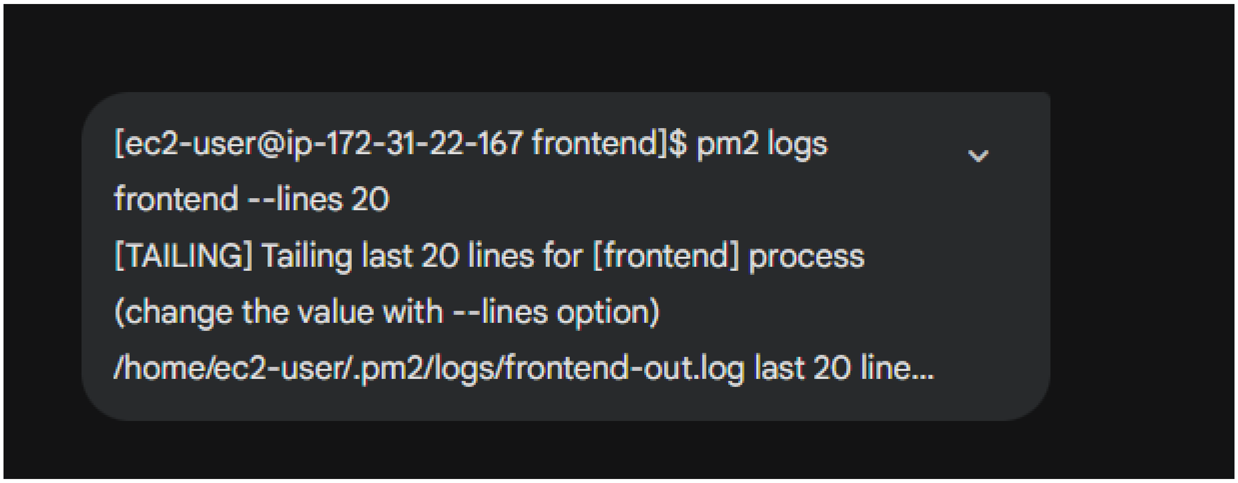
http://54.172.127.74:3000
ye bhi open nhi ho raha
yaar ye run build choro jese pehle chall raha tha wese hi challa lete hn koi baat nhi slow he to guzara ho jae ga



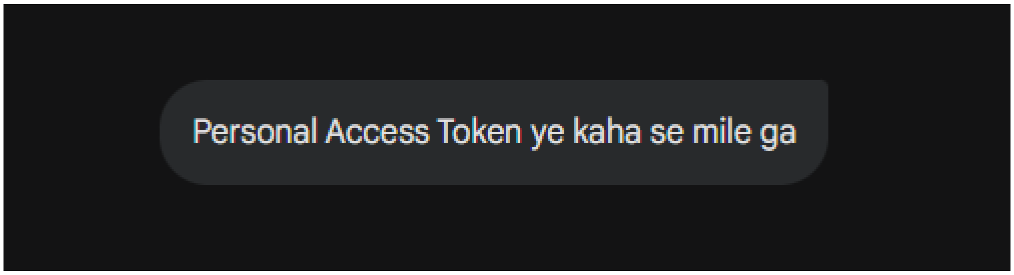
Blocked request. This host ("ec2-54-172-127-74.compute-1.amazonaws.com") is not allowed.
To allow this host, add "ec2-54-172-127-74.compute-1.amazonaws.com" to `server.allowedHosts` in vite.config.js....



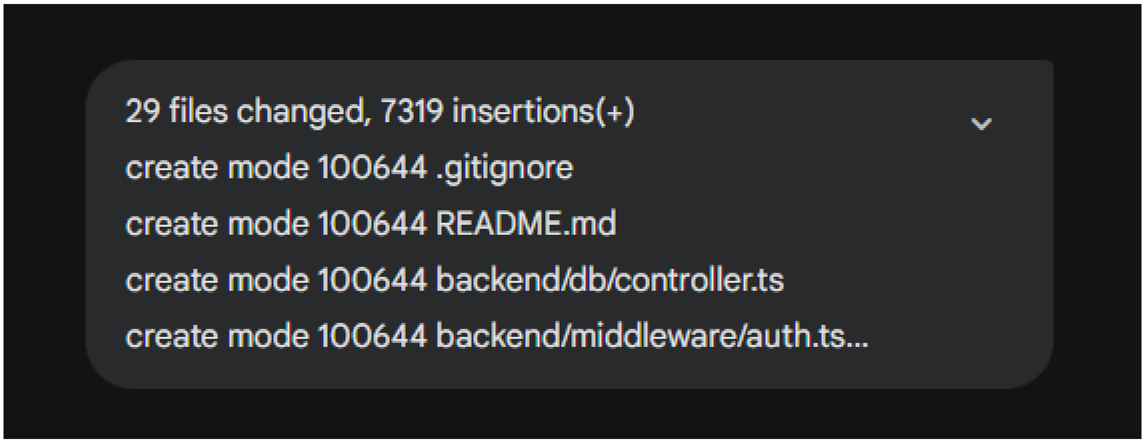
http://13.221.25.200:3000 is se website nhi khulli



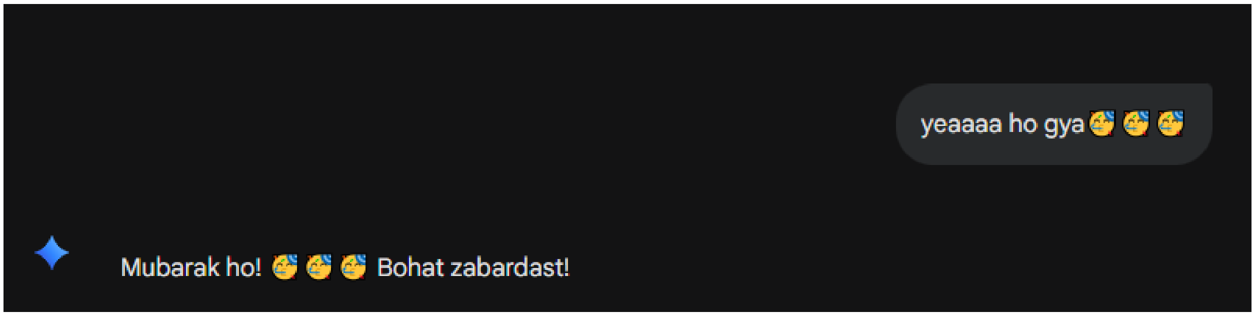
```
[ec2-user@ip-172-31-22-167 frontend]$ pm2 logs
frontend --lines 20
[TAILING] Tailing last 20 lines for [frontend] process
(change the value with --lines option)
/home/ec2-user/.pm2/logs/frontend-out.log last 20 line...
```



Personal Access Token ye kaha se mile ga



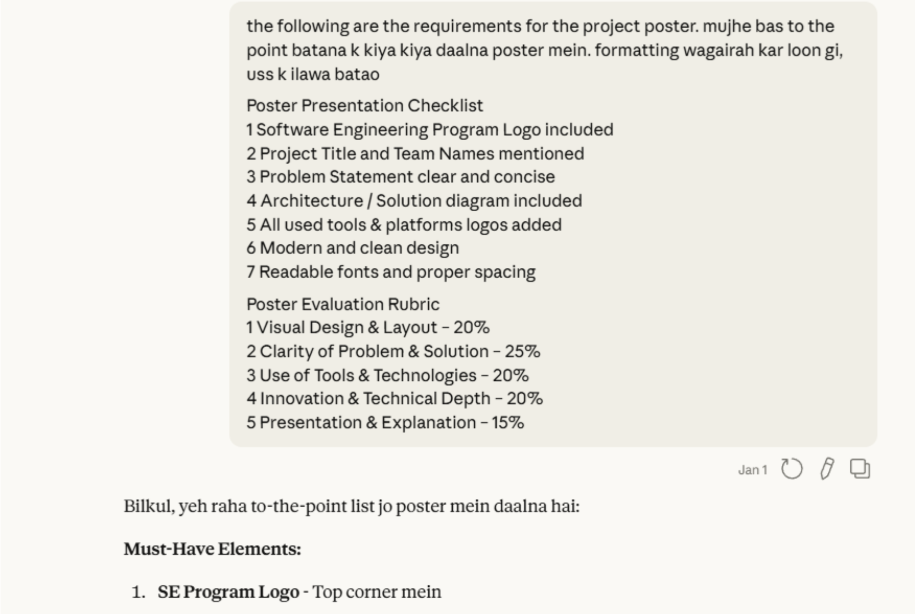
```
29 files changed, 7319 insertions(+)
create mode 100644 .gitignore
create mode 100644 README.md
create mode 100644 backend/db/controller.ts
create mode 100644 backend/middleware/auth.ts...
```



yeaaaa ho gya🥳 🥳 🥳

Mubarak ho! 🥳 🥳 🥳 Bohat zabardast!

Poster:



the following are the requirements for the project poster. mujhe bas to the point batana k kiya kiya daalna poster mein. formatting wagairah kar loon gi, uss k ilawa batao

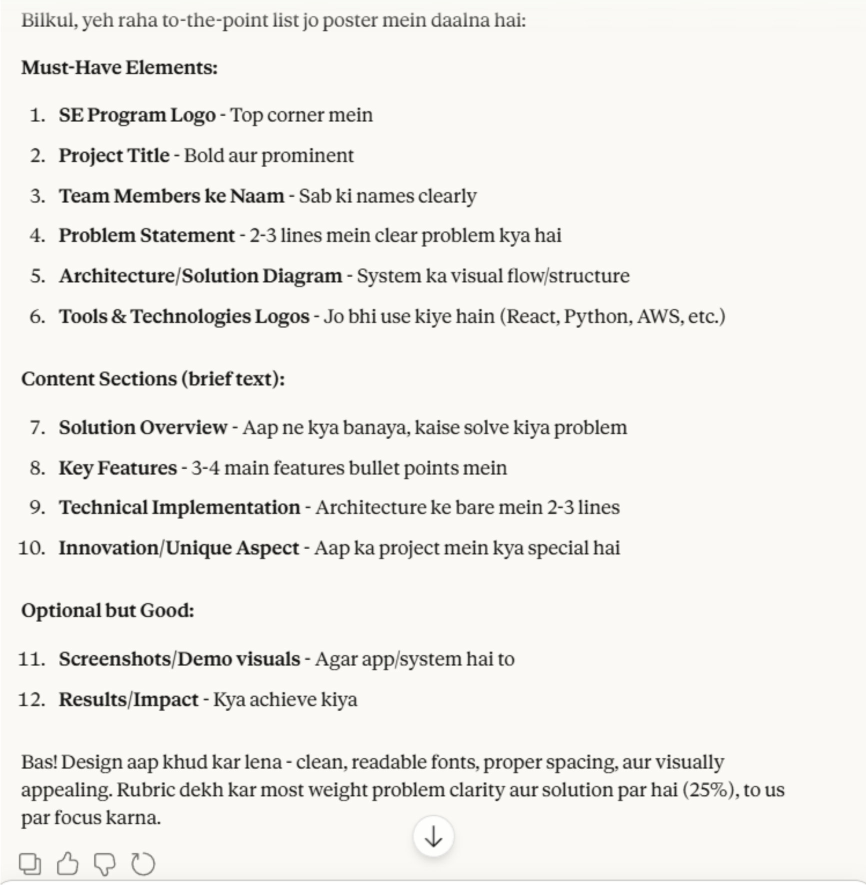
Poster Presentation Checklist
1 Software Engineering Program Logo included
2 Project Title and Team Names mentioned
3 Problem Statement clear and concise
4 Architecture / Solution diagram included
5 All used tools & platforms logos added
6 Modern and clean design
7 Readable fonts and proper spacing

Poster Evaluation Rubric
1 Visual Design & Layout – 20%
2 Clarity of Problem & Solution – 25%
3 Use of Tools & Technologies – 20%
4 Innovation & Technical Depth – 20%
5 Presentation & Explanation – 15%

Bilkul, yeh raha to-the-point list jo poster mein daalna hai:

Must-Have Elements:

1. **SE Program Logo** - Top corner mein



Bilkul, yeh raha to-the-point list jo poster mein daalna hai:

Must-Have Elements:

1. **SE Program Logo** - Top corner mein

2. **Project Title** - Bold aur prominent
3. **Team Members ke Naam** - Sab ki names clearly
4. **Problem Statement** - 2-3 lines mein clear problem kya hai
5. **Architecture/Solution Diagram** - System ka visual flow/structure
6. **Tools & Technologies Logos** - Jo bhi use kiye hain (React, Python, AWS, etc.)

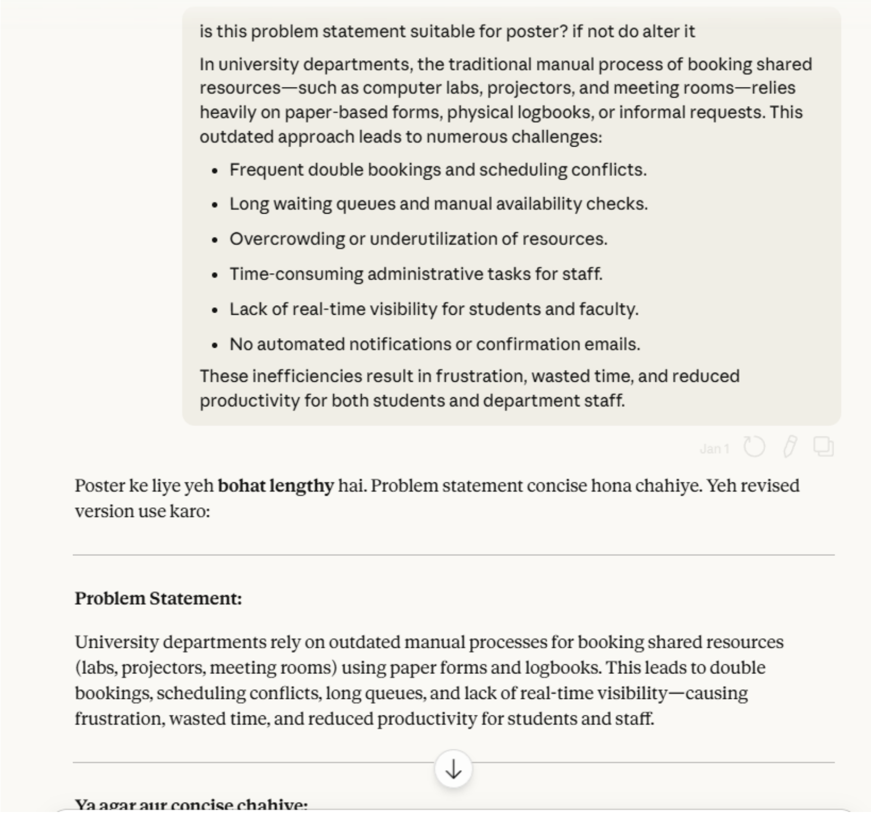
Content Sections (brief text):

7. **Solution Overview** - Aap ne kya banaya, kaise solve kiya problem
8. **Key Features** - 3-4 main features bullet points mein
9. **Technical Implementation** - Architecture ke bare mein 2-3 lines
10. **Innovation/Unique Aspect** - Aap ka project mein kya special hai

Optional but Good:

11. **Screenshots/Demo visuals** - Agar app/system hai to
12. **Results/Impact** - Kya achieve kiya

Bas! Design aap khud kar lena - clean, readable fonts, proper spacing, aur visually appealing. Rubric dekh kar most weight problem clarity aur solution par hai (25%), to us par focus karna.



is this problem statement suitable for poster? if not do alter it

In university departments, the traditional manual process of booking shared resources—such as computer labs, projectors, and meeting rooms—relies heavily on paper-based forms, physical logbooks, or informal requests. This outdated approach leads to numerous challenges:

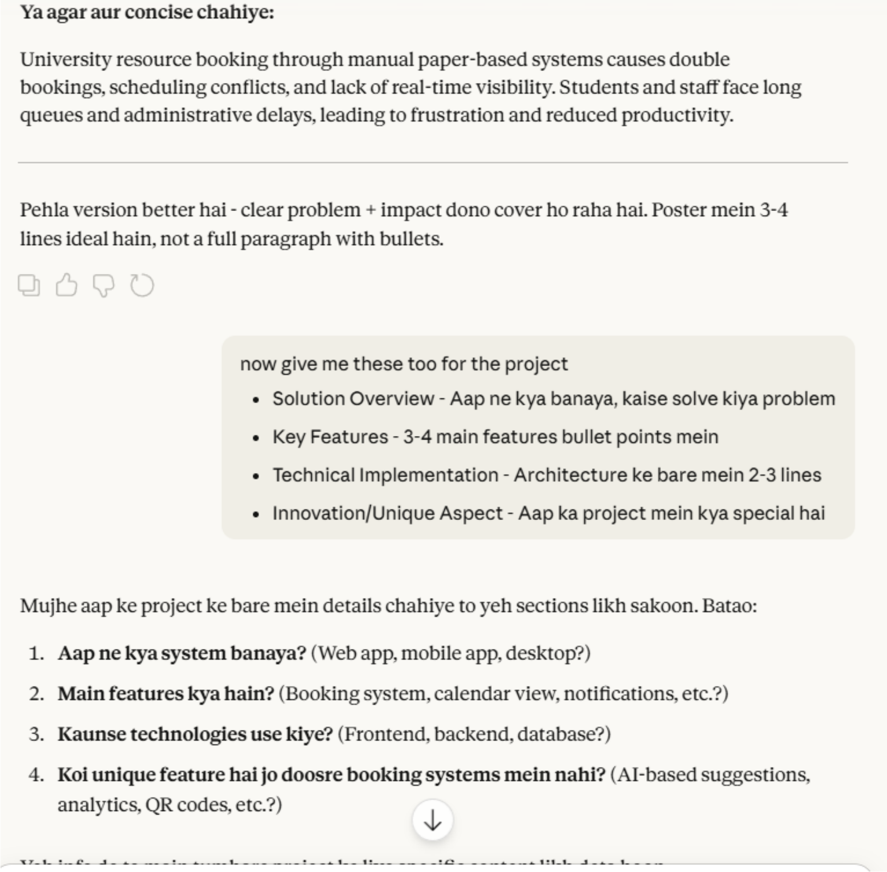
- Frequent double bookings and scheduling conflicts.
- Long waiting queues and manual availability checks.
- Overcrowding or underutilization of resources.
- Time-consuming administrative tasks for staff.
- Lack of real-time visibility for students and faculty.
- No automated notifications or confirmation emails.

These inefficiencies result in frustration, wasted time, and reduced productivity for both students and department staff.

Poster ke liye yeh **bohat lengthy** hai. Problem statement concise hona chahiye. Yeh revised version use karo:

Problem Statement:

University departments rely on outdated manual processes for booking shared resources (labs, projectors, meeting rooms) using paper forms and logbooks. This leads to double bookings, scheduling conflicts, long queues, and lack of real-time visibility—causing frustration, wasted time, and reduced productivity for students and staff.



Ya agar aur concise chahiye:

University resource booking through manual paper-based systems causes double bookings, scheduling conflicts, and lack of real-time visibility. Students and staff face long queues and administrative delays, leading to frustration and reduced productivity.

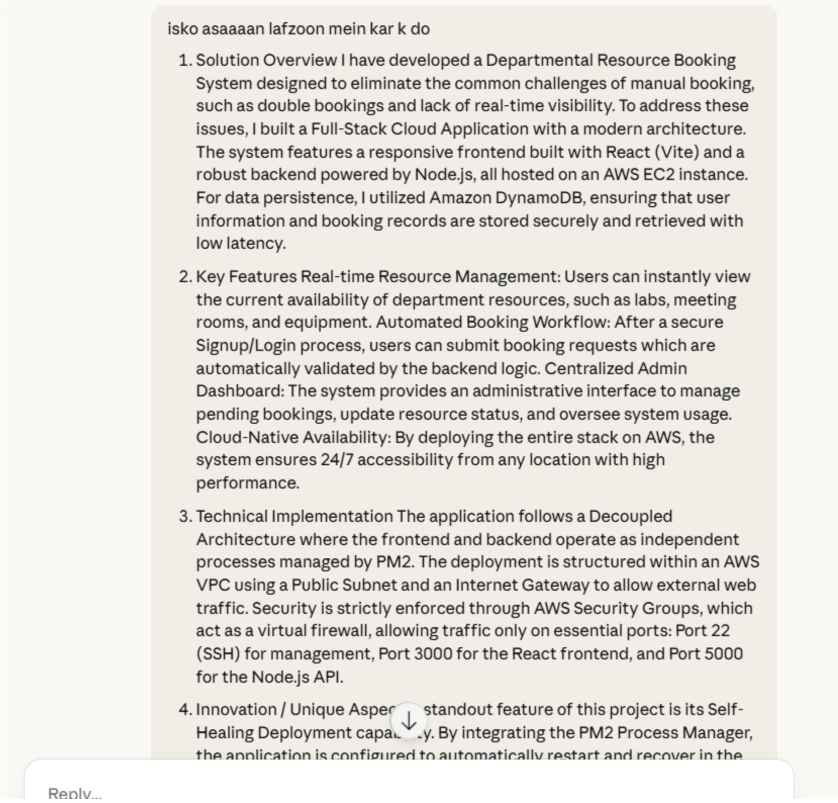
Pehla version better hai - clear problem + impact dono cover ho raha hai. Poster mein 3-4 lines ideal hain, not a full paragraph with bullets.

now give me these too for the project

- Solution Overview - Aap ne kya banaya, kaise solve kiya problem
- Key Features - 3-4 main features bullet points mein
- Technical Implementation - Architecture ke bare mein 2-3 lines
- Innovation/Unique Aspect - Aap ka project mein kya special hai

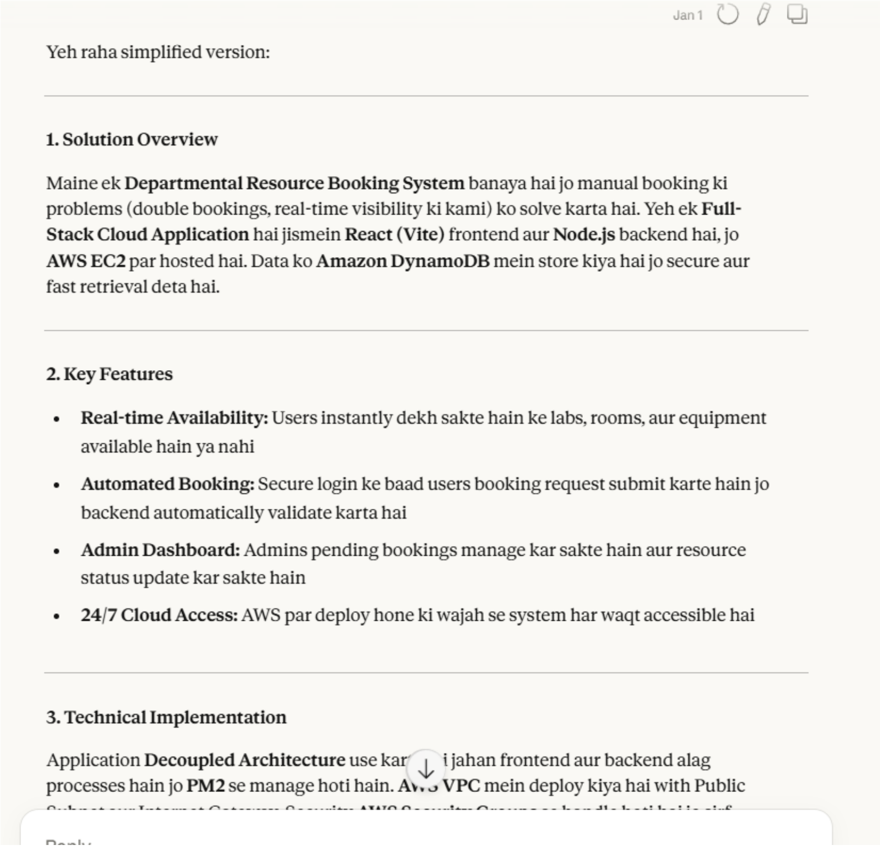
Mujhe aap ke project ke bare mein details chahiye to yeh sections likh sakoon. Batao:

1. **Aap ne kya system banaya?** (Web app, mobile app, desktop?)
2. **Main features kya hain?** (Booking system, calendar view, notifications, etc.?)
3. **Kaunse technologies use kiye?** (Frontend, backend, database?)
4. **Koi unique feature hai jo doosre booking systems mein nahi?** (AI-based suggestions, analytics, QR codes, etc.?)



isko asaaaan lafzoon mein kar k do

1. Solution Overview I have developed a Departmental Resource Booking System designed to eliminate the common challenges of manual booking, such as double bookings and lack of real-time visibility. To address these issues, I built a Full-Stack Cloud Application with a modern architecture. The system features a responsive frontend built with React (Vite) and a robust backend powered by Node.js, all hosted on an AWS EC2 instance. For data persistence, I utilized Amazon DynamoDB, ensuring that user information and booking records are stored securely and retrieved with low latency.
2. Key Features Real-time Resource Management: Users can instantly view the current availability of department resources, such as labs, meeting rooms, and equipment. Automated Booking Workflow: After a secure Signup/Login process, users can submit booking requests which are automatically validated by the backend logic. Centralized Admin Dashboard: The system provides an administrative interface to manage pending bookings, update resource status, and oversee system usage. Cloud-Native Availability: By deploying the entire stack on AWS, the system ensures 24/7 accessibility from any location with high performance.
3. Technical Implementation The application follows a Decoupled Architecture where the frontend and backend operate as independent processes managed by PM2. The deployment is structured within an AWS VPC using a Public Subnet and an Internet Gateway to allow external web traffic. Security is strictly enforced through AWS Security Groups, which act as a virtual firewall, allowing traffic only on essential ports: Port 22 (SSH) for management, Port 3000 for the React frontend, and Port 5000 for the Node.js API.
4. Innovation / Unique Aspec standout feature of this project is its Self-Healing Deployment capa ty. By integrating the PM2 Process Manager, the application is configured to automatically restart and recover in the



Yeh raha simplified version:

---

1. Solution Overview

Maine ek **Departmental Resource Booking System** banaya hai jo manual booking ki problems (double bookings, real-time visibility ki kami) ko solve karta hai. Yeh ek **Full-Stack Cloud Application** hai jismein **React (Vite)** frontend aur **Node.js** backend hai, jo **AWS EC2** par hosted hai. Data ko **Amazon DynamoDB** mein store kiya hai jo secure aur fast retrieval deta hai.

---

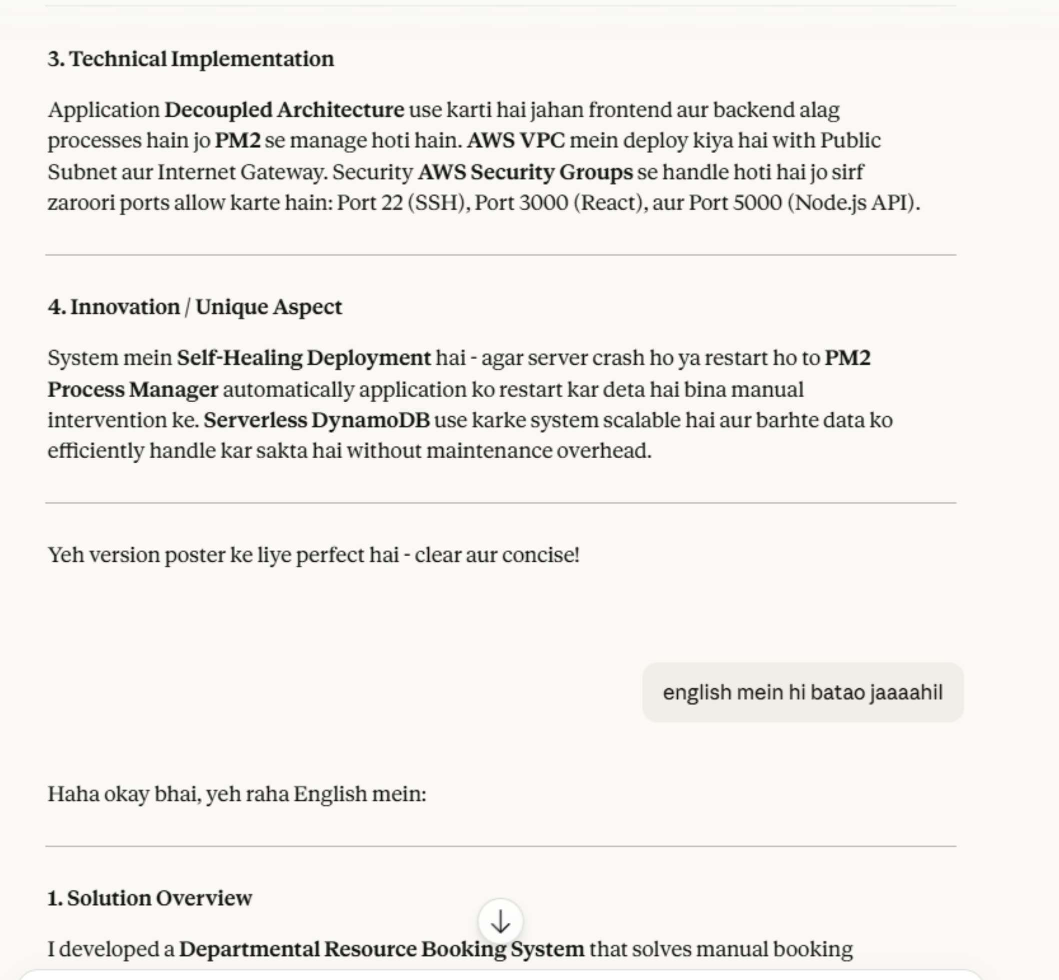
2. Key Features

- **Real-time Availability:** Users instantly dekh sakte hain ke labs, rooms, aur equipment available hain ya nahi
- **Automated Booking:** Secure login ke baad users booking request submit karte hain jo backend automatically validate karta hai
- **Admin Dashboard:** Admins pending bookings manage kar sakte hain aur resource status update kar sakte hain
- **24/7 Cloud Access:** AWS par deploy hone ki wajah se system har waqt accessible hai

---

3. Technical Implementation

Application **Decoupled Architecture** use kar i jahan frontend aur backend alag processes hain jo **PM2** se manage hoti hain. **AWS VPC** mein deploy kiya hai with Public



3. Technical Implementation

Application **Decoupled Architecture** use karti hai jahan frontend aur backend alag processes hain jo **PM2** se manage hoti hain. **AWS VPC** mein deploy kiya hai with Public Subnet aur Internet Gateway. Security **AWS Security Groups** se handle hoti hai jo sirf zaroori ports allow karte hain: Port 22 (SSH), Port 3000 (React), aur Port 5000 (Node.js API).

---

4. Innovation / Unique Aspect

System mein **Self-Healing Deployment** hai - agar server crash ho ya restart ho to **PM2 Process Manager** automatically application ko restart kar deta hai bina manual intervention ke. **Serverless DynamoDB** use karke system scalable hai aur barhte data ko efficiently handle kar sakta hai without maintenance overhead.

Yeh version poster ke liye perfect hai - clear aur concise!

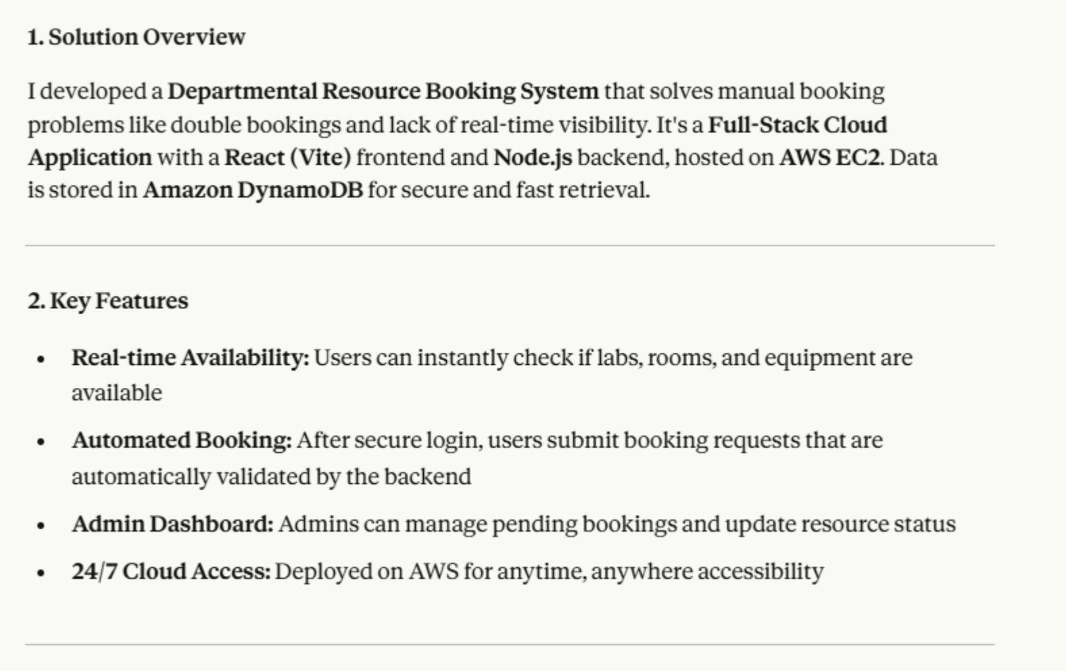
english mein hi batao jaaaahil

Haha okay bhai, yeh raha English mein:

---

1. Solution Overview

I developed a **Departmental Resource Booking System** that solves manual booking



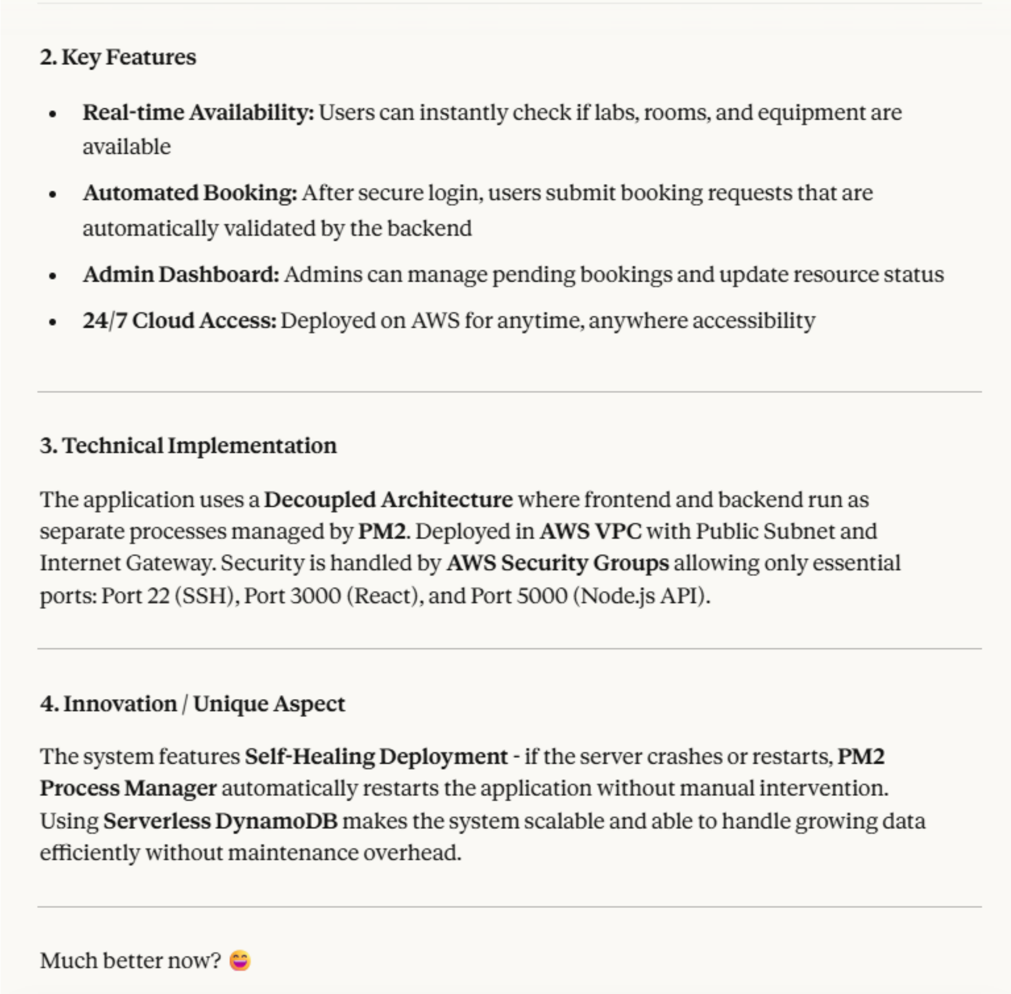
1. Solution Overview

I developed a **Departmental Resource Booking System** that solves manual booking problems like double bookings and lack of real-time visibility. It's a **Full-Stack Cloud Application** with a **React (Vite)** frontend and **Node.js** backend, hosted on **AWS EC2**. Data is stored in **Amazon DynamoDB** for secure and fast retrieval.

---

2. Key Features

- **Real-time Availability:** Users can instantly check if labs, rooms, and equipment are available
- **Automated Booking:** After secure login, users submit booking requests that are automatically validated by the backend
- **Admin Dashboard:** Admins can manage pending bookings and update resource status
- **24/7 Cloud Access:** Deployed on AWS for anytime, anywhere accessibility



2. Key Features

- **Real-time Availability:** Users can instantly check if labs, rooms, and equipment are available
- **Automated Booking:** After secure login, users submit booking requests that are automatically validated by the backend
- **Admin Dashboard:** Admins can manage pending bookings and update resource status
- **24/7 Cloud Access:** Deployed on AWS for anytime, anywhere accessibility

3. Technical Implementation

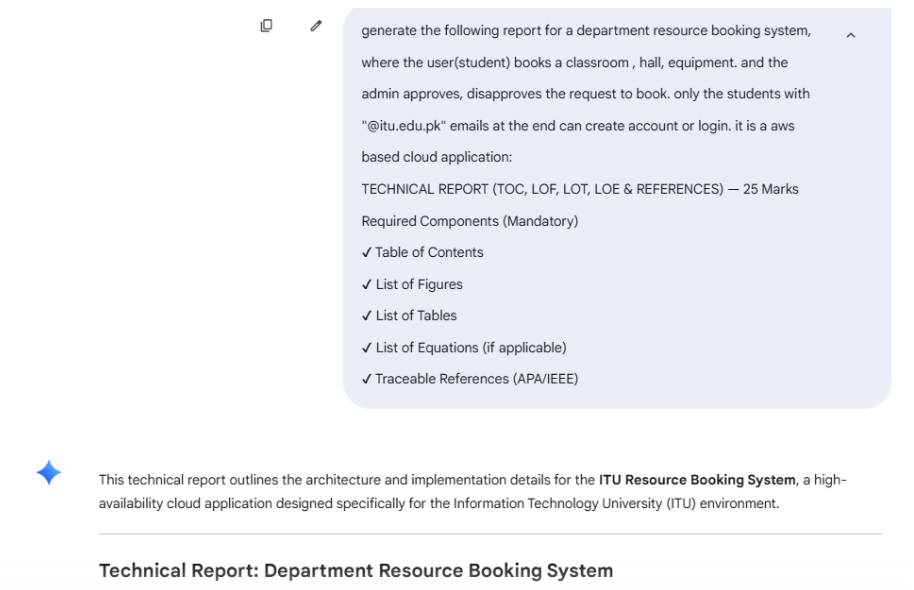
The application uses a **Decoupled Architecture** where frontend and backend run as separate processes managed by **PM2**. Deployed in **AWS VPC** with Public Subnet and Internet Gateway. Security is handled by **AWS Security Groups** allowing only essential ports: Port 22 (SSH), Port 3000 (React), and Port 5000 (Node.js API).

4. Innovation / Unique Aspect

The system features **Self-Healing Deployment** - if the server crashes or restarts, **PM2 Process Manager** automatically restarts the application without manual intervention. Using **Serverless DynamoDB** makes the system scalable and able to handle growing data efficiently without maintenance overhead.

Much better now? 😄

Report:



generate the following report for a department resource booking system, where the user(student) books a classroom , hall, equipment. and the admin approves, disapproves the request to book. only the students with "@itu.edu.pk" emails at the end can create account or login. it is a aws based cloud application:

TECHNICAL REPORT (TOC, LOF, LOT, LOE & REFERENCES) — 25 Marks

Required Components (Mandatory)

✓ Table of Contents

✓ List of Figures

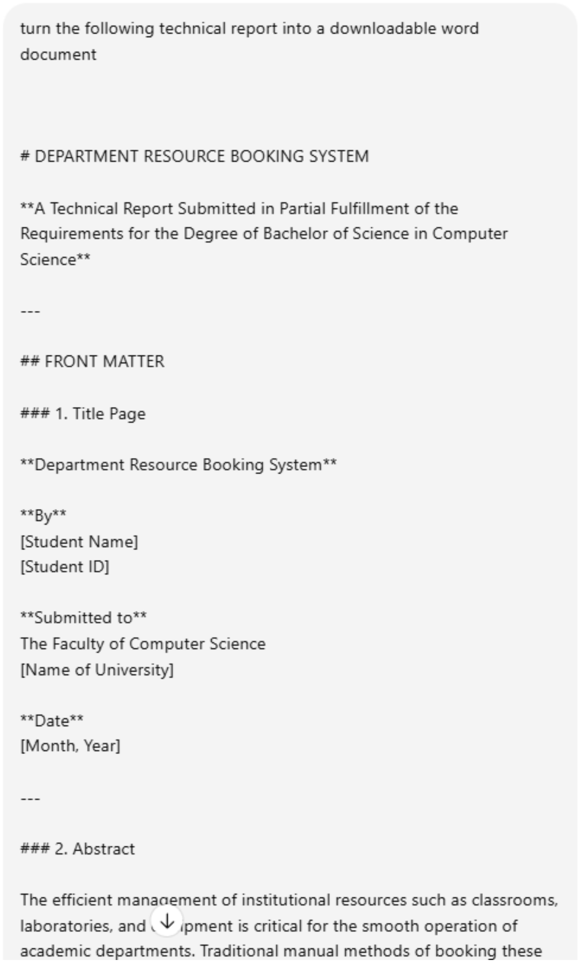
✓ List of Tables

✓ List of Equations (if applicable)

✓ Traceable References (APA/IEEE)

This technical report outlines the architecture and implementation details for the **ITU Resource Booking System**, a high-availability cloud application designed specifically for the Information Technology University (ITU) environment.

Technical Report: Department Resource Booking System



turn the following technical report into a downloadable word document

DEPARTMENT RESOURCE BOOKING SYSTEM

A Technical Report Submitted in Partial Fulfillment of the Requirements for the Degree of Bachelor of Science in Computer Science

---

FRONT MATTER

1. Title Page

Department Resource Booking System

By
[Student Name]
[Student ID]

Submitted to
The Faculty of Computer Science
[Name of University]

Date
[Month, Year]

---

2. Abstract

The efficient management of institutional resources such as classrooms, laboratories, and equipment is critical for the smooth operation of academic departments. Traditional manual methods of booking these paraphrase to avoid plagerism:

Acknowledgements
The authors would like to thank the faculty and staff of the Information Technology University
of Punjab for their guidance and support throughout the development of this project. Special
thanks are due to our supervisors, Dr. Zunnurain and Umair Makhdom, for their valuable
feedback on system design decisions and AWS best practices.

Acknowledgements

The authors express their sincere gratitude to the faculty and staff of the Information Technology University of Punjab for their continuous support and guidance during the course of this project. We are especially thankful to our supervisors, **Dr. Zunnurain** and **Umair Makhdom**, whose insightful suggestions, constructive feedback, and expertise in system architecture and AWS best practices greatly contributed to the successful completion of this work.

transform this into technical report suitable text:

Backend aur frontend ki testing ke liye hum ne Manual aur Automated dono tareeqon ka istemal kiya hai taake ye confirm ho sake ke dono servers internet par sahi se kaam kar rahe hain:

1. Backend Testing (API Verification)
Backend ko test karne ke liye hum ne multiple stages istemal kiye:

Process Monitoring: Hum ne pm2 list command ka istemal kiya ye check karne ke liye ke backend "Online" hai aur uski memory usage (e.g., 90.4mb) normal hai.

Local Connectivity: Terminal par curl http://localhost:5000/api/auth/login chala kar hum ne ye dekha ke server requests accept kar raha hai ya nahi.

Public Access: Browser mein direct Public IP (http://54.82.99.197:5000/api/resources) enter karke check kiya gaya ke AWS Security Group ne connection allow kiya hai ya nahi.

Logs Analysis: pm2 logs backend ke zariye hum ne runtime errors (jaise ERR_UNKNOWN_FILE_EXTENSION) ko identify aur fix kiya.

2. Frontend Testing (UI & Integration)
Frontend ki testing ke liye ye steps follow kiye gaye:

Deployment Check: pm2 list se confirm kiya ke frontend server (Port 3000) par online hai.

Service Binding: frontend/src/services/api.ts mein naye instance ki IP (54.82.99.197) update karke ye ensure kiya gaya ke frontend sahi backend se baat kar raha hai.