1. Write Test Cases for the above.

Uploaded in github- test\_backend.py

2: How do you plan on deploying this to the production environment?

**Answer:**

I plan to deploy the backend using **Render** or **PythonAnywhere**, which are beginner-friendly platforms for deploying Flask-based web apps.

Here’s how I’d deploy it step-by-step on **Render** (free tier):

**1. Prepare the project for production**

* Add a requirements.txt file:

pgsql

pip freeze > requirements.txt

* Create a Procfile (no extension) with this line:

arduino

CopyEdit

web: gunicorn run:app

* Make sure environment variables like SECRET\_KEY, JWT\_SECRET\_KEY, etc., are moved to .env and loaded with python-dotenv.

**2. Push code to GitHub**

* Initialize Git:

bash

CopyEdit

git init

git remote add origin <your-repo-url>

git add .

git commit -m "Initial deployment"

git push -u origin main

**3. Deploy to Render**

* Go to <https://render.com>
* Click "New Web Service"
* Connect your GitHub repo
* Choose:
  + **Environment**: Python 3
  + **Build command**: pip install -r requirements.txt
  + **Start command**: gunicorn run:app
* Add environment variables manually in the Render dashboard

**Production Notes**

* Use gunicorn instead of Flask's dev server
* Set debug=False in production
* Use .env for secrets (and never commit it)

**Optional: Use Docker (if required for scalable infra)**

For more advanced production environments (like AWS EC2, Heroku, or Docker-based servers), I’d containerize it and use Nginx + Gunicorn combo behind a reverse proxy.