Problem Statement:

Many coding platforms (LeetCode, GFG, CodeChef) provide **final solutions** directly, which discourages critical thinking. Students often copy solutions without learning the actual problem solving approach.

Our Solution:

Our system will act as a **step by step coding coach**, providing students with guided hints, questions, and explanations **instead of giving full answers immediately**.

Objectives:

1. Develop a **Chrome Extension and Web Application** that seamlessly integrates with coding platforms.
2. Provide comprehensive **guidance** through AI-driven step-by-step solutions, eliminating the need for onetime fixes.
3. Monitor user progress and **construct a personalized skill profile**.
4. Recommend appropriate problems and learning paths based on identified weaknesses.
5. Ensure scalability through the utilization of cloud-based infrastructure**, DevOps practices, and AI microservices**.

Tech Stack:

- **Frontend:** React.js, Chrome Extension (Manifest V3), TailwindCSS
- **Backend:** Spring Boot (REST APIs, WebSockets, Authentication)
- **AI Service:** Python (FastAPI/Flask), Hugging Face/OpenAI APIs, Machine Learning recommendation models
- **Database:** PostgreSQL (structured data), MongoDB (logs, hints)
- **Cloud & DevOps:** Docker, Kubernetes, AWS/GCP, GitHub Actions (Continuous Integration/Continuous Deployment), Monitoring (Prometheus/Grafana)

3 Semester Execution Plan

Semester 1: Foundation & MVP (Core System)

Goal: Establish a functional base system (no AI yet, rulebased hints).

- **Deliverables:**
  - Chrome Extension: Overlay UI on coding platforms (LeetCode, GFG).

    - Web Dashboard (React): User login, progress tracking, problem history.
    - Backend (Spring Boot): APIs for authentication, storing user attempts, and progress.
    - Database Schema (Users, Problems, Progress, Hints): Define the structure of the database.
    - Initial Hinting System (Manual/RuleBased Hints): Store hints in the database.
    - Cloud Deployment (Basic): Deploy backend and database on AWS/GCP.
    - CI/CD Pipeline for Frontend and Backend: Implement continuous integration and continuous deployment pipelines.

Semester 2: Intelligence and AI Integration

Goal: Enhance the system with real AI capabilities and personalized learning features.

- **Deliverables:**
    - **AI Microservice (Python FastAPI):** A microservice connected to the backend.
    - **LLM Integration (Hugging Face / OpenAI):** Integration of LLMs for generating **contextual hints**.
    - **"Socratic Guidance" Module:** A module where AI asks questions instead of providing direct answers.
    - **Skill Graph Model:** A model to track strengths and weaknesses (e.g., DP weak, Arrays strong).
    - **Recommendation Engine:** An engine that suggests next problems based on the skill graph.
    - **Realtime Hinting via WebSockets:** Realtime hinting capabilities.
    - **Monitoring and Logging System:** A system for monitoring and logging user activities.

Semester 3: Scaling, Cloud, and Polish

Goal: Make the system robust, productionready, and userfriendly.

- **Deliverables:**
    - **Advanced Hint Personalization:** Finetuned models, hybrid rule systems, and AI systems for personalized hints.
    - **CloudNative Deployment:** Deployment using Docker, Kubernetes, and load balancing.
    - **Optimized AI Service:** Optimized AI service with caching, reduced latency, and model selection.
    - **ElasticSearch Integration:** Integration of ElasticSearch for fast problem and hint search.
    - **Analytics Dashboard:** A dashboard to track student improvement over time.
    - **Security Enhancements:** Security enhancements such as Spring Security and OAuth2 (if needed).
    - **Final Testing with Users:** Final testing with users (classmates, juniors) and a feedback loop.
    - **Documentation and Research Paperstyle Report:** Documentation and a research paperstyle report for Major submission.

Final Deliverables

- **Chrome Extension + Web App** (React)
- **Spring Boot Backend** with APIs
- **AI Microservice** (Python + Hugging Face/OpenAI)

- **Databases** (PostgreSQL + MongoDB)
- **Cloud Deployment** (AWS/Google Cloud Platform, containerized with CI/CD)
- **Project Report + Presentation + Live Demo**

Why This Project Is Strong

- **Novelty:** Unlike coding platforms that provide prewritten solutions, this *project emphasizes critical thinking*.
- **Realworld Relevance:** Students often resort to copying solutions. Your system aims to develop problem solving skills.
- **Scalability:** The project has the potential to evolve into a viable product, such as an educational technology startup.
- **Technical Depth: It covers a comprehensive range of topics**, including **frontend development, backend programming, artificial intelligence and machine learning, cloud computing, and DevOps** practices, which are essential components of a major project.

Tech Stack Summary

- **Frontend:** React.js, Chrome Extension, TailwindCSS
- **Backend:** Spring Boot (REST/GraphQL, Authentication, Database Integration)
- **AI/ML Service:** Python (FastAPI/Flask), Hugging Face/OpenAI, Custom Recommendation ML
- **Databases:** PostgreSQL, MongoDB
- **Cloud/DevOps:** Docker, Kubernetes, AWS/GCP, CI/CD Pipelines
- **Extras:** WebSockets for Realtime Hints, ElasticSearch for Search

Frontend (User Interface)

Purpose: Dashboard + Chrome Extension UI for hints, progress, and interaction.

- **React.js:** Main web dashboard
- **Chrome Extension (React + Manifest V3):** Popup hints and injected UI on coding sites (LeetCode, GFG, etc.)
- **Redux Toolkit / React Query:** State management and asynchronous calls
- **TailwindCSS / Material UI:** Fast and clean UI components

Backend (APIs & Business Logic)

Purpose: User authentication, progress tracking, managing hint sessions, and communication with AI service.

- **Spring Boot (Java):** Robust API layer
- **Spring Security + JWT:** Secure authentication and user management
- **REST APIs / GraphQL:** For communication with frontend and extension
- **WebSockets: Real**time hint updates or interactive sessions

AI/ML Layer (Hint Generation + Personalization)

Purpose: The “intelligence” will guide users step by step.

- **Python (FastAPI / Flask microservice):** Separate AI service (easier than embedding within Spring Boot).
- **LLM Integration:**
    - **Use OpenAI API / Hugging Face models for generating guided hints.**
    - **Finetune or create prompt engineering logic for "step-by-step guidance."**
- **Recommendation Engine:** Suggest next problems, difficulty level (could use collaborative filtering / skill graph model).

Knowledge Tracking: Build a user knowledge profile (strength/weakness detection via ML).

Deployment and Scaling:

- **Cloud Provider:** AWS / GCP / Azure (any, depending on credits or preference).
- **Containers:** Docker + Kubernetes (for backend + AI service + frontend).
- **CI/CD:** GitHub Actions / Jenkins → automated builds, tests, deployments.
- **Cloud Databases:**
    - **PostgreSQL:** Relational data (users, progress, sessions).
    - **MongoDB:** Flexible storage (problem attempts, hint logs).
- **Cloud Storage (S3/Blob):** Store logs, session histories.
- **Monitoring:** Prometheus + Grafana or CloudWatch.

Database Design (HighLevel)

- **Users:**
- id
- name
- email
- Auth_info

- **Problems:**
- id
- source (e.g., LeetCode, GFG)
- difficulty
- Tags

- **UserProgress:**
- user_id
- problem_id
- attempts
- status
- Hints_used

- **Hints:**
- problem_id
- step_number
- hint_text

- AI_source

- **SkillGraph:**
- user_id
- topic
- score