1. Explain how you would implement generative AI for code generation in a full stack project. Outline specific implementation steps and discuss the potential benefits and challenges of this approach. (5 Marks)

When building a full stack project, I would use generative AI as a coding assistant at different stages. For example, during backend development, I could use it to quickly generate REST API endpoints, database schema definitions, or boilerplate code like authentication middleware. On the frontend, I could use AI to prepare reusable React components, CSS layouts, or form validation logic. Instead of writing everything from scratch, I'd describe what I need in natural language, and the AI would provide a starting point that I can refine.

The main benefits are saving time on repetitive code and helping me learn best practices by seeing different code patterns. However, challenges include making sure the generated code is secure, efficient, and consistent with the rest of the project. For example, AI might generate code that works in isolation but doesn't fully fit with my chosen libraries or style, so I need to review and test.

2. How can Generative AI be used to convert natural language requests into SQL queries or API calls? Provide an example and discuss the implications for database management and API design. (5 Marks)

Generative AI can act like a translator between what a user says and the actual query the system needs. For example, if someone types *"Show me all orders from last month where the total was over $500"*, the AI could turn that into an SQL query like:

```
SELECT * FROM orders
WHERE total > 500
AND order_date >= '2025-08-01'
AND order_date < '2025-09-01';
```

This means that non-technical users can get the data they want without knowing SQL or how the API works. But this also means database schemas and API endpoints need to be designed clearly and consistently, otherwise the AI might map the request to the wrong thing.

3. Describe the process of fine-tuning a pre-trained language model for a specific task. What are the advantages of this approach compared to training a model from scratch? (5 Marks)

Fine-tuning involves taking an existing large pre-trained model and training it further on task-specific data. For example, if I want a chatbot specialized in medical advice, I would feed it high-quality medical conversations and let it adjust its weights slightly. The main advantage is efficiency, as the model already understands language broadly, so I only need less data and compute power to adapt it. This is much faster and cheaper than training from scratch, which would require billions of tokens and huge infrastructure. It also gives better performance in domain-specific tasks without losing the general language abilities of the base model.

4. What is prompt engineering, and why is it important when working with large language models? Provide an example of how you would craft an effective prompt for a text generation task. (5 Marks)

Prompt engineering is the practice of carefully designing the input instructions given to a large language model so that it produces the desired output. It matters because the same model can give very different results depending on how you ask the question. For example, instead of asking, “Write about the importance of AI” (which is too broad), I could craft: “Write a 150-word article explaining the importance of AI in e-commerce, using simple language and one real-world example.” This gives the model clear direction and helps ensure the output is relevant, accurate, and useful.