**Project Design Phase-II**

**Solution Requirements (Functional & Non-functional)**

| Date | 30 July 2025 |
| --- | --- |
| Team ID | PNT2025TMID09555 |
| Project Name | The Roadmap Generator |
| Maximum Marks | 4 Marks |

## 

## **Functional Requirements**

These define what your system **must do**.

### **1. Skill Input Interface**

* The user must be able to enter a desired skill via a web UI.
* Optional: Auto-suggestions or dropdowns for popular skills.

### **2. Roadmap Generation**

* The system should process the input using the LLaMA2 model via CTransformers.
* A personalized, step-by-step roadmap must be generated.
* Each step should include:
  + Learning objectives
  + Duration or phase
  + Recommended **free** resources

### **3. Integration with Pre-trained Model**

* The application must load the LLaMA2 .gguf model.
* Input prompts should be passed efficiently, and output retrieved and parsed.

### **4. Web Interface via Streamlit**

* Display the generated roadmap cleanly.
* Provide a “Generate Again” button for re-generation.
* Optional: Export roadmap to PDF or markdown.

### **5. Model Management**

* On first run, the app should download the required .gguf model using a script (setup.sh) if it does not already exist.

### **6. User Feedback (Optional for Phase II)**

* Optionally collect user feedback on roadmap accuracy and relevance.

## **Non-Functional Requirements**

These define **how** the system should behave.

### **1. Performance**

* Roadmap generation must complete within **10–15 seconds**.
* The app must not crash for typical user inputs.

### **2. Scalability**

* Must support multiple users simultaneously (if hosted on Streamlit Cloud or expanded to a scalable platform later).
* Modular design to support multiple models or multiple input types in the future.

### **3. Reliability**

* Setup script should ensure the model is always available.
* Proper error handling in place (e.g., invalid input, model download failure).

### **4. Maintainability**

* Code should follow modular structure (e.g., separate model\_loader.py, utils, etc.).
* README and comments should help future developers understand and modify the system.

### **5. Security**

* If hosted publicly:
  + Validate user inputs to avoid prompt injection.
  + No logging of sensitive input data.

### **6. Portability**

* The app should run on any machine with Python 3.8+, Streamlit, and minimal setup (setup.sh to handle model).

## **Deliverables from Phase II**

* Project directory with:
  + Streamlit app (app.py)
  + Model loader module
  + setup.sh to download model
  + requirements.txt
  + .gitignore
  + README.md with setup instructions