Alright Anbhi, here's a **formal Weekly Report** draft for your chatbot project.
I've structured it clearly, so you can directly use it in a Word/PDF document.

Weekly Report
Name: Anbhi Thakur
Date: 17 August 2025

Project Title:

LLM Chatbot with Advanced Features (Tool-Integration)

Overview

The goal of this project is to design and deploy a chatbot with advanced AI capabilities, including document ingestion, retrieval-augmented generation (RAG), agentic reasoning, and integration with external tools. The project has been containerized with Docker for seamless deployment and tested with a frontend built using Streamlit.

Tasks Completed

Task 1 – LLM Chatbot with Docker

- Developed backend using FastAPI and integrated with OpenAI APIs.
- Implemented REST endpoints (/chat, /upload, /docs/list, /agent).

- Dockerized the application with a production-ready Dockerfile.
- Configured deployment for Render/cloud hosting.

Task 2 – Attachment Analysis

- Implemented file ingestion for PDFs, images, and text files.
- Used PyMuPDF, Tesseract OCR, and file-type detection for preprocessing.
- Added text chunking, embedding generation, and database persistence.
- Enabled summarization and preview of uploaded files.

Task 3 – Vector Embeddings and RAG

- Integrated OpenAI's text-embedding-3-small model for embeddings.
- Implemented semantic search using sklearn Nearest Neighbors.
- Created RAG pipeline to include top-k retrieved chunks as context in chatbot responses.
- Verified end-to-end functionality with document-grounded answers.

Task 4 – Agentic AI

- Added /agent endpoint powered by agentic_response.
- Enabled reasoning and planning for multi-step problem solving.
- Extended GUI to support "Agentic AI" queries with reasoning trace visualization.

Task 5 – Function-Calling Framework and Tool Integration

- Designed function-calling system for chatbot to invoke tools.
- Implemented example tools (e.g., data formatting, external API calls, retrieval).
- Built agent loop to detect when tool usage is required and return combined results.
- Extended frontend with "Agent Tools" page for explicit tool-based queries.

Challenges Faced

1. **Dependency Conflicts:** Encountered issues with `pytesseract==0.3.11` on Render (fixed by switching to a supported version).
2. **Streamlit–Backend Integration:** Initial errors such as `Cannot POST /chat` and `Cannot POST /upload` required fixing backend URL paths and ensuring correct routing.
3. **Deployment Debugging:** Required multi-stage fixes in Dockerfile (installing OCR libraries, handling environment variables).

Solutions Implemented

- Updated requirements to compatible versions.
- Aligned frontend API calls with FastAPI backend routes.
- Simplified GUI design to closely resemble ChatGPT's interface.
- Enhanced error handling for file processing and API requests.