GPT-2 Architecture Research and Application Integration

# 1. GPT-2: Core Algorithms and Architecture

## Transformer Architecture

GPT-2 is based on the decoder-only transformer architecture, which uses self-attention to capture dependencies across text sequences. The key components include:  
- Token Embeddings & Positional Encodings: Converts input text into numerical tokens and adds positional information to preserve word order.  
- Multi-Head Self-Attention: Computes attention weights to determine which words in the input are most relevant to each other.  
- Feed-Forward Networks (FFN): Applies non-linear transformations to enhance the feature representation after attention layers.  
- Residual Connections & Layer Normalization: Improves gradient flow and stabilizes training.  
- Output Softmax Layer: Predicts the probability distribution over the next token.

## Training Objective

- Language Modeling: GPT-2 is trained on large corpora to predict the next word given the previous sequence, using a standard causal language modeling objective (minimizing cross-entropy loss).  
- Fine-Tuning: For specialized tasks (e.g., recipe generation), GPT-2 can be fine-tuned on domain-specific datasets like RecipeNLG, where input ingredients are mapped to likely cooking steps or recipes.

# 2. How GPT-2 Integrates with the Application

For the Smart Refrigerator AI Project, GPT-2 serves as the core recipe generation engine, enhanced by Retrieval Augmented Generation (RAG) via LangChain.

## Integration Flow

1. User Input Layer (Streamlit Interface): Users manually input current refrigerator ingredients.  
2. RAG Layer (LangChain): Fetches nutritional data or similar recipes from external sources.  
3. GPT-2 Model: Fine-tuned on RecipeNLG dataset. Generates recipe suggestions based on ingredient inputs.  
4. Inventory Database (DuckDB/SQLite): Stores current fridge inventory and expiration data.  
5. Output Layer: Recipes and suggestions are displayed in the app. User feedback (accept/reject) is stored for iterative improvements.

# 3. Application Architecture Diagram

Here’s the architecture layout (simplified):  
  
 ┌─────────────────────────┐  
 │ User Input │  
 │ (Streamlit Interface) │  
 └──────────┬──────────────┘  
 │  
 ▼  
 ┌─────────────────────────┐  
 │ Inventory Database │  
 │ (DuckDB/SQLite/Parquet)│  
 └──────────┬──────────────┘  
 │  
 ▼  
 ┌─────────────────────────┐  
 │ Retrieval Augmented Gen │  
 │ (LangChain + Agents) │  
 └──────────┬──────────────┘  
 │  
 ▼  
 ┌─────────────────────────┐  
 │ GPT-2 Model (NLG) │  
 │ Fine-tuned on RecipeNLG │  
 └──────────┬──────────────┘  
 │  
 ▼  
 ┌─────────────────────────┐  
 │ Recipe Suggestions │  
 │ (UI Display + Feedback)│  
 └─────────────────────────┘

# 4. Key Points for Documentation

- GPT-2’s transformer-based design makes it ideal for recipe generation tasks requiring coherent text.  
- Integration with LangChain and RAG improves factual accuracy (e.g., nutritional data).  
- Architecture supports modular upgrades (e.g., adding vision-based models in the future).  
- Repository should include:  
 - docs/architecture\_gpt2.pdf (this research + diagram).  
 - Any supporting scripts for connecting GPT-2 to LangChain.