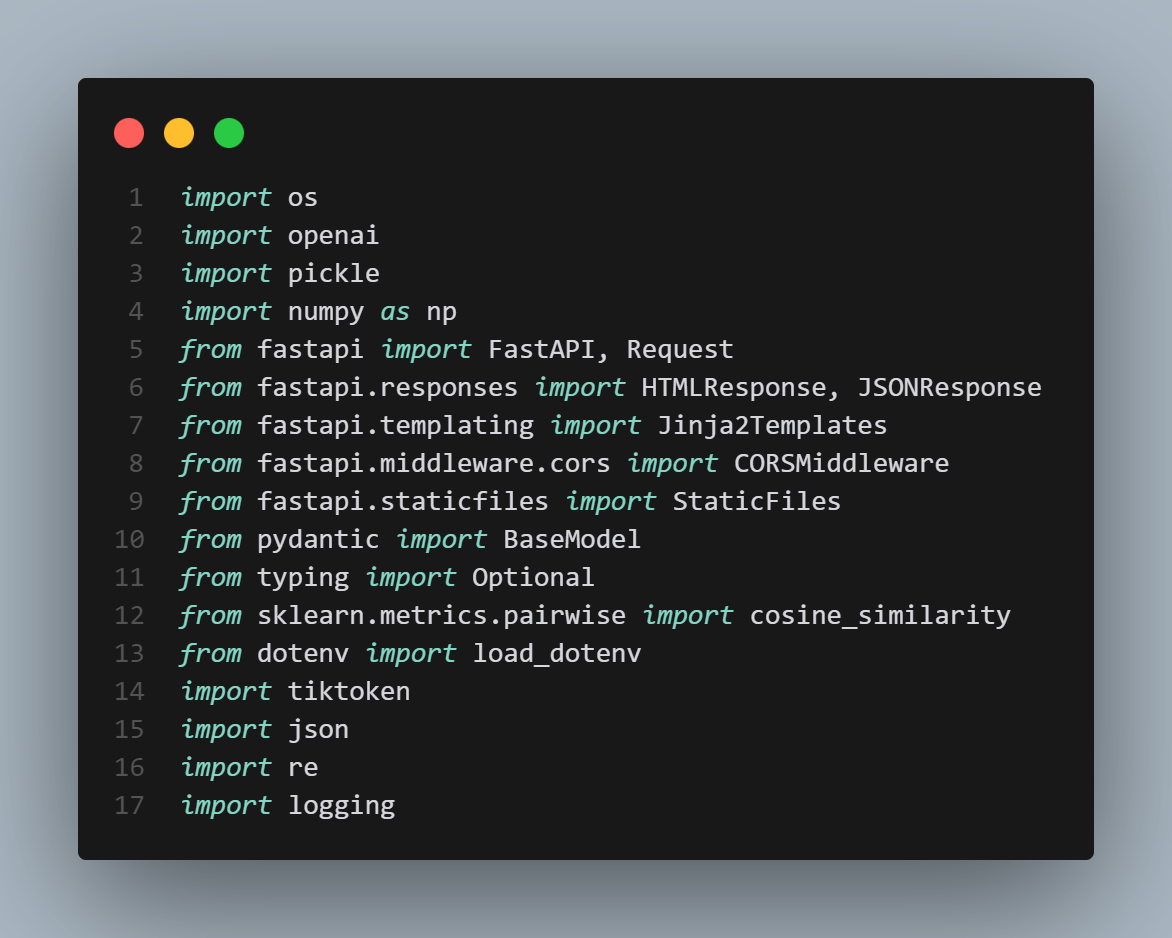
Baumalight AI Chatbot   
  
This is a Code file update initially from the start of main pages,   
  
1. Main.py,   
This snippet is importing section, where all the libraries and modules are imported  


2. Loading initially all the main files where we will be calling in the functions below,   
A screen shot of a computer program

AI-generated content may be incorrect.

3. This section to provide the model details, from basic config to model namingA screenshot of a computer program

AI-generated content may be incorrect.

4. This is only for token utilization verificationA computer screen shot of a program code

AI-generated content may be incorrect.

5. This is model api key details, and its corresponding embedding module detailsA computer screen shot of a black screen

AI-generated content may be incorrect.

6.Building vectors for model to recognize the knowledgebase, converting high level data into machine undersandable data   
A screen shot of a computer

AI-generated content may be incorrect.

7. creating chunks out of the vectorized data,   
A screenshot of a computer program

AI-generated content may be incorrect.

8. Summerize the answer when presented through API   
A screen shot of a computer screen

AI-generated content may be incorrect.

10. Provide a specific models which are present in the knowledgebase, in a list format so all the answers which needs to utilize can be utilized( this only runs once when the vectiorization of knowledgebase is happening)   
A screen shot of a computer program

AI-generated content may be incorrect.

11. Postal code regex function to find the nearest dealer   
A screenshot of a computer program

AI-generated content may be incorrect.

12. this function prints out the nearest dealer with its corresponding details  
A screenshot of a computer program

AI-generated content may be incorrect.

13. this function prints out in the format of how we need it  
A computer screen shot of a program code

AI-generated content may be incorrect.

14.this function is to rewrite the vector index from the start as there are no functions to rewrite the vector data   
A computer screen shot of a program code

AI-generated content may be incorrect.

15. This function provides the description for comparision answer,   
A screen shot of a computer screen

AI-generated content may be incorrect.

16. this is open ai response format function to verify the response and provide the underlying prompts   
A screen shot of a computer screen

AI-generated content may be incorrect.

17. clear pre tags function and add in response format functionality   
A screen shot of a computer screen

AI-generated content may be incorrect.

18. this functionality will make the code run on the desired port   
A screenshot of a computer program

AI-generated content may be incorrect.

Over view of all the functionality : **Core Architecture & Technology Stack**

**Backend Framework**

* **OpenAI Integration** - Uses GPT-4.1 for responses and text-embedding-ada-002 for embeddings
* **Vector Database** - Custom implementation using pickle files for storing embeddings
* **Token Management** - Tracks usage across prompts, completions, and embeddings

**Data Sources**

* **Product Files**: JSON files containing model comparisons, dealer information, and general Q&A
* **Vector Embeddings**: Pre-computed embeddings for semantic search
* **Static Assets**: CSS, JavaScript, and images for the web interface

**Key Functionality Categories**

**1. Product Information & Model Comparisons**

* **Model Data Management**: Handles TX, KR, and QC generator models
* **Detailed Specifications**: Power output, surge capacity, load capacity, voltage/phase, RPM, amperage
* **Pricing Information**: USD/CAD pricing with discount options and lead times
* **Side-by-Side Comparisons**: Direct model-to-model comparisons with power analysis
* **Model Count Queries**: "How many TX models are listed?" type questions

**2. Dealer Location Services**

* **Postal Code Recognition**: Supports Canadian, US, and Indian postal codes
* **Geographic Matching**: Finds nearest dealers based on postal code similarity
* **Dealer Information**: Company details, addresses, phone numbers, emails
* **Location-Based Queries**: "Find dealers near me" functionality

**3. Conversational AI Features**

* **Context-Aware Conversations**: Maintains conversation history per client IP
* **Model Context Tracking**: Automatically references previously discussed models
* **Greeting Detection**: Handles basic greetings and provides helpful links
* **Query Classification**: Distinguishes between product questions, dealer requests, and comparisons

**4. Semantic Search & Information Retrieval**

* **Vector Embedding Search**: Uses cosine similarity for finding relevant content
* **Chunked Text Processing**: Splits documents into manageable chunks with overlap
* **Multi-Source Search**: Searches across product files, comparisons, and general answers
* **Relevance Scoring**: Ranks search results by similarity to user query

**5. Response Formatting & Presentation**

* **HTML Response Generation**: Formats responses with proper HTML structure
* **Product Specification Formatting**: Organizes specs into shipping, pricing, and description sections
* **Link Integration**: Provides relevant product links to Baumalight website
* **Clean Text Processing**: Removes JSON artifacts and formatting issues

**6. Data Management & Storage**

* **Conversation History**: Stores chat history in both JSON and TXT formats
* **Token Usage Tracking**: Monitors API usage for cost management
* **Embedding Index Management**: Rebuildable vector index for search functionality
* **File Processing**: Handles multiple file formats (TXT, JSON, MD)

**7. Web Interface Features**

* **Static File Serving**: CSS, JavaScript, and image assets
* **Template Rendering**: Jinja2 templates for HTML responses
* **CORS Support**: Cross-origin resource sharing enabled
* **Responsive Design**: Modern UI with product cards and dealer information

**8. Advanced Query Processing**

* **Model Extraction**: Regex-based extraction of specific model numbers
* **Power Requirement Matching**: Finds models based on KW requirements
* **Comparison Detection**: Identifies when users want to compare specific models
* **Price Query Handling**: Direct pricing information for specific models

**9. Error Handling & Logging**

* **Comprehensive Logging**: Detailed logging of API calls, embeddings, and errors
* **Graceful Fallbacks**: Fallback responses when specific data isn't found
* **Exception Management**: Proper error handling throughout the application
* **Data Validation**: Input validation and sanitization

**10. Performance Optimizations**

* **Token Counting**: Efficient token usage tracking
* **Chunked Processing**: Optimized text chunking for large documents
* **Caching**: Embedding storage for faster retrieval
* **History Management**: Limited conversation history to prevent memory issues

**API Endpoints**

1. **GET /** - Main web interface
2. **POST /ask** - Primary chatbot endpoint
3. **POST /rebuild-index** - Rebuilds the vector search index

**Data Flow**

1. **User Input** → Query classification and preprocessing
2. **Semantic Search** → Vector embedding search across product data
3. **Context Building** → Combines search results with conversation history
4. **OpenAI Processing** → Generates contextual responses
5. **Response Formatting** → Formats output with HTML and links
6. **History Storage** → Saves conversation for future context

This chatbot provides a comprehensive solution for Baumalight's customer support needs, combining product information, dealer location services, and intelligent conversation management in a single, well-architected application.