OwlAI – Complete Backend Documentation

Introduction  
  
OwlAI is an intelligent conversational assistant designed to help students prepare effectively for the UGC NET Paper 1 examination. This documentation provides an in-depth look at the backend architecture, detailing each file, its functions, and how they work together to deliver a seamless, adaptive, and contextually aware chatbot experience.  
  
Project Structure and Detailed Explanation  
  
1. Intent, Tone, and Language Classification  
- intent\_classifier.py  
 - Function: classify\_intent\_tone\_language(query)  
 - Classifies user's query into intent, tone, and language using regex and keywords.  
- intent\_tone\_classifier.py  
 - Function: classify\_intent\_tone\_language(query)  
 - Uses regex with GPT-3.5 fallback for accuracy.  
- language\_detect.py  
 - Function: detect\_language\_hint(query)  
 - Quick heuristic-based check for language classification.

***KEYWORDS:***

HINDI CHARACTERS = re.findall(r'[\u0900-\u097F]', q)

ENGLISH CHARACTERS = re.findall(r'[a-zA-Z]', q)

HINDI WORDS = ["hai", "kya", "nahi", "karna", "kyun", "tum", "mera", "batao", "shuru"]

***TONE DETECTION:***

SIMPLE TONE = ["simple terms", "like a kid", "easy way"]

DETAILED TONE = ["professor", "deep dive", "in detail"]

EMOTIONAL TONE = ["feeling low", "can't do this", "tired", "hopeless"]

CASUAL TONE = ["hi", "hello", "namaste", "kaise ho"]

2. Moderation System  
- moderation.py  
 - Function: run\_moderation\_check(query)  
 - Matches queries against unsafe phrases ensuring conversatioaanal safety.  
- response\_control.yaml  
 - Stores trigger phrases and safe fallback responses.  
  
3. Retrieval-Augmented Generation (RAG)  
- embedder.py  
 - Function: get\_embedding(text)  
 - Converts text into semantic embeddings using MiniLM.  
- vector\_search.py  
 - Function: query\_vector\_store(query\_text, top\_k, user\_id)  
 - Retrieves relevant content from Pinecone.  
- chunk\_push.py  
 - Chunks syllabus, books, and PYQs into semantic chunks for Pinecone.  
- create\_pinecone.py  
 - Initializes the Pinecone index.

4. Prompt Engineering and Persona Customization  
- llm.py  
 - Function: build\_persona\_prompt(language, tone, intent, last\_topic)  
 - Constructs the system prompt defining OwlAI's speaking style.  
 - Function: build\_prompt(query, history, context, session\_id, user\_id, intent, tone, language)  
 - Combines query, history, and context into structured GPT prompts.  
 - Function: get\_response\_from\_llm(prompt)  
 - Generates responses using GPT-3.5.  
  
5. Quiz Generation and Management  
- quiz\_generator.py  
 - Functions: generate\_quiz\_questions(topic), parse\_quiz\_from\_text(text)  
 - Generates quizzes, parses GPT-generated quiz content.  
- session\_memory.py  
 - Functions: start\_quiz(session\_id, topic), get\_quiz\_state(session\_id), update\_quiz\_state(session\_id, quiz\_data)  
 - Manages quiz state and progression.  
- ask\_route.py  
 - Handles quiz interactions and feedback.  
  
6. Auto-Learning from User Feedback  
- auto\_learn.py  
 - Function: retry\_failed\_answer(chat\_id)  
 - Improves answers based on negative feedback using GPT.  
  
7. Chat and Session Management  
- chats.py  
 - Functions: save\_chat(), get\_chat\_history()  
 - Manages conversations in Firestore.  
- chat\_route.py  
 - API endpoints for chat operations.  
- session\_route.py  
 - Manages session lifecycle.  
- session\_memory.py  
 - Real-time session management.  
  
8. Main API Routing and User Management  
- main.py  
 - Initializes FastAPI and route management.  
- user\_route.py  
 - Manages user account lifecycle.  
- ask\_route.py  
 - Core logic for processing queries.

9. Professional Query Handling  
- professional\_handler.py  
 - Provides authoritative responses regarding bot sources, privacy, and creators.

System Architecture Flow  
1. User Query -> FastAPI (ask\_route.py)  
2. Intent, Tone, Language Detection  
3. Moderation Check  
4. Session Memory and Context Retrieval  
5. Vector Search (RAG)  
6. Prompt Construction and Persona Setting  
7. GPT-3.5 Response  
8. Quiz and Feedback Management  
9. Response Stored in Firestore  
10. Final Response to User  
  
Technologies Used  
- OpenAI GPT-3.5 Turbo  
- FastAPI  
- Firebase Firestore  
- Pinecone Vector Database  
- SentenceTransformers (MiniLM)