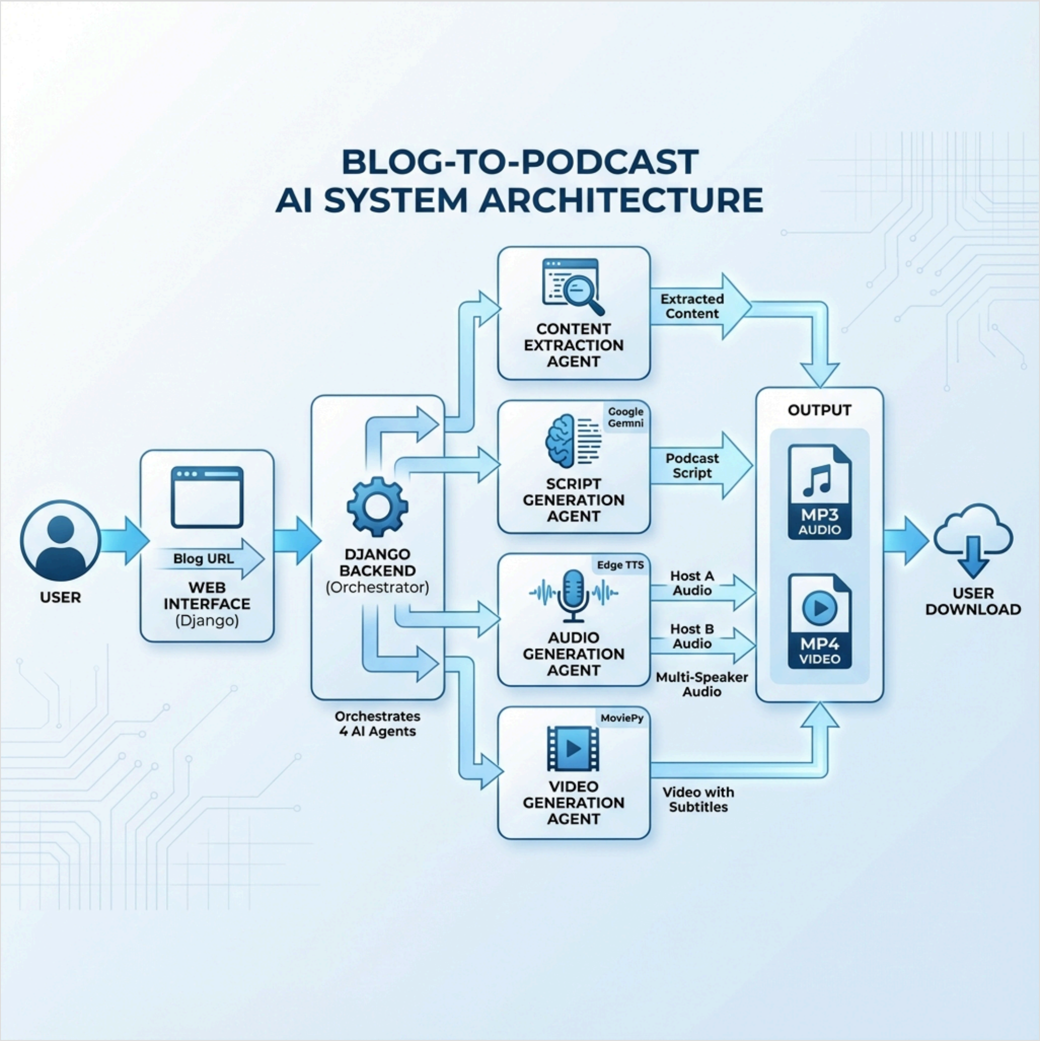
Blog-to-Podcast Architecture

Overview

The **Blog-to-Podcast** system converts blog posts into multi-speaker video podcasts using 4 AI agents orchestrated by a Django backend. It handles content extraction, script writing, audio synthesis, and video production.

Architecture Diagram

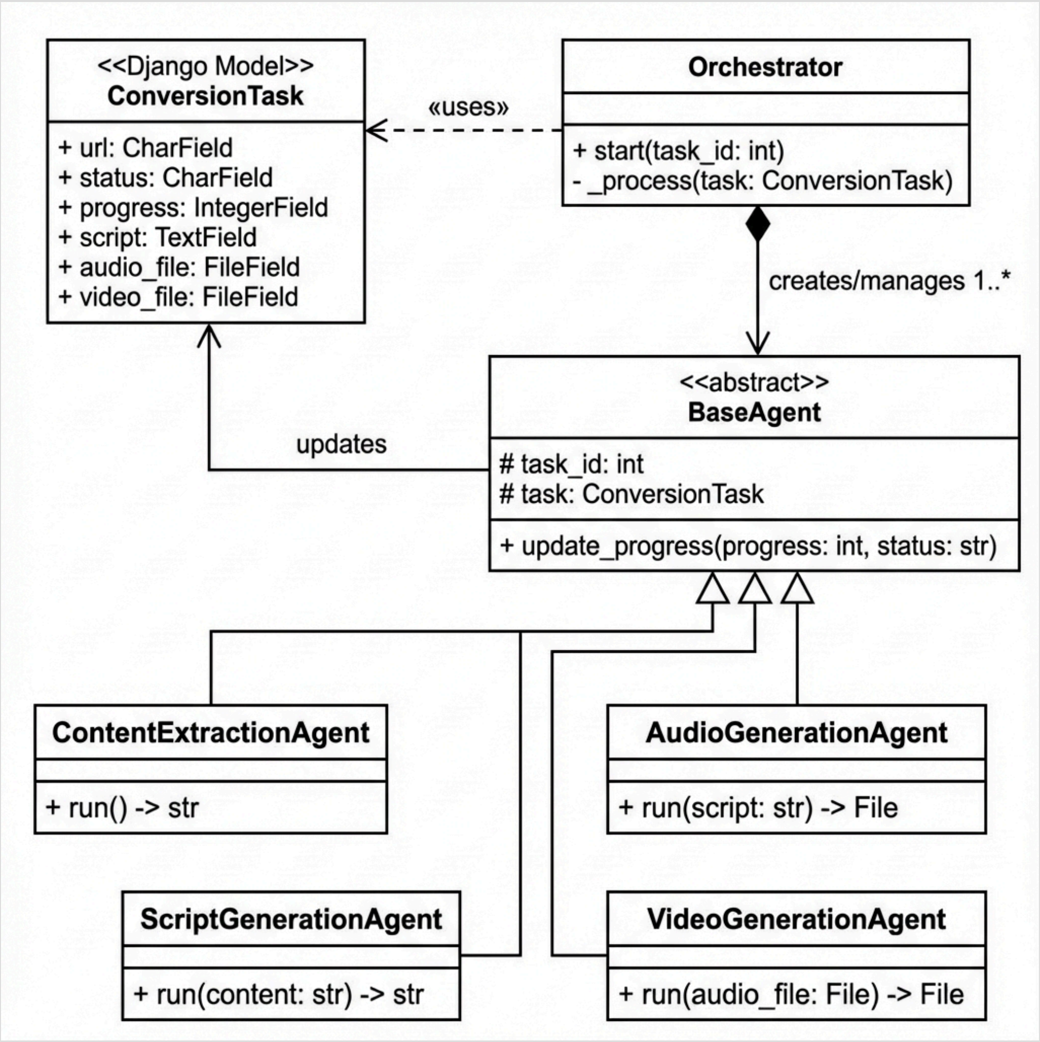


Use Cases

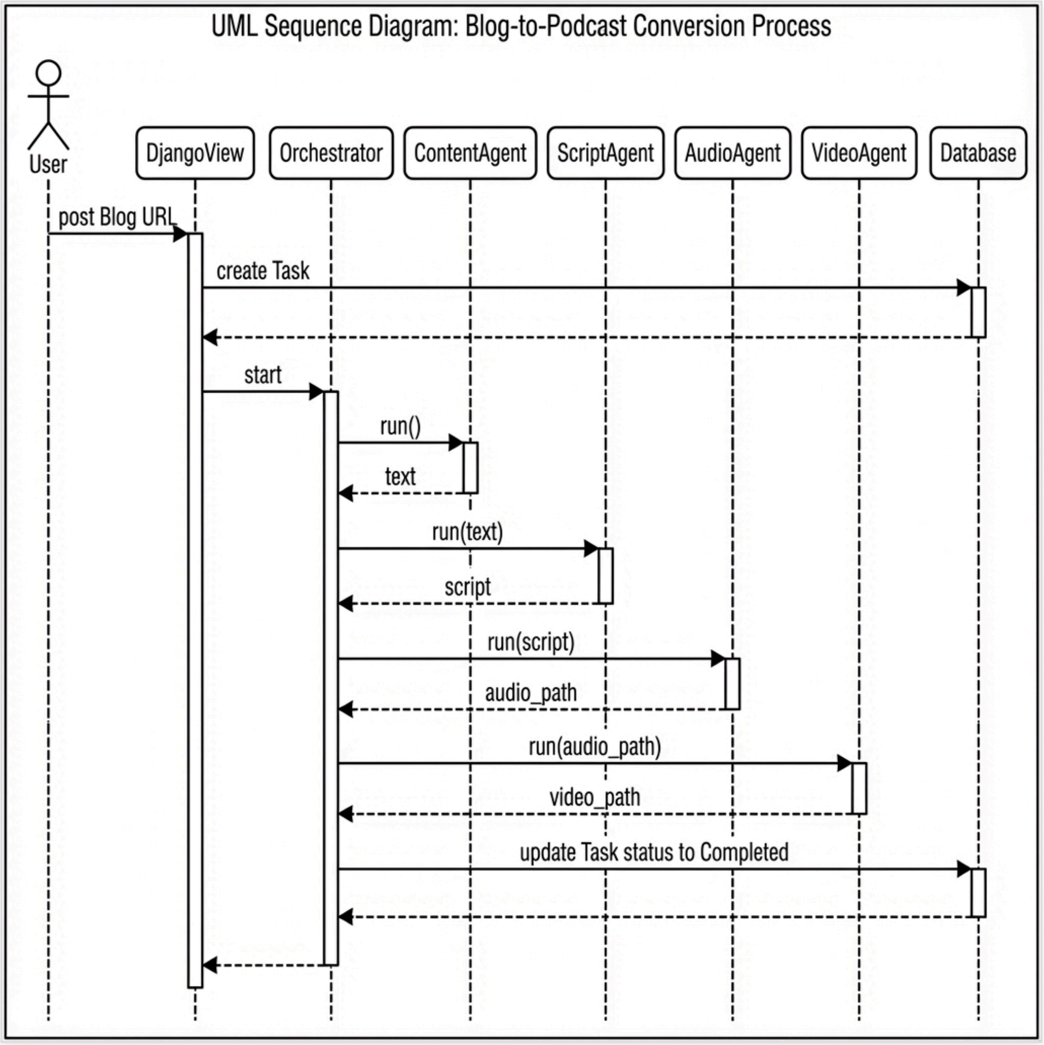
1. **Content Repurposing**: Convert written blogs into engaging audio/video content for YouTube or Spotify.
2. **Accessibility**: Make text-heavy content accessible to users who prefer listening or watching.
3. **Learning on the Go**: Allow users to consume educational articles while commuting or exercising.
4. **Multi-Language Support**: (Future) Translate blogs into different languages with localized voices.

UML Diagrams

Class Diagram



Sequence Diagram



System Components

- **Django Backend**: Manages `ConversionTask` lifecycle and serves the web UI.
- **Orchestrator**: Runs in a background thread to sequentially trigger agents.
- **Content Extraction Agent**: Scrapes and cleans blog text using `BeautifulSoup` .
- **Script Generation Agent**: Uses **Gemini 2.0 Flash** to write conversational scripts.
- **Audio Generation Agent**: Synthesizes speech with **Edge TTS** (Host A/B) and maps timing.
- **Video Generation Agent**: Creates video with **MoviePy**, syncing audio and subtitles.

Technology Stack

- **Core**: Django 5.x, Python 3.11+
- **AI/ML**: Google Gemini 2.0 Flash, Microsoft Edge TTS
- **Media**: MoviePy, Pillow, NumPy
- **Data**: SQLite, BeautifulSoup4

Deployment Guide

Prerequisites

- Python 3.11+
- Google API Key (Gemini)
- 4GB RAM minimum

Installation

```
 # Clone and setup
git clone <repo-url>
cd blog-to-podcast
python -m venv venv
venv\Scripts\activate
pip install -r requirements.txt
```

```
# Configure
echo "GOOGLE_API_KEY=your_key" > .env

# Run
python manage.py migrate
python manage.py runserver
```

Production

- Use **Gunicorn** or **uWSGI**
- Set `DEBUG=False`
- Configure **Nginx** reverse proxy
- Use **PostgreSQL** for production DB

Security Considerations

- **API Keys**: Store in environment variables, never commit to Git
- **Input Validation**: Sanitize URLs to prevent SSRF attacks
- **Rate Limiting**: Implement per-user request limits
- **CORS**: Configure allowed origins for API access
- **File Storage**: Validate file types, implement size limits

Performance Metrics

Metric	Value
Avg Processing Time	2-3 minutes per blog
Max Concurrent Tasks	5 (configurable)
Audio Generation	~30s per minute of speech
Video Rendering	~45s per minute
Memory Usage	500MB-1GB per task

API Documentation

POST /convert/

Request:

```
{
  "url": "https://example.com/blog-post"
}
```

Response:

```
{
  "task_id": "uuid",
  "status": "PENDING"
}
```

GET /status/{task_id}/

Response:

```
{
  "status": "COMPLETED",
  "progress": 100,
  "audio_url": "/media/podcast_uuid.mp3",
  "video_url": "/media/podcast_uuid.mp4"
}
```

Troubleshooting

Issue	Solution
"No content found"	Check URL accessibility, ensure blog has text content
Audio generation fails	Verify Edge TTS is accessible, check network
Video rendering slow	Reduce video resolution, optimize subtitle count
API key error	Verify `GOOGLE_API_KEY` in `.env` file

Test Cases

ID	Scenario	Input	Expected Outcome
TC01	**Valid Blog URL**	`https://example.com/blog-post`	Status: `COMPLETED` , MP3/MP4 files generated.
TC02	**Invalid URL**	`https://invalid-url.com`	Status: `FAILED` , Error message logged.
TC03	**Empty Content**	URL with no text	Status: `FAILED` , "No content found" error.
TC04	**Long Content**	5000+ word article	Script truncated/summarized, Audio < 10 mins.
TC05	**Concurrent Requests**	2+ users submit URLs	Tasks queued/processed in parallel threads.