Presentation Builder Backend Documentation

Overview

The Presentation Builder Backend is a Flask-based web service that generates professional presentations using AI. This backend leverages OpenAI's language models to create structured presentation plans and content, then converts them into PDF and PowerPoint formats for educational and professional use.

System Architecture

The application follows a modular architecture with clear separation of concerns:

- **Application Core**: Flask application factory pattern for configuration and setup
- **RESTful API**: Endpoints for plan generation, content creation, and file output
- **Service Layer**: Components for interacting with AI models and generating documents
- **Utility Layer**: Helper functions for file operations and common tasks

Project Structure

```
presentation-builder/
├── app/
│   ├── __init__.py           # Flask application factory
│   ├── routes.py             # API endpoint definitions
│   ├── services/
│   │   ├── __init__.py       # Service imports
│   │   ├── plan_generator.py # Generates presentation plan
│   │   ├── content_generator.py # Creates section content
│   │   ├── pdf_generator.py  # PDF output generation
│   │   └── pptx_generator.py # PowerPoint output generation
│   ├── utils/
│   │   ├── __init__.py
│   │   └── helpers.py        # Utility functions
│   └── static/
│       └── ODC_logo.jpeg     # Logo for presentations
├── config.py                 # Application configuration
├── requirements.txt          # Dependencies
└── run.py                    # Entry point
```

API Endpoints

Health Check

- **GET /health**
  - Confirms the API is running
  - Response: `{"status": "OK", "message": "Presentation Builder API is running"}`

File Upload

- **POST /api/upload**
  - Uploads files such as logos
  - Form-data with file field
  - Returns file path information

Generate Plan

- **POST /api/generate-plan**
  - Creates a structured presentation plan
  - Request body:

```
{
  "domaine": "Java and Spring Boot",
  "sujet": "Introduction to Spring Boot",
  "description_sujet": "Overview of Spring Boot for beginners",
  "niveau_apprenant": "Beginners"
}
```

  - Response:

```
{
  "success": true,
  "execution_time_seconds": 3.45,
  "plan": {
    "titre": "Introduction to Spring Boot",
    "sections": [
      {
        "section": "What is Spring Boot?",
        "sous-sections": [
          "Definition and origins",
          "Advantages over traditional Spring"
        ]
      },
      ...
    ]
  }
}
```

Generate Content

- **POST /api/generate-content**
  - Creates detailed content for each section in the plan
  - Request body:

```
{
  "domaine": "Java and Spring Boot",
  "sujet": "Introduction to Spring Boot",
  "plan": { ... } // Plan object from generate-plan response
}
```

  - Response:

```
{
  "success": true,
  "execution_time_seconds": 15.67,
  "content": [ ... ] // Detailed content for all sections
}
```

Generate Files

- **POST /api/generate-files**
  - Creates PDF and/or PowerPoint files from the content
  - Request body:

```
{
  "sujet": "Introduction to Spring Boot",
  "contenu": [ ... ], // Content from generate-content response
  "format": "both", // "pdf", "pptx", or "both"
  "trainer_name": "Your Name"
}
```

  - Response:

```
{
  "success": true,
  "execution_time_seconds": 5.23,
  "files": [
    {
      "type": "pdf",
      "filename": "Introduction_to_Spring_Boot_1234567890.pdf",
      "path": "/app/static/outputs/Introduction_to_Spring_Boot_1234567890.pdf",
      "download_url": "/api/download/Introduction_to_Spring_Boot_1234567890.pdf"
    },
    {
      "type": "pptx",
      "filename": "Introduction_to_Spring_Boot_1234567890.pptx",
      "path": "/app/static/outputs/Introduction_to_Spring_Boot_1234567890.pptx",
      "download_url": "/api/download/Introduction_to_Spring_Boot_1234567890.pptx"
    }
  ]
}
```

Download Files

- **GET /api/download/{filename}**
  - Downloads a generated file
  - Returns the file as an attachment

Core Components

Plan Generator

The `plan_generator.py` module creates a structured presentation outline with sections and subsections based on the given domain, subject, and target audience level. It uses AI to ensure the plan is pedagogically sound and properly organized.

Content Generator

The `content_generator.py` module expands each section and subsection with detailed explanations, examples, code snippets, and tables as appropriate. It creates rich, educational content tailored to the audience level.

PDF Generator

The `pdf_generator.py` module converts the generated content into a professional PDF presentation with proper formatting, styling, and branding elements like logos.

PowerPoint Generator

The `pptx_generator.py` module creates a PowerPoint presentation from the content, including title slides, properly formatted content slides, and styling consistent with the brand.

Configuration Options

The application is configurable through `config.py` and environment variables:

- **API_KEY**: OpenAI API key for generating content
- **BASE_URL**: API endpoint for the language model
- **MODEL**: Specific language model to use
- **UPLOAD_FOLDER**: Directory for uploaded files
- **OUTPUT_FOLDER**: Directory for generated presentations
- **ALLOWED_EXTENSIONS**: File types allowed for upload
- **MAX_CONTENT_LENGTH**: Maximum file size for uploads

Setup and Installation

1. Clone the repository
2. Create and activate a virtual environment
3. Install dependencies: `pip install -r requirements.txt`
4. Set up environment variables:

```
export FLASK_APP=run.py
export FLASK_ENV=development
export OPENAI_API_KEY=your-api-key-here
```

5. Run the application: `python run.py`

Error Handling

The API includes robust error handling:

- Input validation for required fields
- Try-except blocks for external API calls
- Detailed error messages in responses
- Logging for debugging purposes

Performance Considerations

- AI content generation can take time depending on the complexity and length
- Larger presentations may require more memory and processing power
- Consider implementing caching for frequently generated content
- For high-load environments, consider adding request queuing

Security Considerations

- API keys are stored in configuration and should be kept secure

- Uploaded files are validated for type and size
- Filenames are sanitized to prevent directory traversal attacks
- Consider implementing authentication for production use

Extending the Application

The modular design allows for easy extension:

- Add new output formats by creating additional generator services
- Support more customization options in the request
- Implement user authentication and presentation storage
- Add template selection for different presentation styles

This documentation provides a comprehensive overview of the Presentation Builder Backend. For specific implementation details, refer to the source code and comments within each module.