Structural Design Patterns

Definition:

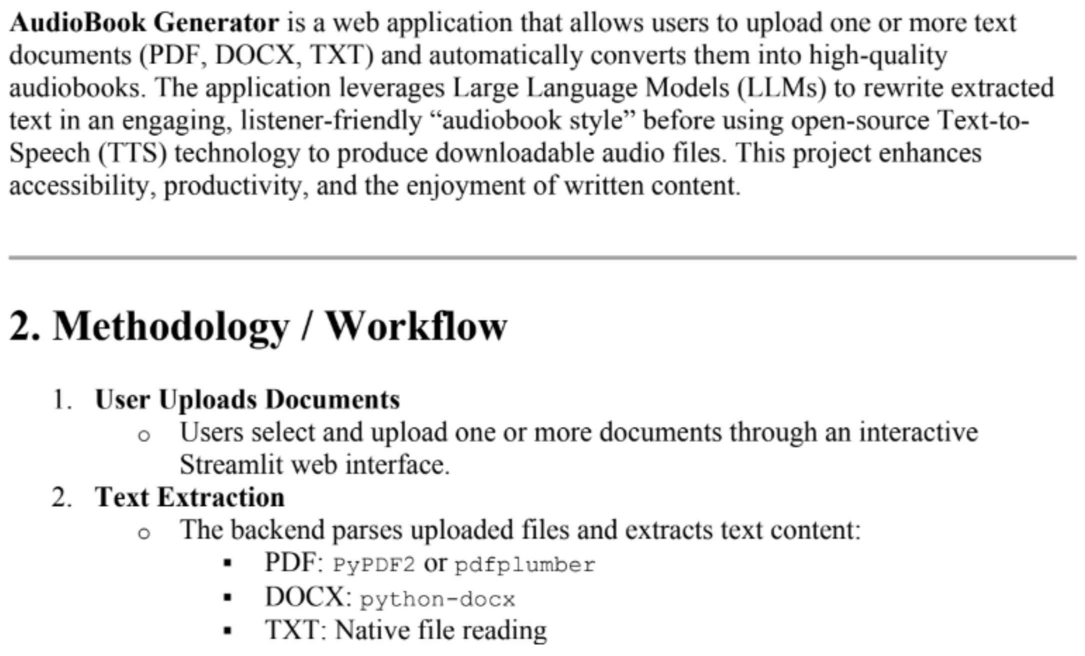
Structural design patterns focus on how classes and objects are combined to create larger, more complex structures. They simplify structure by identifying relationships.

Key Points:

- Concerned with class and object composition.

- Simplifies relationships and improves maintainability.

- Often used for scalable, reusable designs.



AudioBook Generator is a web application that allows users to upload one or more text documents (PDF, DOCX, TXT) and automatically converts them into high-quality audiobooks. The application leverages Large Language Models (LLMs) to rewrite extracted text in an engaging, listener-friendly "audiobook style" before using open-source Text-to-Speech (TTS) technology to produce downloadable audio files. This project enhances accessibility, productivity, and the enjoyment of written content.

---

2. Methodology / Workflow

1. **User Uploads Documents**
   - Users select and upload one or more documents through an interactive Streamlit web interface.
2. **Text Extraction**
   - The backend parses uploaded files and extracts text content:
     - PDF: `PyPDF2` or `pdfplumber`
     - DOCX: `python-docx`
     - TXT: Native file reading

Components:

- Adapter – allows incompatible interfaces to work together.

- Bridge – separates abstraction from implementation.

- Composite – treats individual and composite objects uniformly.

- Decorator – adds functionality dynamically.

- Facade – provides simplified interface to a complex system.

- Flyweight – shares common data to save memory.

- Proxy – placeholder for another object.

Example:
Example: A facade interface to simplify access to multiple subsystems in a library.

Advantages:
- Simplifies system structure.

- Encourages reuse.

- Improves flexibility by decoupling components.

Behavioral Design Patterns

Definition:
Behavioral design patterns focus on how objects interact and communicate with each other to carry out specific tasks.

Key Points:
- Defines common communication patterns.

- Encapsulates behaviors for flexibility.

- Promotes loose coupling.

Components:
- Observer – one-to-many dependency.

- Strategy – chooses algorithm behavior at runtime.

- Command – encapsulates requests as objects.

- Mediator – controls communication between objects.

- Template Method – skeleton of an algorithm.

Example:
Example: Observer pattern in a chat app where new messages update all active user screens.

Advantages:
- Encourages flexibility.

- Easier maintenance.

- Supports dynamic behavior changes.

Abstract Factory Pattern

Definition:
Creates families of related or dependent objects without specifying their concrete classes.

Key Points:
- Higher-level abstraction than Factory Method.

- Switch between product families easily.

- Promotes consistent product creation.

Components:
- Abstract Factory – blueprint for creating products.

- Concrete Factory – implements creation rules.

- Abstract Product – defines common interfaces.

- Concrete Product – actual implementations.

- Client – uses factory without knowing concrete classes.

Example:
Example: Regional car manufacturing factories for Europe and North America.

Advantages:
- Ensures consistency among products.

- Simplifies switching between product families.

Builder Design Pattern

Definition:
Constructs complex objects step-by-step, allowing different representations from the same process.

Key Points:
- Separates construction from representation.

- Useful when objects have many optional parts.

Components:
- Product – final complex object.

- Builder – defines construction steps.

- Concrete Builder – implements steps.

- Director – controls construction order.

- Client – initiates building process.

Example:
Example: Building custom computers with different parts.

Advantages:
- Provides control over construction.

- Allows creating different variations easily.

Prototype Design Pattern

Definition:
Creates new objects by cloning existing ones instead of building from scratch.

Key Points:
- Saves time and resources.

- Uses clone() method for duplication.

- Supports runtime object creation.

Components:
- Prototype Interface – declares clone method.

- Concrete Prototype – implements clone.

- Client – requests new objects via cloning.

Example:
Example: Drawing app cloning shapes to create variations.

Advantages:
- Reduces object creation cost.

- Simplifies creation of many similar instances.

Singleton Design Pattern

Definition:
Ensures only one instance of a class exists and provides a global access point.

Key Points:
- Centralized control.

- Can use lazy or eager initialization.

- Must be thread-safe in multi-threaded apps.

Components:

- Private constructor – restricts direct creation.

- Static method – provides global access.

- Single instance – reused throughout.

Example:

Example: Database connection manager.

Advantages:

- Reduces memory usage.

- Centralized configuration.

- Prevents conflicting instances.

Observer Design Pattern

Definition:

Defines a one-to-many relationship between objects where changes in one (subject) update all dependents (observers).

Key Points:

- Supports event-driven systems.

- Decouples subject from observers.

Components:

- Subject – manages observers.

- Concrete Subject – holds state, notifies observers.

- Observer – defines update method.

- Concrete Observer – reacts to changes.

Example:

Example: Stock price tracker updating multiple dashboards.

Advantages:

- Loose coupling between components.

- Easy to add/remove observers.

- Supports dynamic subscriptions.