Low Level Design Document

Introduction

This Low Level Design (LLD) document outlines the core architecture and implementation details for **StoryWeaver - AI Creative Writing Workshop**. StoryWeaver is a collaborative platform enabling users to co-author stories with generative AI, featuring real-time suggestions, version control, peer feedback, and exportable manuscripts. The system emphasizes modularity, secure user management, and support for multiple genres and languages.

---

1. System Components

Component	Technology	Key Responsibilities
Frontend	React	UI/UX, real-time collaboration, feedback, export
Backend API	Python (FastAPI)	Auth, business logic, AI orchestration, data access
AI Service	Python	Generate suggestions (plot, dialogue, style)
Database	PostgreSQL	Store users, stories, versions, feedback
Auth Service	JWT/OAuth	User authentication & authorization

---

2. Class/Interface Overview

Class/Interface	Description	Key Methods/Attributes
`User`	User profile & permissions	`id`, `username`, `email`, `role`
`Story`	Story metadata & content	`id`, `title`, `genre`, `language`, `authors`, `current_version`
`StoryVersion`	Versioned story content	`id`, `story_id`, `content`, `timestamp`, `author_id`
`Feedback`	Peer feedback on stories/versions	`id`, `story_id`, `user_id`, `comment`, `rating`
`AIService` (interface)	AI suggestion engine	`generate_suggestion(context, type)`
`AuthManager`	Auth/session management	`login()`, `logout()`, `validate_token()`

Relationships:

- `User` ↔ `Story` (many-to-many, co-authors)
- `Story` ↔ `StoryVersion` (one-to-many)
- `User` ↔ `Feedback` (one-to-many)

- `Story` ↔ `Feedback` (one-to-many)

---

3. Data Structure Overview

Table/Model	Fields
`users`	`id`, `username`, `email`, `password_hash`, `role`, `created_at`
`stories`	`id`, `title`, `genre`, `language`, `created_at`, `updated_at`
`story_authors`	`story_id`, `user_id`
`story_versions`	`id`, `story_id`, `content`, `author_id`, `timestamp`
`feedback`	`id`, `story_id`, `user_id`, `comment`, `rating`, `created_at`

---

4. Algorithms/Logic

AI Suggestion Flow (Pseudocode):

```
def generate_suggestion(story_id, user_id, context, suggestion_type):
    if not user_has_access(user_id, story_id):
        raise PermissionError
    prompt = build_prompt(context, suggestion_type)
    suggestion = ai_model.generate(prompt)
    return suggestion
```

Version Control (Summary):

- On story edit, create new `StoryVersion` with diff.
- Allow rollback to previous versions.
- Track author and timestamp for each version.

---

5. Error Handling

Scenario	Handling Approach
Unauthorized access	Return 401/403, log attempt
Invalid input/data	Return 400, validate on both frontend/backend
AI service unavailable	Fallback to cached suggestions, notify user
Database errors	Return 500, log error, retry if safe
Version conflict (edit)	Prompt user to resolve, merge or overwrite

---

End of Document