Product Requirements Document

"TheFabled.ai" Application

---

Version Information

Version: 1.0

1. Introduction

Purpose

The "TheFabled.ai" application is an innovative platform that allows users to engage in immersive conversations with characters from their favorite books. Users can upload a book in PDF or EPUB format, and the application analyzes the text to extract detailed character profiles using advanced language models. The core application logic focuses on character extraction, analysis, and interaction simulation, providing realistic dialogues that reflect each character's unique personality and experiences.

This Product Requirements Document (PRD) outlines the detailed functional and non-functional requirements for the development of the core application. The PRD serves as a foundational guide for building both the backend services and the front-end interface, ensuring seamless integration and fulfilling user needs.

Scope

This PRD covers:

The core backend application logic for file processing, text extraction, character analysis, and interaction simulation.

API specifications for integration with front-end applications and other functionalities.

Detailed functional and non-functional requirements.

System architecture, including components, data flow, and algorithms.

Considerations for future front-end development and integration.

Definitions, Acronyms, and Abbreviations

PRD: Product Requirements Document

AI: Artificial Intelligence

LLM: Large Language Model

API: Application Programming Interface

UI: User Interface

UX: User Experience

PDF: Portable Document Format

EPUB: Electronic Publication

---

2. Overall Description

Product Perspective

The " TheFabled.ai " application is designed as a backend service that provides APIs for front-end clients (web, mobile, etc.) to interact with. It leverages external AI services to perform complex text processing and analysis. The application can be integrated into various front-end platforms, allowing for flexibility in UI design and user experience.

Product Functions

File Upload and Processing: Accepts book files in PDF or EPUB formats and extracts text content.

Character Extraction and Analysis: Performs multi-pass analysis to identify significant characters and build detailed profiles.

Interaction Simulation: Generates authentic responses from characters based on user inputs.

API Services: Exposes endpoints for front-end applications to interact with the core functionalities.

Session Management: Manages user sessions, including uploaded files, analysis results, and conversation history.

User Characteristics

Literature Enthusiasts: Individuals who wish to engage more deeply with characters from books.

Students and Educators: Users looking for educational tools to study literature and character development.

Writers and Authors: Individuals interested in character dynamics and dialogues for inspiration.

Users are expected to be familiar with basic web or mobile applications.

---

3. Specific Requirements

Functional Requirements

3.1. File Upload and Processing

FR1: The system shall provide an API endpoint to receive book files in PDF or EPUB formats.

FR1.1: The endpoint shall accept files up to a configurable size limit (e.g., 100 MB).

FR1.2: The system shall validate the file format and return appropriate error messages for unsupported formats.

FR2: The system shall extract text content from uploaded book files.

FR2.1: For PDF files, utilize a PDF parsing library (e.g., PyPDF2, pdfminer).

FR2.2: For EPUB files, use a parsing library (e.g., ebooklib) to extract text.

FR2.3: Clean and normalize extracted text to prepare it for analysis.

FR3: The system shall handle large texts efficiently.

FR3.1: Implement text splitting into manageable chunks for processing.

FR3.2: Provide mechanisms to monitor and manage long-running processing tasks.

3.2. Character Extraction and Analysis

FR4: The system shall perform multi-pass character analysis on the extracted text.

First Pass (FR4.1): Identify significant characters and their basic profiles.

FR4.1.1: Extract characters' names, roles, plot importance, importance levels, and key relationships.

FR4.1.2: Use the primary AI model for analysis.

FR4.1.3: Implement a fallback mechanism using an alternative model or method in case of failure.

Second Pass (FR4.2): Generate detailed character descriptions.

FR4.2.1: Analyze physical appearance, personality traits, psychological aspects, background, and context.

FR4.2.2: Create an LLM persona prompt for each character.

Third Pass (FR4.3): Analyze character depth.

FR4.3.1: Outline character development arcs, provide examples of personality traits, and extract memorable quotes.

FR4.3.2: Handle cases where certain information is not available gracefully.

FR5: The system shall store analysis results for later retrieval.

FR5.1: Save results in a structured format (e.g., JSON, database).

FR5.2: Associate analysis results with user sessions or identifiers.

3.3. Interaction Simulation

FR6: The system shall provide an API for initiating conversations with characters.

FR6.1: Allow clients to specify the character they wish to interact with.

FR6.2: Generate initial prompts based on the character's profile.

FR7: The system shall process user inputs and generate character responses.

FR7.1: Accept user messages via an API endpoint.

FR7.2: Use the character's persona and conversation history to generate responses.

FR7.3: Ensure responses are authentic to the character's personality and knowledge.

3.4. Session Management

FR8: The system shall manage user sessions.

FR8.1: Assign unique session identifiers to users.

FR8.2: Store session data including conversation history and selected character.

FR8.3: Provide APIs to retrieve and manage session data.

3.5. API Services

FR9: The system shall expose FAST APIs for all functionalities.

FR9.1: Provide endpoints for file upload, analysis initiation, character selection, and conversation handling.

FR9.2: Utilize standard HTTP methods and status codes.

FR9.3: Ensure APIs are well-documented using tools like OpenAPI/Swagger.

3.6. Error Handling

FR10: The system shall handle errors gracefully and provide meaningful responses.

FR10.1: Return appropriate HTTP status codes and error messages for API requests.

FR10.2: Log errors internally for troubleshooting.

3.7. Security and Authentication

FR11: The system shall secure APIs and protect user data.

FR11.1: Implement authentication mechanisms (e.g., API keys, tokens).

FR11.2: Validate and sanitize all inputs to prevent security vulnerabilities.

Non-Functional Requirements

3.8. Performance

NFR1: The system should process and analyze books efficiently.

NFR1.1: Target processing time for average books (up to 500 pages) under 5 minutes.

NFR1.2: Optimize resource usage during text extraction and analysis.

NFR2: Conversation responses should have low latency.

NFR2.1: Aim for response times under 3 seconds for generating character replies.

3.9. Scalability

NFR3: The system should support concurrent users and scale horizontally.

NFR3.1: Design the architecture to allow for load balancing and clustering.

NFR3.2: Use scalable storage solutions for session and analysis data.

3.10. Reliability and Availability

NFR4: The system should be robust and highly available.

NFR4.1: Implement redundancy for critical components.

NFR4.2: Ensure that system failures do not result in data loss.

3.11. Usability

NFR5: API interfaces should be intuitive and developer-friendly.

NFR5.1: Provide clear API documentation and usage examples.

NFR5.2: Maintain consistent API design patterns.

3.12. Maintainability

NFR6: The codebase should be modular and easy to maintain.

NFR6.1: Use clean coding practices and design patterns.

NFR6.2: Include comprehensive documentation and comments.

3.13. Security

NFR7: Protect user data and ensure compliance with security standards.

NFR7.1: Encrypt sensitive data in transit and at rest.

NFR7.2: Regularly update dependencies to patch vulnerabilities.

---

4. System Architecture

High-Level Architecture

The application is composed of several key components:

API Layer: Exposes endpoints for clients to interact with the application.

Business Logic Layer: Handles core functionalities such as file processing, text extraction, character analysis, and interaction simulation.

AI Integration Module: Interfaces with external AI services for language processing tasks.

Data Storage: Manages storage of analysis results, session data, and other persistent information.

Authentication and Security Module: Manages user authentication, authorization, and data protection.

Component Descriptions

4.1. API Layer

Function: Receives HTTP requests from clients and routes them to the appropriate handlers.

Responsibilities:

Validate incoming requests.

Enforce authentication and authorization policies.

Format and send responses back to clients.

4.2. Business Logic Layer

Function: Contains the core application logic.

Responsibilities:

File processing and text extraction.

Coordinating character analysis passes.

Managing interaction simulation.

Session management.

4.3. AI Integration Module

Function: Handles communication with external AI services.

Responsibilities:

Prepare prompts and requests for AI models.

Process and parse responses.

Implement fallback mechanisms and error handling for AI services.

4.4. Data Storage

Function: Stores persistent data required by the application.

Responsibilities:

Maintain analysis results and session data.

Ensure data consistency and integrity.

Provide efficient data retrieval mechanisms.

4.5. Authentication and Security Module

Function: Secures the application and protects data.

Responsibilities:

Manage user authentication tokens.

Enforce access controls.

Monitor for security threats and vulnerabilities.

---

5. Detailed Design

Core Application Logic

The core application logic is designed to be modular and extensible, encapsulating key functionalities into dedicated classes and modules.

Data Models

5.1. Character Models

Defined using data serialization and validation libraries (e.g., pydantic, dataclasses).

BasicCharacter

CharacterDescription

CharacterDepth

5.2. Session Data

Structured to maintain user-specific data:

SessionID: Unique identifier for each session.

Uploaded Files: References to the user's uploaded book files.

Analysis Results: Stored character analysis data linked to the session.

Conversation History: List of messages exchanged between the user and characters.

Algorithms and Processes

5.3. Multi-Pass Character Analysis

First Pass:

Input: Extracted text from the book.

Process:

Generate a prompt for the AI model to identify significant characters.

Parse the AI model's response into BasicCharacter objects.

Output: List of BasicCharacter instances.

Second Pass:

Input: List of characters from the first pass.

Process:

Generate detailed prompts for each character.

Gather detailed descriptions using the AI model.

Parse responses into CharacterDescription objects.

Output: List of CharacterDescription instances.

Third Pass:

Input: List of characters from the first pass.

Process:

Generate prompts focusing on character depth and development.

Parse the responses into CharacterDepth objects.

Output: List of CharacterDepth instances.

5.4. Interaction Simulation

Preparation:

Combine character details to form a comprehensive persona prompt.

Include guidelines and constraints to ensure authentic responses.

Conversation Handling:

Input: User message and conversation history.

Process:

Build a message payload including the persona prompt and conversation history.

Invoke the AI model to generate the character's response.

Output: Character's reply.

External Interfaces

5.5. AI Model APIs

Google Generative AI:

Used for initial analysis passes and complex language tasks.

Requires secure API key management.

OpenAI GPT Models:

Used as a fallback or for specific tasks like formatting and parsing.

API interactions handled via official SDKs or RESTful calls.

---

6. API Design

Endpoints

6.1. File Upload

POST /upload

Description: Uploads a book file for processing.

Parameters: File (multipart/form-data)

Response: { "session\_id": "...", "status": "processing" }

6.2. Analysis Status

GET /analysis/status

Parameters: session\_id

Response: { "status": "completed", "progress": 100 }

6.3. Get Character List

GET /characters

Parameters: session\_id

Response: { "characters": [ { "name": "...", "role": "..." }, ... ] }

6.4. Start Conversation

POST /conversations/start

Parameters: session\_id, character\_name

Response: { "conversation\_id": "..." }

6.5. Send Message

POST /conversations/{conversation\_id}/messages

Parameters: message

Response: { "reply": "...", "conversation\_id": "..." }

Request and Response Formats

Format: JSON for both requests (where applicable) and responses.

Error Responses: Standardized error messages with HTTP status codes and error details.

Authentication and Security

API Keys/Tokens: Use API keys or tokens passed in headers for authentication.

Encryption: Use HTTPS for all communication.

---

7. User Interface Considerations

Front-End Integration

While the core application focuses on backend functionalities, it's designed to be integrated with various front-end interfaces:

Web Applications: Providing interactive web pages for users.

Mobile Applications: Allowing users to interact via mobile devices.

Third-Party Integrations: Enabling other applications to leverage the API.

User Interface Requirements

UIR1: Front-end should handle file selection and upload via the provided API.

UIR2: Display analysis progress and allow users to select characters once analysis is complete.

UIR3: Provide a chat interface that interacts with the conversation APIs.

UIR4: Handle error messages and status updates gracefully.

---

8. Error Handling and Logging

EH1: Implement comprehensive exception handling in all modules.

EH2: Use structured logging frameworks to record errors and operational logs.

EH3: Ensure sensitive information is not logged or exposed.

EH4: Provide clear and consistent error responses to API clients.

---

9. Security Requirements

SEC1: Protect against common web vulnerabilities (e.g., injection attacks, cross-site scripting).

SEC2: Implement rate limiting and throttling to prevent abuse.

SEC3: Regularly audit code and dependencies for security issues.

SEC4: Ensure compliance with data protection regulations.

---

10. Performance Requirements

PERF1: Optimize algorithms for text processing and AI model interactions.

PERF2: Utilize asynchronous processing where appropriate.

PERF3: Implement caching strategies to reduce redundant computations.

PERF4: Monitor system performance and resource utilization.

---

11. Deployment Considerations

DEP1: Containerize the application using Docker for consistent deployments.

DEP2: Use orchestration tools like Kubernetes for scaling and management.

DEP3: Automate deployments using CI/CD pipelines.

DEP4: Configure environment variables and secrets securely in deployment environments.

---

12. Testing and Quality Assurance

TEST1: Write unit tests for all core functionalities with high code coverage.

TEST2: Perform integration testing for API endpoints.

TEST3: Conduct load testing to assess performance under stress.

TEST4: Implement automated testing as part of the CI/CD pipeline.

TEST5: Validate AI model interactions and outputs for correctness.

---

13. Maintenance

MAINT1: Set up monitoring and alerting systems for uptime and error tracking.

MAINT2: Document APIs and codebase thoroughly for ease of understanding.

MAINT3: Establish processes for updating dependencies and addressing deprecations.

MAINT4: Provide support channels for clients to report issues.