**Project Planning: AI-Powered Character Generator**

**1. Project Overview**

**1.1 Project Scope**

* **Primary Goal**: Develop an AI-powered character generator that creates both textual stories and visual portraits
* **Core Features**:
  + Character story generation using LLaMA2 model
  + AI-generated character portraits using Stable Diffusion
  + Interactive web interface built with Streamlit
* **Target Users**: Writers, game developers, content creators, and storytelling enthusiasts

**1.2 Success Criteria**

* Generate coherent character backstories (max 150 words)
* Produce high-quality character portraits matching descriptions
* Provide intuitive user interface for character customization
* Support multiple genres (Fantasy, Sci-Fi, Historical, Modern, etc.)

**2. Technical Architecture**

**2.1 System Components**

* **Frontend**: Streamlit web application (Final\_Deliverables/UI/main.py)
* **Story Generation**: LangChain + Ollama LLaMA2 integration (App/story\_prompt.py)
* **Image Generation**: Stable Diffusion v1.5 pipeline (App/image\_generator.py)
* **Orchestration**: Main generator module (App/generator.py)

**2.2 Technology Stack**

* **Programming Language**: Python 3.11+
* **Web Framework**: Streamlit
* **AI Models**:
  + Text: LLaMA2 via Ollama
  + Image: Stable Diffusion v1.5
* **Key Libraries**: LangChain, Transformers, Diffusers, PyTorch, Pillow

**3. Development Phases**

**3.1 Phase 1: Foundation (Completed)**

* ✅ Basic Streamlit UI setup
* ✅ Prompt template creation
* ✅ Initial character description generation
* **Location**: Project-Design\_01/

**3.2 Phase 2: AI Integration (Completed)**

* ✅ LLaMA2 model integration via Ollama
* ✅ Stable Diffusion pipeline implementation
* ✅ Story and image generation coordination
* **Location**: Project-Design\_02/

**3.3 Phase 3: Final Implementation (Completed)**

* ✅ Enhanced UI with error handling
* ✅ Expanded genre selection
* ✅ Image display and management
* ✅ Output directory structure
* **Location**: Final\_Deliverables/

**4. Resource Requirements**

**4.1 Hardware Requirements**

* **GPU**: CUDA-compatible GPU (recommended for Stable Diffusion)
* **RAM**: Minimum 16GB (32GB recommended)
* **Storage**: 10GB+ for model weights and outputs
* **CPU**: Modern multi-core processor

**4.2 Software Dependencies**

As specified in requirements.txt:

* streamlit
* langchain
* torch
* transformers
* diffusers
* accelerate
* sentencepiece
* Pillow

**5. Data Flow Architecture**

**5.1 User Input Processing**

1. User enters character details (name, genre, traits)
2. Input validation in Streamlit form
3. Data passed to [generate\_character](vscode-file://vscode-app/c:/Users/mohak/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html" \o ") function

**5.2 Story Generation Pipeline**

1. [generate\_story](vscode-file://vscode-app/c:/Users/mohak/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html) receives user inputs
2. LangChain prompt template formats the request
3. Ollama LLaMA2 model generates structured response
4. Response parsed for story and appearance sections

**5.3 Image Generation Pipeline**

1. Character appearance description extracted
2. [generate\_image](vscode-file://vscode-app/c:/Users/mohak/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html) processes description
3. Stable Diffusion pipeline creates portrait
4. Image saved to Outputs/ directory

**6. Quality Assurance Plan**

**6.1 Testing Strategy**

* **Unit Testing**: Individual component validation
* **Integration Testing**: End-to-end workflow verification
* **User Acceptance Testing**: Interface usability evaluation
* **Performance Testing**: Model response time measurement

**6.2 Error Handling**

* Input validation for required fields
* Exception handling for model failures
* Graceful degradation for resource constraints
* User-friendly error messages in UI

**7. Deployment Strategy**

**7.1 Local Development**

* Python virtual environment setup
* Dependency installation via pip
* Local Streamlit server execution
* Model downloading and caching

**7.2 Production Considerations**

* GPU resource allocation
* Model weight storage optimization
* Session state management
* Output file cleanup policies

**8. Maintenance and Updates**

**8.1 Model Updates**

* LLaMA model version upgrades
* Stable Diffusion model improvements
* Prompt template refinements

**8.2 Feature Enhancements**

* Additional genre categories
* Character trait suggestions
* Export functionality for generated content
* User preference saving

**9. Risk Assessment**

**9.1 Technical Risks**

* **Model Availability**: Dependency on external model services
* **Resource Constraints**: High GPU/memory requirements
* **Generation Quality**: Inconsistent AI output quality

**9.2 Mitigation Strategies**

* Local model deployment for reliability
* Fallback options for resource limitations
* Quality validation and regeneration options

**10. Timeline Summary**

* **Phase 1**: Foundation setup and basic UI
* **Phase 2**: AI model integration and coordination
* **Phase 3**: Final implementation and polish
* **Current Status**: Production-ready application with full feature set

This project successfully demonstrates the integration of multiple AI technologies to create a comprehensive character generation tool suitable for creative professionals and enthusiasts.