# Fantasy RPG Text Adventure Game - Project Overview

## Project Summary

This project is a text-based adventure game that combines traditional RPG mechanics with modern technologies. It uses MongoDB for data persistence and Google Gemini AI for generating dynamic, contextual responses. The game allows players to create characters, explore a fantasy world, interact with NPCs, complete quests, and engage in combat.

## Key Technologies

- **Python**: Core programming language
- **MongoDB**: Database for storing game state, player data, and world information
- **Google Gemini AI**: Generative AI model for creating dynamic game content and responses

## Architecture

### Database Layer (MongoDB)

The game uses MongoDB to store:

- Player data (stats, inventory, quest progress, choices)
- World information (locations, connections)
- Items (weapons, armor, consumables)
- Quests (objectives, rewards)
- NPCs and enemies

### AI Integration (Google Gemini)

Google Gemini is used to generate:

- Rich location descriptions
- NPC dialogue
- Combat narratives
- Quest interactions
- Responses to player actions

### Game Engine

The core game engine handles:

- Character creation and progression
- World navigation
- Inventory management
- Quest tracking
- Combat mechanics
- Command processing

# Features

## Character System

- Multiple character classes (Warrior, Mage, Rogue)
- Character stats and leveling
- Inventory management
- Equipment system

## World Exploration

- Multiple interconnected locations
- Location descriptions enhanced by AI
- Random encounters
- Discovery mechanics

## Quest System

- Multiple quests with objectives
- Quest rewards (XP, gold, items)
- Quest progress tracking
- NPC interactions

## Combat System

- Turn-based combat mechanics
- Various enemy types
- Combat rewards
- AI-enhanced combat descriptions

## Command Interface

- Natural language command processing
- Contextual responses
- Help system

# Project Structure

```
fantasy-rpg/
├── .env                  # Environment variables (MongoDB URI and Gemini API key)
├── main.py               # Main entry point for the game
├── init_mongodb.py       # Script to initialize MongoDB with game data
├── README.md             # Project documentation
├── overview.md           # This project overview
├── game/                 # Main game package
│   ├── __init__.py
│   ├── database.py       # MongoDB connection and operations
│   ├── ai_generator.py   # Google Gemini integration
│   ├── game_engine.py    # Core game mechanics
│   └── data/             # Game data
```

```
│       ├── __init__.py
│       ├── enemies.py    # Enemy definitions
│       └── npcs.py       # NPC definitions
```

## MongoDB Collections

- **players**: Player character data, inventory, quest progress
- **items**: Game items with properties and effects
- **quests**: Available quests with objectives and rewards
- **world**: World locations and connections
- **enemies**: Enemy types and properties
- **npcs**: Non-player characters with dialogue and quests

## AI Integration Details

The game uses Google Gemini 2.0 Flash model to generate:

- Enhanced location descriptions based on basic data
- Combat narratives that describe the action
- NPC dialogue for quest interactions
- Item discovery descriptions
- Responses to player actions not explicitly handled by the game engine

## Current Status

The project has a functional core with:

- Basic game mechanics implemented
- MongoDB integration for data persistence
- Google Gemini integration for text generation
- Initial game content (locations, items, quests, NPCs)

## Future Enhancements

Potential areas for expansion:

- More complex combat system
- Expanded world with additional locations
- More character classes and abilities
- Crafting system
- Weather and time systems
- Multiplayer capabilities
- Web-based interface

## Getting Started

1. Set up MongoDB connection in `.env`
2. Configure Google Gemini API key in `.env`
3. Run `init_mongodb.py` to initialize the database

4. Start the game with `python main.py`
5. Create a character and begin your adventure!

## Technical Challenges

- Balancing between pre-defined game logic and AI-generated content
- Ensuring database operations are efficient
- Maintaining game state consistency
- Handling natural language commands effectively
- Managing AI response context and consistency