

Game Development Document System - Comprehensive Specification

Project Documentation: Part 1 of 2

1. System Overview

The Game Development Document System (GDDS) is a Chrome extension that enhances Claude.ai with specialized functionality for creating a comprehensive suite of game development documents. The system leverages Claude's conversational AI capabilities to guide users through an adaptive questioning process that builds each document iteratively, with each document informing the creation of subsequent documents.

2. Core Document Types

The system supports the following document types, typically created in this sequence:

1. **Game Vision** - High-level concept and creative direction
2. **Core Game Concept** - Fundamental gameplay and experience definition
3. **Target Audience** - Detailed analysis of intended players
4. **Core Pillars and Values** - Guiding principles and design values
5. **Why Would They Play It?** - Player motivations and retention hooks
6. **What Should They Feel?** - Emotional experience and player impact
7. **Unique Selling Points** - Distinctive features and market differentiation
8. **Game Loop** - Core gameplay cycle and mechanics
9. **Player Journey** - Player progression and experience arc
10. **Story Overview** - Narrative foundation and thematic elements
11. **Presentation** - How the game will be shown/marketed
12. **Key Questions** - Critical design and development considerations
13. **Core Design Details** - Specific systems and design specifications
14. **Strategic Direction** - Business and development strategy

Each document has a specific purpose and structure but is created through a dynamic conversation rather than a fixed template.

3. User Experience Flow

3.1 Project Management




1. Chrome Extension Installation

- User installs extension from Chrome Web Store
- Extension adds a collapsible sidebar to Claude.ai interface

2. Project List View

- Sidebar shows folders for each game project
- "Start New Project" button at the top
- Each project folder expands to show document list

3. Document List View

- Completed documents shown with full name and icons for actions:
 -  (view document)
 -  (refine document)
 -  (export document)
- Pending documents shown in gray with a "+" button to initiate creation
- Documents in-progress shown with a progress indicator

3.2 Document Creation Flow

1. Initiation

- User clicks "+" on a document from the list
- System prepares Claude with specialized context for that document type
- If this isn't the first document, system provides Claude with content from prior documents

2. Guided Conversation

- Claude begins with an open-ended prompt about the document topic
- Based on user responses, Claude asks targeted follow-up questions
- Questions adapt to previous answers, exploring different aspects
- The conversation follows Claude's natural questioning style, not preset questions

3. Document Drafting

- Once sufficient information is gathered, Claude drafts the document
- User reviews the draft and can request changes
- Claude refines based on feedback
- Process continues until user is satisfied

4. Document Finalization

- User approves the final version
- System stores the document in the project folder
- Document becomes available for reference in creating subsequent documents

3.3 Document Refinement

1. Access

- User clicks the "refine" icon on a completed document
- System loads document content and prepares Claude with context

2. Refinement Process

- Claude presents the current document with options for refinement
- User specifies desired changes or improvements
- Claude updates the document based on feedback
- Changes are saved to the document

3.4 Document Export

1. Export Options

- PDF Export (for sharing and printing)
- Google Drive Export (for collaborative editing)
- Markdown Export (for technical documentation)
- Plain Text Export (for maximum compatibility)

2. Export Process

- User clicks export icon on a document
- System presents export format options
- User selects desired format
- System generates file and initiates download
- Option to save to connected cloud storage

4. Technical Architecture

4.1 Chrome Extension Components

1. Background Script

- Manages extension lifecycle

- Handles authentication with storage services
- Coordinates data synchronization

2. **Content Script**

- Injects UI elements into Claude.ai
- Manages interaction between Claude and extension
- Captures Claude responses
- Injects prompts and context into Claude conversations

3. **Popup Interface**

- Settings and configuration
- Account management
- Quick actions

4. **Sidebar Component**

- Project and document navigation
- Document status monitoring
- Action buttons for documents

4.2 Data Architecture

1. **Local Storage**

- Project metadata
- Document metadata
- User preferences
- Authentication tokens

2. **Chrome Extension Storage**

- Document content (limited by 5MB Chrome extension storage)
- Conversation history (limited to recent/current)

3. **Cloud Storage Integration**

- Google Drive for document storage
- Option for Dropbox integration
- Authentication and permissions management

4.3 Claude Integration

1. **Context Injection Methods**

- Document-specific system prompts
- Previous document content injection
- Conversation state management

2. Response Capturing

- Identify and extract document drafts from Claude responses
- Track conversation flow for document building

4.4 Document Processing

1. Format Conversion

- Markdown as internal format
- Conversion to PDF, Google Docs formats
- Style preservation during export

2. Version Control

- Document revision history
- Changes tracking between versions
- Restore previous versions

5. AI Prompting Framework

5.1 Document-Specific Context

For each document type, Claude needs specific context about:

1. **Document Purpose** - What this document aims to accomplish
2. **Typical Content** - Key elements usually found in this document
3. **Relationship to Other Documents** - How it builds on or informs other docs
4. **Target Audience** - Who will read/use this document

5.2 Question Generation Guidance

Claude should be guided to:

1. Begin with open-ended questions about the document topic
2. Follow up with specific questions based on user responses
3. Explore multiple aspects of the topic (creative, technical, business)
4. Reference content from previous documents when relevant

5. Identify inconsistencies or gaps in the game concept
6. Help users think more deeply about their game vision

5.3 Document Context Examples

Here are examples of the context provided to Claude for each document type:

Game Vision Document Context:

You are helping create a Game Vision document. This is the foundational document that captures the creative essence and high-level concept of the game. Ask open-ended questions to understand what makes this game special, its core experience, and the creator's inspiration. Focus on emotional impact, unique selling points, and creative direction. This document will inform all subsequent design decisions.

Core Game Concept Context:

You are helping create a Core Game Concept document. Building on the Game Vision, this document defines the fundamental gameplay experience. Ask questions about core mechanics, player actions, world/setting details, and how these elements deliver on the vision. Focus on what players actually do in the game and how it feels to play.

Target Audience Context:

You are helping create a Target Audience document. Using information from the Vision and Core Concept, explore who this game is designed for. Ask questions about demographic factors, player preferences, gaming habits, and why this audience would be attracted to this game. Focus on creating specific player personas and understanding their motivations.

(Additional document contexts would follow similar patterns)

6. Implementation Plan

6.1 Phase 1: Core Extension

1. Basic Chrome Extension Setup

- Manifest configuration
- Content script injection into Claude.ai
- Basic sidebar UI

2. Project Management

- Create/list/delete projects

- Local storage implementation
- Basic document status tracking

3. **Simple Document Creation**

- Manual context injection
- Conversation capture
- Basic document storage

6.2 Phase 2: Enhanced Document Creation

1. **Improved Context Injection**

- Document-specific prompting
- Previous document reference system
- Conversation flow management

2. **Document Processing**

- Markdown formatting
- Basic export functionality
- Document versioning

3. **UI Refinements**

- Improved sidebar UI
- Document progress indicators
- Error handling and recovery

6.3 Phase 3: Cloud Integration & Advanced Features

1. **Cloud Storage**

- Google Drive API integration
- Authentication and security
- Sync mechanism

2. **Advanced Export**

- Multiple format support
- Styled PDF generation
- Batch export options

3. **Collaboration Features**

- Shared projects (if demand exists)

- Comment/feedback system
- Team workflow support

6.4 Phase 4: Refinement & Optimization

1. Performance Optimization

- Reducing storage requirements
- Improving load times
- Efficient context management

2. User Experience Improvements

- Onboarding process
- Tooltips and guidance
- Visual design refinement

3. Advanced Features

- Custom document types
- Template management
- Analytics for document creation

7. Integration Specifics

7.1 Claude.ai DOM Integration

The extension needs to interact with Claude.ai's interface:

1. Sidebar Injection

- Target: Right side of Claude interface
- Method: DOM insertion via content script
- Style: Match Claude's dark mode aesthetic

2. Prompt Injection

- Target: Claude's input field
- Method: Programmatic input insertion
- Timing: On document creation/refinement initiation

3. Response Capture

- Target: Claude's response containers
- Method: DOM observation and content extraction

- Processing: Parse markdown/text content

7.2 Google Drive Integration

For document storage and sync:

1. Authentication

- OAuth 2.0 flow
- Permission scopes: Files (read/write)
- Token storage and refresh

2. File Operations

- Create folders for projects
- Store documents as individual files
- Update existing files during refinement

3. Sync Management

- Bidirectional sync capability
- Conflict resolution
- Offline support

8. Security Considerations

1. User Data Protection

- Local encryption of sensitive data
- Secure authentication token handling
- Minimal data collection policy

2. API Security

- Secure API key management
- Rate limiting to prevent abuse
- Proper error handling

3. Privacy Measures

- Clear data usage policies
- Option to delete all data
- Transparency about storage locations

9. Technical Requirements

9.1 Development Tools

1. Frontend

- JavaScript/TypeScript
- React for UI components
- Chrome Extension APIs

2. Backend (if needed)

- Node.js for any server components
- Firebase/similar for lightweight backend

3. Storage

- Chrome storage API
- Google Drive API
- IndexedDB for larger local storage

9.2 APIs and Services

1. Required APIs

- Chrome Extensions API
- Google Drive API
- Claude API (if direct integration is preferred over web injection)

2. Optional Services

- Firebase (authentication, backend)
- PDF generation service
- Analytics service

10. Document-Specific AI Guidance

For each document type, Claude should follow these specific approaches:

10.1 Game Vision Document

Purpose: Capture the creative essence and high-level concept of the game.

Question Strategy:

- Begin with: "Tell me about your game vision at a high level."
- Explore inspirations and influences
- Dig into emotional goals and player feelings

- Investigate unique aspects and innovation
- Understand creative direction and aesthetic goals

Document Structure Guidance:

- High concept statement (1-2 sentences)
- Creative vision and inspiration
- Core experience and emotions
- Unique selling points
- Target feelings and player takeaways

10.2 Core Game Concept

Purpose: Define the fundamental gameplay experience.

Question Strategy:

- Begin with: "Describe the basic gameplay of your game."
- Explore core actions and mechanics
- Investigate world/setting details
- Understand progression and goals
- Dig into how mechanics support the vision

Document Structure Guidance:

- Game type/genre
- Core gameplay loop
- Primary mechanics
- Setting/world overview
- Player objectives and goals

10.3 Target Audience

Purpose: Identify and understand the intended players.

Question Strategy:

- Begin with: "Who do you see playing this game?"
- Explore demographic factors
- Investigate player preferences and habits

- Understand market positioning
- Dig into audience needs and desires

Document Structure Guidance:

- Primary audience description
- Player personas (2-3)
- Audience gaming preferences
- Market positioning
- Audience needs fulfilled

(Continue with similar detailed guidance for each document type)

11. Implementation Examples

11.1 Example Extension Manifest

json

```
{
  "manifest_version": 3,
  "name": "Game Development Document System",
  "version": "1.0",
  "description": "Create comprehensive game development documents with Claude AI",
  "permissions": [
    "storage",
    "identity",
    "scripting"
  ],
  "host_permissions": [
    "https://claude.ai/*"
  ],
  "action": {
    "default_popup": "popup.html",
    "default_icon": {
      "16": "images/icon16.png",
      "32": "images/icon32.png",
      "48": "images/icon48.png",
      "128": "images/icon128.png"
    }
  },
  "content_scripts": [
    {
      "matches": ["https://claude.ai/*"],
      "js": ["content.js"],
      "css": ["sidebar.css"]
    }
  ],
  "background": {
    "service_worker": "background.js"
  }
}
```

11.2 Example DOM Injection Code

javascript

// This is a simplified example of how to inject the sidebar into Claude.ai

```
function injectSidebar() {
  const sidebarHTML = `
    <div id="gdds-sidebar" class="gdds-sidebar">
      <div class="gdds-sidebar-header">
        <h3>Game Documents</h3>
        <button id="gdds-new-project">New Project</button>
      </div>
      <div class="gdds-project-list">
        <!-- Projects will be inserted here -->
      </div>
    </div>
  `;

  // Find the right element to append to in Claude.ai
  const claudeContainer = document.querySelector('.claude-container');
  if (claudeContainer) {
    const sidebarContainer = document.createElement('div');
    sidebarContainer.innerHTML = sidebarHTML;
    claudeContainer.appendChild(sidebarContainer);
    setupSidebarListeners();
  }
}

// Call this when the page loads
document.addEventListener('DOMContentLoaded', injectSidebar);
```

11.3 Example Context Injection

javascript

```

// This is a simplified example of how to inject context into Claude
function injectDocumentContext(documentType, previousDocuments) {
  // Get the appropriate context for this document type
  const context = getDocumentContext(documentType, previousDocuments);

  // Find Claude's input field
  const inputField = document.querySelector('.claude-input-field');
  if (inputField) {
    // Set the value
    inputField.value = context;

    // Dispatch an input event to make sure Claude recognizes it
    inputField.dispatchEvent(new Event('input', { bubbles: true }));

    // Submit the form
    const submitButton = document.querySelector('.claude-submit-button');
    if (submitButton) {
      submitButton.click();
    }
  }
}

function getDocumentContext(documentType, previousDocuments) {
  // Base context for all document types
  let context = "You are helping create a game development document. ";

  // Add document-specific context
  switch(documentType) {
    case "GameVision":
      context += "This is a Game Vision document that captures the creative essence and high-level goals of a game. Ask open-ended questions to understand what makes this game special, its core mechanics, and its unique selling points."
      break;
    case "CoreGameConcept":
      context += "This is a Core Game Concept document that defines the fundamental gameplay elements and mechanics of a game. Ask questions about core mechanics, player actions, and how these elements interact with each other."
      context += previousDocuments.GameVision || "";
      break;
    // Add cases for other document types
  }

  // Add general guidance
  context += "Begin with open-ended questions. Follow up with specific questions based on responses. Help the user think deeply about their game by exploring different aspects and possibilities."
}

```



```
    return context;  
}
```

12. User Onboarding

12.1 First-Time Experience

1. Welcome Screen

- Brief explanation of the system
- Value proposition
- Setup guidance

2. Google Drive Connection

- Optional but recommended
- Clear permission explanation
- Quick setup flow

3. First Project Creation

- Guided creation of first project
- Explanation of document flow
- Tips for effective document creation

12.2 Ongoing Guidance

1. Tooltips and Hints

- Contextual guidance during use
- Shortcuts and efficiency tips
- Best practices

2. Help Resources

- FAQ section in extension
- Quick reference guides
- Example documents

13. Test Cases and Quality Assurance

13.1 Technical Testing

1. Extension Installation

- Clean installation
- Updates
- Multiple browser compatibility

2. **Core Functionality**

- Project management
- Document creation flow
- Storage and retrieval
- Export functionality

3. **Edge Cases**

- Large documents
- Network interruptions
- Claude API changes
- Storage limitations

13.2 User Experience Testing

1. **Usability Goals**

- Document creation under 15 minutes
- Intuitive navigation without guidance
- Clear error messaging
- Responsive UI performance

2. **User Testing Scenarios**

- First-time user journey
- Complete project creation
- Document refinement process
- Export and sharing workflow

14. Maintenance and Updates

14.1 Monitoring

1. **Claude.ai Changes**

- DOM structure changes
- API behavior changes
- Feature additions/removals

2. **Browser Compatibility**

- Chrome updates
- Manifest version changes
- Permission model changes

14.2 Update Process

1. **Version Management**

- Semantic versioning
- Change logs
- Deprecation notices

2. **User Communication**

- Update notifications
- Feature announcements
- Maintenance warnings

15. Future Expansion Possibilities

15.1 Additional Document Types

1. **Technical Documents**

- Technical Design Document (TDD)
- Art Style Guide
- Audio Design Document

2. **Production Documents**

- Production Schedule
- Risk Assessment
- Resource Allocation

15.2 Advanced Features

1. **AI-Enhanced Analysis**

- Consistency checking across documents
- Market viability assessment
- Design pitfall detection

2. **Collaborative Features**

- Real-time collaboration
- Comment and feedback system
- Role-based permissions

3. **Integration Ecosystem**

- Integration with project management tools
- Game engine documentation export
- Publishing platform submission preparation

16. Closing Notes

This specification provides a comprehensive blueprint for implementing the Game Development Document System as a Chrome extension that enhances Claude.ai. The system leverages Claude's natural conversational abilities while adding structured document management capabilities.

The implementation should prioritize maintaining the quality of Claude's dynamic questioning process while adding value through organization, persistence, and document management features.

By following this specification, developers should be able to create a system that significantly streamlines the game development documentation process while maintaining the creative benefits of AI-assisted document creation.