[PRD] RIKU Chatbot

TL;DR: RIKU is an AI chatbot that gives gamers convenient access to helpful information, statistics, maps, and guides about their favorite video games in Q&A format, allowing gamers to find answers to their in-game queries efficiently.

Executive Summary

RIKU: An AI-Powered Productivity Tool

Problem Statement

Gamers often struggle to find specific information in the vast sea of internet resources while playing. Switching between the game and various online resources such as blogs, wikis, and guides disrupts the gaming experience and is time-consuming. This issue is particularly pronounced in complex game genres like RPGs and MMOs, where players need detailed information to optimize their gameplay strategies, navigation, builds, class selection, boss preparation, and more.

Solution Overview

RIKU addresses this problem as a centralized AI-powered chatbot that pulls information from wikis and other reliable sources. It offers precise and quick answers to user queries (e.g., “where can I find item X?, what’s the hidden ability for class Y?”), thereby enhancing the gaming experience with reliable information at gamers' fingertips. By reducing the need to pause and search through fragmented sources, RIKU supports a smoother and more immersive gaming experience.

Overview

What problem are we solving?

Core Problem: Gamers often face significant interruptions in their gaming experience due to the need to search for specific information online. Whether it’s looking up the location

of a crafting material or comparing weapon stats, players are forced to pause their game, disrupting their immersion and flow. The core problem is the inefficiency and disruption caused by having to search through various fragmented sources, especially in genres like RPGs and MMOs where detailed information is crucial for optimal gameplay

Solutions today: Current solutions involve fragmented, inconsistently formatted information spread across numerous sites and pages. This leads to a time-consuming and disruptive process that breaks the immersive gaming experience.

This matters to our users because:

1. **Emotional:** It's frustrating to have to pause their focus and engagement with the game to hunt for information, especially if it requires a repetitive, trial-and-error ridden process of switching between the game and various pages.
2. **Functional:** Searching for information can be extremely time-consuming, especially when players need to sift through multiple sources or pages to collect fragmented information.

Customer, research, and market signals indicating the problem:

1. **Time and Effort Investment:** Gamers dedicate many hours to understanding game mechanics and optimizing strategies, especially in competitive gaming and evolving metas.
2. **Growth in RPGs and MMOs:** RPGs and MMOs are increasingly popular and data-intensive genres in which players require detailed information on quests, items, maps, and character builds.
3. **Static Existing Options:** Current gaming wikis and guides lack real-time, interactive capabilities. They are mostly static and do not provide personalized responses, leading to fragmented and inefficient information retrieval.
4. **Community Knowledge:** Many gamers rely on platforms like Reddit or Discord to share tips and strategies, but such information is inherently fragmented and inaccessible to many.

How might we tackle this problem?

> **Solution overview:** Our solution is a web app featuring AI capabilities that allow gamers to ask questions about in-game information for their favorite video games. The AI analyzes data pulled from wikis, guides, and other resources, helping gamers find accurate in-game information quickly and efficiently. This tool addresses the need for a streamlined, interactive, and real-time solution for accessing game data.

List of Features we brainstormed:

- **P0s**
  - **Accept user questions:** Allow users to input questions about specific game details (e.g., what a certain debuff does in Xenoblade).
  - **Collect game data for key titles:** Collect and store information from pre-selected resources (e.g., Fandom wikis) for various games.
  - **RAG-capable AI:** Integrate LLMs so that RIKU can answer user queries dynamically
- **P1s**
  - **Voice capabilities:** Allow users to ask RIKU questions through voice conversation, further reducing disruption of gameplay by not needing to type or look away from the screen.
  - **Scrape entire game libraries for key platforms:** Increase RIKU's library of games by scraping a greater breadth and depth of video game information repositories
  - **Fine-tuned models:** Fine-tune the AI models on video game data to improve performance and accuracy
  - **Recommendations:** Provide strategies and build recommendations (e.g., class and equipment combinations) for the games, based on game performance data or expert inputs.
  - **Multilingual support:** Provide support for multiple languages to cater to a global user base.
- **P2s**
  - **Spoilers:** Ensure spoilers and sensitive story elements are not unintentionally revealed to the player.
  - **PC / Steam integration:** Provide game-specific data directly from these platforms via a desktop app or in-game overlay.

- **Context-aware responses**: Enhance AI to provide context-aware responses that understand the current game state or specific situations within the game.

What key benefits will we provide?

1. **Efficiency:** Streamlines the process of finding specific game information, reducing the time gamers spend searching through multiple sources.
2. **Comprehensive and centralized:** Aggregates and synthesizes information from multiple sources or pages, providing a centralized platform for game queries.

Approach

Who are we building for?

1. **RPG and MMO Gamers:** Players of complex, data-intensive games like Xenoblade Chronicles, Final Fantasy, and other popular RPGs and MMOs who require detailed information to optimize their gameplay.
2. **Competitive Gamers:** Players involved in competitive gaming who need quick and accurate information to maintain a competitive edge during gameplay.

What does success look like?

1. **Adoption**
    a. **Metric:** Achieve 10,000 active users within the first six months of launch.
    b. **Feeling:** Users find RIKU valuable, additive, and undisruptive for their gaming sessions, relying on it to enhance their gameplay.
2. **Retention:**
    a. **Metric:** Maintain a monthly user retention rate of 75%.
    b. **Feeling:** Users consistently use RIKU because it provides immediate value without disrupting their gaming flow.
3. **Accuracy:**
    a. **Metric:** Metric: Achieve a response accuracy rate of at least 90% as rated by user feedback.
    b. **Feeling:** Users trust the accuracy and quality of the information provided by RIKU.

What are our non-goals?

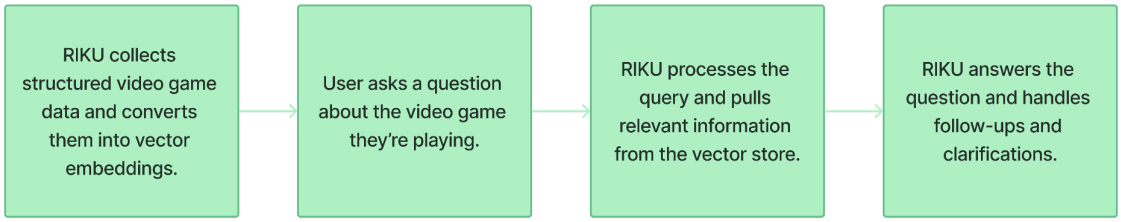
1. **Creating a comprehensive gaming wiki:** We'll focus on giving quick, reliable answers to specific user queries, not a comprehensive database of game data.
2. **Full game integration:** We are not aiming to integrate fully with every game's internal mechanics but rather to provide an external, supportive tool that complements the gaming experience.

Paint the story?

- Alex is deeply engaged in playing Xenoblade Chronicles and often needs specific information about quests, items, and character builds. With RIKU, Alex can input questions and get precise answers without pausing the game, enhancing the overall gameplay experience.
- Jamie is in the middle of an intense MMO raid and needs quick information about a boss's weaknesses and strategies. Using RIKU, Jamie can get real-time, accurate responses, ensuring the raid's success without disrupting the flow of the game.

Solution

What does a solution look like?



1. **RIKU scrapes and collects game data:** Scrape, parse, and organize information for select titles in a way that's accessible for the AI.
2. **User asks a question about a supported title:** The user inputs a specific question about game details (e.g., "when does Metang evolve in Pokemon Ruby?").
3. **RIKU generates an answer:** The AI model processes the user's question, matching it with the parsed data from the wiki to find the most relevant information via RAG.
4. **User continues the conversation or continues on with their gameplay:** Follow-up questions and more detailed responses available if needed.

UI/UX

- **Chat Window:**
  - **[P0 Standalone Mode]:** A separate web app that users can open alongside their game or on their phones.
  - **[P2 Overlay Mode]**: An in-game overlay that allows users to interact with RIKU without leaving their game.
- **Input Methods:**
  - **[P0 Text Input]**: A simple text box for users to type their questions.
  - **[P1 Text Input]:** Incorporate autocomplete suggestions to help users form their queries.
  - **[P1 Voice Input]**: For hands-free interaction, include a microphone button to enable voice queries. Use speech-to-text technology to transcribe the queries.
- **Response Display:**
  - **[P0] Text Responses:** Display concise, clear answers in the chat window. Highlight key information and use bullet points for readability.
  - **[P1] Visual Aids**: Include images, maps, or charts when relevant (e.g., location maps, item stats).
- **User Feedback:**
  - **[P1] Error Reporting:** Provide a simple way for users to report incorrect or unhelpful information.
  - **[P2] Rating System:** Allow users to rate the helpfulness of responses to gather feedback and improve the AI.

Technical considerations

- **To build our prototype…**
  - **Data Sources:** Prototype to leverage Fandom for specific game titles
    - Future: Gamefaqs, IGN, and full Fandom wiki scrapes for wider set of game titles and franchises; support for custom inputs (e.g., Meta tier list for that month for game XYZ)
    - Tools to do this: Beautiful Soup, WikiExtractor
  - **Architecture:** Prototype to leverage ChatGPT + Langchain + Streamlit
    - ChatGPT models for embeddings, summarization, query parsing, etc.

    - Future: Voice models. Fine-tuning to video game queries.
- LangChain to handle vector embeddings, conversational logic chains, and data access.
    - Future: Cloud vector storage solutions like Pinecone for scale and deployment.
- Streamlit to simplify front-end and event-handling.
    - Future: React app for greater interactivity and flexibility.

List of Prioritized Features

Priority	Feature	Requirement	Considerations
P0	**Scrape and Collect Game Data**	**Scrape, parse, and organize information for select titles**	Use web scraping tools like Beautiful Soup; ensure data is structured and accessible for AI via RAG.
P0	**Accept User Questions**	**Allow users to input questions about specific game details**	Develop a front-end interface with a text input box; incorporate autocomplete suggestions for queries.
P0	**Generate Answer**	**AI processes user questions to provide relevant answers**	Integrate ChatGPT; fine-tune on game-specific datasets; ensure real-time response generation.
P0	**Text Input**	**Simple text box for user questions**	Ensure user-friendly design; incorporate autocomplete suggestions for ease of use.
P0	**Text Responses**	**Display concise, clear answers in the chat window**	Highlight key information; use bullet points for readability.
P1	**Voice Capabilities**	**Enable voice queries through a microphone button**	Implement speech-to-text technology for transcribing queries; ensure hands-free interaction.
P1	Scrape Entire Libraries	Increase library of games by scraping more repositories	Expand scraping capabilities to cover a broader range of video game information sources.
P1	Learning-capable Bot	Improve AI performance over time	Implement machine learning algorithms to enhance AI's ability to answer queries accurately.

P1	Recommendations	Provide strategies and build recommendations	Base recommendations on game performance data or expert inputs; integrate AI for personalized advice.
P1	Visual Aids	Include images, maps, or charts when relevant	Develop a system to display visual aids alongside text responses; ensure clarity and relevance.
P1	Error Reporting	Allow users to report incorrect or unhelpful information	Create a simple feedback mechanism; ensure user feedback is processed to improve AI accuracy.
P2	Spoiler Prevention	Ensure spoilers are not revealed	Implement algorithms to detect and avoid spoilers; allow users to set preferences for spoiler alerts.
P2	Multilingual Support	Provide support for multiple languages	Integrate language translation capabilities; ensure accurate responses across different languages.
P2	PC / Steam Integration	Provide game-specific data via desktop app or in-game overlay	Develop integration with platforms like Steam; create an in-game overlay for seamless interaction.
P2	Context-aware Responses	Enhance AI to understand the current game state	Develop AI capabilities to provide contextually relevant responses based on the game state.
P2	Rating System	Allow users to rate the helpfulness of responses	Implement a rating system to gather feedback; use data to improve AI performance and user satisfaction.

What is the acceptance criteria?

1. **Accuracy:** The scraping tool must accurately collect and parse data from Fandom wikis with an accuracy rate of at least 90%.
2. **Responsiveness:** The system must accept and process user questions within 5 seconds of input.
3. **Clarity**: Answers must be clear, concise, and well-formatted, using bullet points where applicable, especially given proper nouns and non-standard English.