Sparky

A friendly Steam assistant here to help you out!

Created by Dylan Jones, Natalia Makarewicz, Sanya Verma

UXD 225 Coding Design Frameworks Fall Quarter 2025

Introduction

Abstract

Sparky is an AI-powered virtual assistant designed to revolutionize game discovery on the Steam platform. The project addresses a critical gap in Steam's current recommendation system by leveraging artificial intelligence and Steam's comprehensive APIs to deliver personalized game suggestions that truly align with user preferences. Unlike Steam's existing Discovery Queue, which often fails to accurately capture user interests, Sparky analyzes multiple data points from a user's Steam history – including playtime statistics, game completion rates, and gaming patterns – to generate tailored recommendations. By combining Steam's robust API ecosystem with Microsoft Copilot's AI capabilities, Sparky creates an intelligent recommendation engine that understands not just what games users own, but how they actually play them.

The system features Steam credential authentication for seamless integration, the ability to leverage both individual user data and global gaming trends, and an optional survey system that allows users to refine recommendations based on specific preferences such as price range, content ratings, platform compatibility, and estimated completion time. This multi-layered approach ensures that recommendations are both personalized and practical, helping users discover games they'll actually enjoy while avoiding the frustration of irrelevant suggestions that plague the current discovery systems.

Design Problem

Steam's Discovery Queue represents a well-intentioned but fundamentally flawed approach to game recommendation. Users frequently report receiving suggestions for games that bear little resemblance to their actual gaming preferences, resulting in wasted time scrolling through irrelevant titles and missed opportunities to discover games they would genuinely enjoy. The current system appears to rely heavily on broad categorical matching and trending titles rather than deep analysis of individual play patterns and preferences. This creates a disconnect between what the algorithm thinks users want and what they actually play, leading to user frustration and decreased engagement with Steam's discovery features.

The problem extends beyond simple genre matching. Steam's library contains hundreds of thousands of titles spanning every conceivable genre, playstyle, and quality level. Without intelligent filtering that considers factors like typical session length, achievement-hunting behavior, multiplayer preferences, and content maturity preferences, users are overwhelmed by choice while simultaneously struggling to find games that match their specific interests. Additionally, technical considerations such as system requirements and platform compatibility are often overlooked in recommendations, leading uninformed users to games they cannot even run on their hardware.

Sparky solves this problem by implementing a comprehensive, AI-driven analysis system that examines multiple dimensions of user behavior. Rather than relying solely on genre tags or popularity metrics, the assistant analyzes hours spent on specific games to identify true preferences, recognizes player archetypes (Achievers, Explorers, Socializers, Killers, Speedrunners, or Casual players), and incorporates optional survey data to refine recommendations based on practical constraints like budget, available gaming time, and hardware capabilities. This holistic approach ensures that every recommendation is not just theoretically interesting but practically suitable for the individual user.

Research

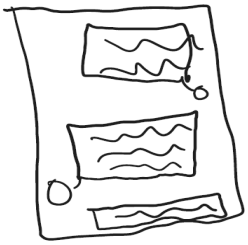
The development of Sparky required extensive research into both AI platforms and API ecosystems. For the AI component, we evaluated several platforms including OpenAI's GPT models, Google's Gemini, Anthropic's Claude, and Microsoft Copilot. Each platform offered distinct advantages: OpenAI's GPT excels at natural language understanding and conversational interactions, Google Gemini provides strong multimodal capabilities, and Claude offers detailed reasoning capabilities. However, we ultimately selected Microsoft Copilot as our AI platform due to its robust integration capabilities, strong performance in data analysis tasks, and ability to process structured data from APIs effectively. Copilot's architecture is particularly well-suited for synthesizing multiple data streams such as user game libraries, playtime statistics, and trends into personalized recommendations.

The API research phase focused primarily on Steam's comprehensive Web API documentation available through their developer portal. Steam provides several critical APIs for this project: the SteamUser API enables secure authentication and retrieval of user-specific data including game libraries and play history; SteamUserStats API provides detailed statistics about gameplay patterns, achievement progress, and completion percentages; and SteamNews API offers information about trending and newly released titles. We also considered the ISteamApps interface for accessing the complete Steam catalog and game metadata. These APIs were chosen because they provide official, supported access to the exact data points needed for intelligent recommendation generation: playtime per game, friend activity, achievement completion rates, and global popularity metrics.

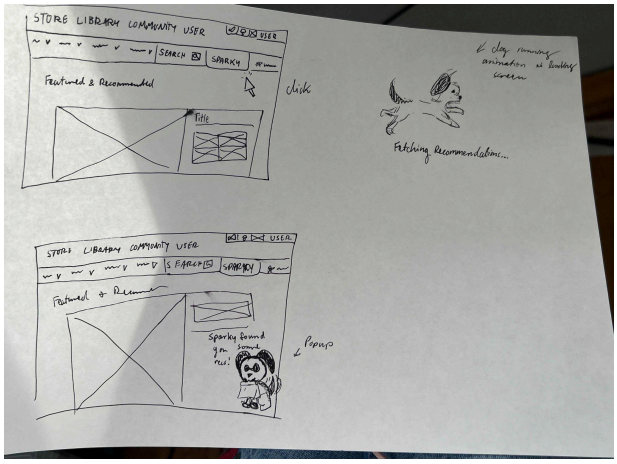
Additional research explored complementary APIs that could enhance the recommendation system. The "Can You Run It" concept mentioned in our documentation suggests integration with system requirement checking services, though this would likely require custom implementation or third-party hardware detection libraries. We also investigated Steam's store API endpoints for retrieving pricing information, user review aggregations, and detailed game specifications including genre tags, content descriptors, and platform compatibility. The combination of these data sources creates a comprehensive foundation for the AI to make informed recommendations that consider not just preference matching but also practical factors like affordability and technical compatibility. The official Steam Web API documentation (https://steamcommunity.com/dev) served as our primary technical reference, ensuring that all planned integrations follow Steam's official guidelines and terms of service.

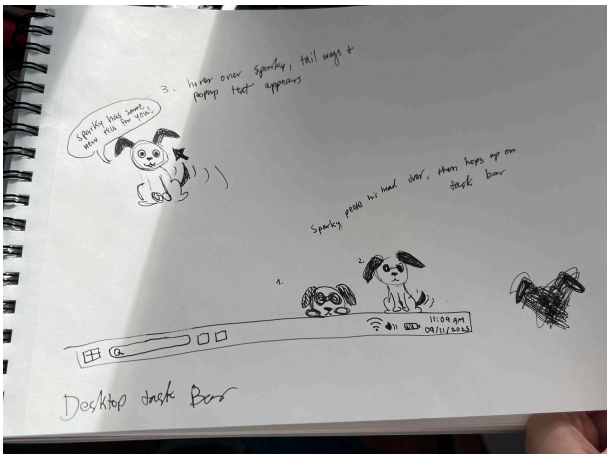
Low-Fidelity Prototypes

Dylan:

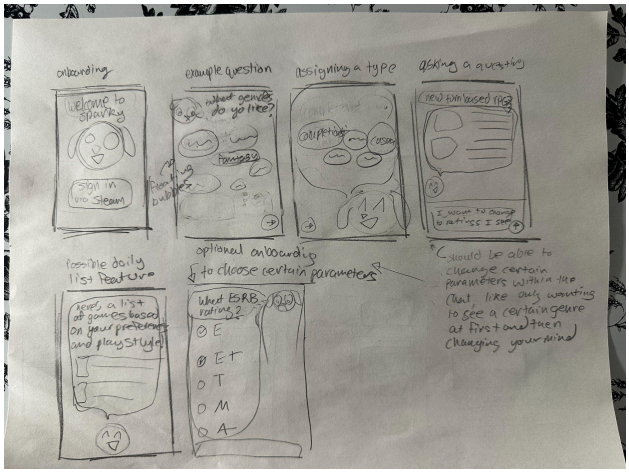


Natalia:



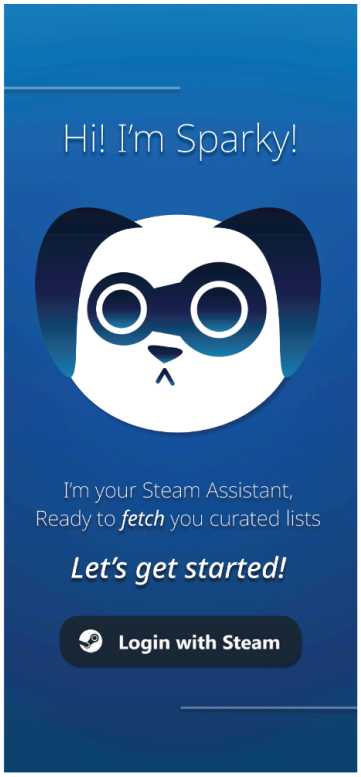


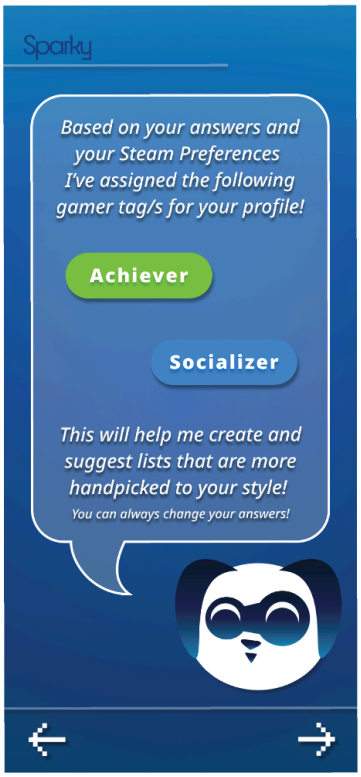
Sanya:



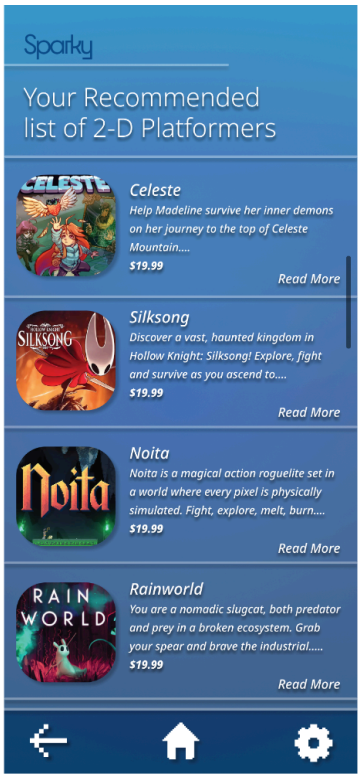
Mid/High-Fidelity Prototypes

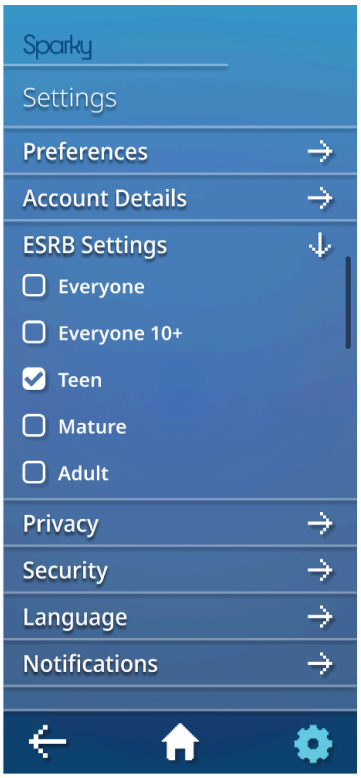
Onboarding





Getting a list, game listing and settings





Conclusion

AI and API Usage

The integration of Microsoft Copilot with Steam's API ecosystem creates a powerful tool that directly addresses the design problem of inadequate game recommendations. Copilot's natural language processing capabilities enable conversational interactions where users can describe their gaming preferences in plain English rather than navigating complex filter systems. When a user asks for recommendations "similar to" a specific game or requests titles matching a particular mood or playstyle, the AI can interpret these nuanced requests and translate them into meaningful data queries across Steam's APIs. The SteamUser and SteamUserStats APIs provide the raw data – what games users own, how many hours they've invested, which achievements they've earned, while Copilot transforms this data into actionable insights about user preferences and gaming patterns.

The true power of this AI-API combination lies in pattern recognition and personalization at scale. At the end of the day, pattern recognition and token prediction is what LLMs do by design. By analyzing playtime distribution across a user's library, the AI can identify which games genuinely captured their interest versus titles that were briefly tried and abandoned. Cross-referencing this data with SteamUserStats information reveals player archetypes: users who pursue every achievement are likely Achievers or Completionists, those with extensive multiplayer hours in social games fit the Socializer category, and players who rapidly complete games and move on might be Speedrunners. The AI can then query Steam's broader database through various APIs to find games that match these identified patterns, considering factors like similar gameplay mechanics, matching difficulty curves, comparable time-to-completion metrics, and appropriate content ratings.

The optional survey system amplifies the AI's effectiveness by providing explicit constraints and preferences that complement the behavioral data. When users specify price ranges, preferred platforms (Windows, Mac, Linux, Steam Deck), or content maturity preferences, the AI can filter its recommendations to ensure practical compatibility. This prevents the frustration of being recommended games that users cannot afford, cannot run on their hardware, or find objectionable due to content concerns. The combination of implicit behavioral analysis and explicit preference declaration creates a recommendation engine that understands users better than they might understand themselves, surfacing hidden gems that match their playstyle while respecting their practical constraints.

Furthermore, the AI's ability to use global user data through Steam's APIs enables sophisticated collaborative filtering. By identifying users with similar play patterns and library compositions, Sparky can recommend titles that "users like you" have enjoyed, expanding beyond the limitations of pure content-based filtering. The SteamNews API integration ensures that recommendations stay current, alerting users to new releases or trending titles that align with their preferences. This dynamic approach prevents the recommendation engine from becoming stale or repetitive, continuously adapting as both the user's preferences evolve and Steam's catalog expands.

Conclusion

The Sparky project is definitely feasible with current technology and existing APIs. Steam's Web API provides comprehensive access to all necessary user data and game information, while Microsoft Copilot offers mature AI capabilities sufficient for parsing this data and generating intelligent recommendations. The core functionality – authentication, data retrieval, pattern analysis, and conversational recommendation delivery – can be implemented today without requiring any technological breakthroughs. Similar recommendation systems exist across various platforms (Netflix, Spotify, YouTube), demonstrating that the fundamental approach is proven and reliable. The primary challenges are implementation-focused rather than conceptual: properly handling Steam's authentication flow, efficiently processing large user libraries, and fine-tuning the AI's recommendation algorithms to achieve optimal accuracy.

However, certain advanced features outlined in the documentation would benefit from future technological improvements. Real-time hardware compatibility checking would require either integration with third-party system detection services or development of custom hardware profiling tools, both of which present technical and privacy challenges. The in-Steam overlay feature would necessitate approval and cooperation from Valve, potentially requiring native Steam client integration rather than a purely web-based solution. Additionally, while current AI technology can make recommendations based on explicit preferences and play patterns, truly understanding the subjective "feel" of games (what makes one title's gameplay loop satisfying versus another's tedious?) remains an area where human curation still outperforms algorithmic analysis. Despite these limitations, the core Sparky concept is fully realizable with today's technology, offering immediate value to Steam users while leaving room for enhancement as AI and API capabilities continue to evolve.

Sources

Steam Web API Documentation
https://steamcommunity.com/dev

SteamUser API Interface Documentation
https://partner.steamgames.com/doc/api/ISteamUser

SteamUserStats API Interface Documentation
https://partner.steamgames.com/doc/api/ISteamUserStats

Steam News API Documentation
https://partner.steamgames.com/doc/webapi/ISteamNews

Microsoft Copilot
https://www.microsoft.com/en-us/microsoft-copilot

Steam Store Web API
https://wiki.teamfortress.com/wiki/User:RJackson/StorefrontAPI