Final Project Report

Generated on: December 19, 2025

1. Conclusion & Research Outcomes

The League of Legends Champion Recommender System successfully fulfills its primary aim of calculating appropriate champions for novice players through a sophisticated website-based machine learning model. By rigorously addressing the study's research objectives, the project demonstrates a clear methodology for mapping abstract player behavioral and psychological traits directly to concrete in-game preferences. Through the analysis of questionnaire data, the system effectively bridges the gap between soft personality characteristics-such as risk tolerance or strategic thinking-and hard gameplay statistics like utility and durability scores.

To forecast the most appropriate champions, the study implemented and validated an Ensemble machine learning approach. This 'Panel of Experts' strategy, combining Random Forest, Decision Tree, and K-Nearest Neighbors, proved to be far more effective than any single technique, achieving a high precision rate of 93.8%. This success confirms that integrating diverse logical models is the optimal way to connect distinct personality profiles with the complex, multifaceted nature of League of Legends champions. Ultimately, the website meets all its design and research objectives, delivering a fast, private, and accurate tool that empowers players to find their perfect match while validating the theoretical link between human psychology and digital gameplay.

2. Limitations

While effective, the current system has specific boundaries that define its scope:

- Static Data Updates: The system relies on a fixed database of champions. League of Legends changes every two weeks (patches), shifting the power of champions significantly. Currently, these changes must be manually entered, meaning recommendations might sometimes be based on outdated 'meta' information.
- Browser Resource Limits: Because the application runs 100% in the user's browser, it is limited by the device's memory and processing power. While fast now with ~170 champions, adding complex features like match history analysis or neural networks could slow it down, especially on older mobile devices.
- Self-Reporting Bias (Cold Start): The system relies entirely on what the user *says* they like via the questionnaire. If a user misunderstands their own playstyle (e.g., they think they play safely but actually play aggressively), the recommendations will match their answers, not their actual behavior. It cannot 'see' how they actually play.
- Subjectivity in Design: Assigning values to abstract concepts like 'Strategic Thinking' or 'Team Reliance' involves human judgment. The current scoring model assumes certain champions fit these traits based on design intent, but individual player experiences may vary.

League of Legends Recommender System - Project Summary

3. Future Work & Recommendations

To evolve this project into a professional-grade product, the following steps are recommended:

- Automated Data Pipeline (Riot API): Instead of manual updates, the system should connect directly to Riot Games' official API. This would allow it to automatically fetch the latest champion stats, difficulty ratings, and new releases the moment a patch goes live.
- Hybrid Intelligent Filtering: The system currently uses 'Content-Based Filtering' (matching attributes). Adding 'Collaborative Filtering' would make it smarter by looking at patterns: 'Players who liked Ahri also enjoyed Lux.' This requires a backend database to anonymously store and compare user choices.
- Deep Learning Integration: Moving beyond standard algorithms to simple Neural Networks could help capture non-linear relationships. For example, realizing that a preference for 'High Mobility' often correlates with 'Low Durability' in complex ways that simple rules might miss.
- Modern Component Architecture: Migrating the code from a single large file to a modern framework like React or Vue.js would make it easier to maintain. It would allow for reusable UI components (like a 'Champion Card') and better state management, making the code cleaner and easier for other developers to contribute to.
- Feedback & Reinforcement Learning: Implementing a 'Thumps Up/Down' feature on recommendations would allow the system to learn from its mistakes. If it recommends a champion and the user rejects it, the system should slightly adjust its internal weights to avoid similar mistakes in the future.