Mythopoly Project Backlog

Detailed Development Plan for PFE Validation

Prepared for Project Review

Date: July 15, 2025

1 Overview

This document presents the development backlog for the *Mythopoly* project, a multiplayer board game developed using Unity and Mirror, with Firebase integration for data persistence and a Node.js-based matchmaking system. The backlog is organized into three categories: tasks completed (Done), tasks currently in progress (In Progress), and critical tasks to be completed (To Do) to meet the engineering project requirements for PFE validation.

2 Done

The following tasks have been successfully completed across the initial sprints, establishing the core gameplay mechanics, multiplayer functionality, and data persistence.

Table 1: Completed Tasks

Sprint	**Tasks Realized**
Sprint 1	• Prototype of the 3D game board • Local player movement • Animated dice roll
Sprint 2	• Integration of Mirror networking • Multiplayer lobby creation • Player movement synchronization • Networked turn-based system
Sprint 3	• Property purchase/sale system • Money management • Rent and debt payment mechanics
Sprint 4	• Integration of two simple mini-games • Loading via additive scenes • Networked score synchronization
Sprint 5	• Firebase integration • Player authentication • Saving wins/losses • Data retrieval at game launch

3 In Progress

The following tasks are currently being developed, focusing on matchmaking, testing, and user experience enhancements.

4 To Do Critical for PFE Validation

The following tasks are planned to elevate the project to an engineering level, addressing modularity, networking, security, and performance optimization.

Table 2: Tasks in Progress

Sprint	Tasks in Progress
Sprint 6	• Setup of Node.js matchmaking server • Display of available lobbies • Player reconnection system
Sprint 7	• Writing initial unit tests • GitHub Actions pipeline for builds • Network debugging log analysis
Sprint 8	• Endgame screen with final scores • Addition of sounds and user feedback

Table 3: Critical Tasks to Plan

Technical Sprint	Critical Tasks
Modular Mini-Game Engine	• Create modular system with JSON/ScriptableObject loading • Integrate Factory/Strategy pattern • Enable adding mini-games without core modification
Advanced Networking	• Implement rollback/prediction (buffer, replay) • Handle position desynchronization via server • Log game states server-side
Intelligent Matchmaking	• Calculate ELO or ratio per player • Filter games by skill level • Prioritize fast lobby matching
Security and Persistence	• Secure Firebase with advanced rules • Add API calls via Node.js proxy • Manage authentication tokens
Advanced CI/CD	• Full GitHub Actions integration: build, test, push • Achieve >60% test coverage • Linting and analysis with SonarCloud or equivalent
Performance	• CPU/GPU profiling with Unity Profiler • Object pooling • Reduce draw calls and compress assets