LUDO GAME APP

Bug Fixing, Feature Development & Backend Modernization

Project Context and Required Outcomes

Layer	Present Context	Required Outcome
Unity client	Existing released or release-ready game app	Stable new features, polished UX, optimized performance, compatible builds
Node.js	Existing real-time multiplayer code; 1v1 already available	Secure scalable game server with 2P/3P/4P/team modes, load balancing, observability and load-test evidence
Laravel	Existing admin panel, APIs and record persistence	Rebuilt/upgraded supported codebase, versioned APIs, roles, reports, migrations and maintainable structure
Data & economy	Coins, rewards, purchases, inventory and tournament records	Server-authoritative, auditable and idempotent transactions with safe reconciliation
Delivery	Multiple new modules plus present bugs	Phased, testable releases with source code, documentation, QA evidence and handover

AI Opponent

Objective: Extend the existing AI gameplay into balanced multi-opponent configurations without breaking current 1v1 gameplay.

INCLUDED SCOPE

1. Preserve and stabilize the existing 1v1 AI implementation.
2. Implement 1-human versus 2, 3 and, where the final supported player-cap rules permit, 4 AI-opponent configurations.
3. Create difficulty levels and balancing rules for token selection, risk, capture, safe-zone use, home entry and end-game decisions.
4. Prevent predictable or unfair behavior and ensure AI turns do not freeze, skip or create invalid game states.
5. Integrate AI mode selection, result handling, rewards and relevant analytics into the existing app flow.

Play with Friend

Objective: Deliver reliable private-room gameplay for individual and team-based friend matches.

INCLUDED SCOPE

6. Private-room creation and joining for 2-player, 3-player and 4-player matches.
7. Invite flow using room code/link or the approved in-app invitation method.

8. Team Up invitation, team selection and Team Mode gameplay rules.
9. Room ownership, ready state, start rules, player leaving, timeout, reconnection and rematch behavior.
10. Server-side validation so unauthorized users cannot enter, manipulate or reuse rooms incorrectly.

Online Multiplayer and Player Search

Objective: Provide fast, secure matchmaking and correct real-time gameplay for public online matches.

INCLUDED SCOPE

11. 2-player and 4-player matchmaking, with expandable architecture for 3-player mode if approved during discovery.
12. Player-search screen, search timeout, cancellation, fallback and match-found confirmation.
13. Team selection and team-based matchmaking where applicable.
14. Fix current server-side room, search, synchronization, disconnection and result-related issues.
15. Secure all game events through server-side authorization, validation, rate limiting and anti-abuse controls.
16. Implement reconnect/resume and safe match termination rules for network interruption or abandoned games.

App Launch Screen

Objective: Create a configurable launch experience that presents promotions, rewards and advertising without blocking normal entry into the game.

INCLUDED SCOPE

17. Banner advertisements and promotional banner management.
18. Special Offer tab, Free Coin Offer and Daily Reward entry points.
19. Forced-ad flow with configurable frequency, eligibility, fallback and failure handling.
20. Remote configuration through the approved backend/admin flow where operational control is required.
21. Duplicate reward, repeated-popup and ad-loading error prevention.

Free Coins, Events and Mobile Number

Objective: Deliver one unified tab for free-coin actions, events and optional mobile-number collection.

INCLUDED SCOPE

22. Free Coin button, eligibility rules, cooldown/limit rules and secure wallet updates.
23. Event listing, event details, participation, status and result display.
24. Optional mobile-number input without OTP verification, with validation, consent text and secure storage.
25. Single unified tab and navigation flow covering all three functions.
26. Backend/admin controls required to create, publish, edit, pause or close events and offers.

In-App Purchase and Ludo Inventory

Objective: Create a secure unified shop for currency packs and player-owned cosmetic inventory.

INCLUDED SCOPE

27. Coin packs and any approved gem packs using Google Play Billing.
28. Server-side purchase verification, transaction recording, duplicate-credit prevention, acknowledgement and error recovery.
29. Unified shop UI for dice skins, emoji items, token skins, avatar skins, coins and gems.
30. Owned-item inventory, equip/unequip behavior, item availability and purchase requirements.
31. Laravel admin management for catalog, pricing references, visibility, stock/status where relevant and purchase reports.

BS Inventory

Objective: Implement the existing business requirement for managing physical or promotional items such as mobile phones and headphones.

INCLUDED SCOPE

32. Laravel admin module to upload and manage item name, images, description, quantity/status and other approved fields.
33. Node.js and Laravel integration required to expose the approved inventory data and actions to the game app.
34. Unity screens/components required to list and display items and complete the approved user action.
35. Input validation, authorization, record history and consistent stock/status behavior.
36. Final claim, redemption, order or participation workflow to be confirmed during discovery without weakening the agreed module objective.

All Tournament-Related Implementation

Objective: Complete, secure and optimize the tournament system, including server-authoritative coin-based rewards.

INCLUDED SCOPE

37. Tournament creation, configuration, scheduling, eligibility, joining, match assignment, result recording and completion flows required by the approved tournament format.
38. Coin-based reward distribution using an idempotent transaction ledger so retries or duplicate events cannot credit a reward more than once.
39. Node.js server-side game and reward implementation with Laravel record persistence, reporting and administrative controls.
40. Authorization, server-side result validation, abuse prevention, audit trail, reconciliation and dispute-support records.
41. Optimization of tournament queries, jobs, real-time events and reward processing for peak participation.
42. Fix all currently identified tournament bugs and any regression caused by the related implementation.

Friends Tab

Objective: Add the social relationship and invitation functions needed for repeat friend gameplay.

INCLUDED SCOPE

43. Friend list with relevant player identity and presence/availability state supported by the final architecture.
44. Send, receive, accept, reject and cancel friend requests.
45. Remove friend and handle invalid, duplicate or stale requests.
46. Invite an eligible friend to an approved game mode or room.
47. Backend storage, privacy-safe search and rate limits to prevent spam or enumeration.

Game Optimization and Player Experience

Objective: Improve responsiveness, stability and perceived quality across the complete game.

INCLUDED SCOPE

48. Reduce gameplay lag, unnecessary network traffic, frame drops, memory pressure and loading delays.
49. Optimize real-time event frequency, payload size, serialization, reconnection and state recovery.
50. Profile Unity scenes, assets, animations, object lifecycle and device-specific performance.
51. Polish movement, dice, token, transition, win/lose and other approved animations.
52. Improve UX feedback for loading, searching, reconnecting, waiting, failure and success states.
53. Add actionable crash/error logging and performance monitoring suitable for production support.

Design and UI Polish

Objective: Create and integrate the new UI required by all modules while maintaining a consistent Ludo visual system.

INCLUDED SCOPE

54. New screens, dialogs, tabs, states and components required for the approved modules.
55. Minor UI changes across existing areas affected by the new flows.
56. Responsive behavior for supported screen sizes and safe areas.
57. Consistent typography, spacing, colors, buttons, feedback states and reusable components.
58. Client review of wireframe/design direction before final implementation where a new screen Testing, QA, Revisions and Delivery Buffer

Objective: Validate the combined system, resolve defects and support phased client review through release.

INCLUDED SCOPE

59. Functional, integration, regression, multiplayer, device, network-interruption, security and performance testing.
60. Test matrix covering Unity app, Node.js game server, Laravel admin/API, database, billing, wallet, tournament and inventory interactions.
61. Structured bug tracker with severity, owner, status, evidence and retest result.
62. Client review cycles and reasonable revisions required to match the approved scope and acceptance criteria.

63. Release-candidate build, deployment rehearsal, production release support and post-release verification.
64. Delivery buffer for cross-module fixes and integration issues discovered during final QA.

Node.js High-Scale Modernization and Load Balancing

The team must not treat the existing Node.js server as a feature-only codebase. It must be converted into a measurable, supportable and scalable real-time platform suitable for a large player base. Final technology choices must follow the code audit and be approved before implementation.

REQUIRED ENGINEERING WORK

65. Upgrade to a client-approved supported Node.js LTS runtime and compatible maintained dependencies after a dependency-risk review.
66. Refactor blocking, duplicated or high-cost code paths and remove memory leaks, event-listener leaks and unbounded in-memory state.
67. Separate transient game/session state from individual server instances where required so multiple instances can safely serve concurrent matches.
68. Implement horizontal scaling behind a load balancer, including the correct shared-state, pub/sub or socket-adapter approach for the actual real-time framework.
69. Use connection handling, sticky-session behavior or a stateless alternative only where technically required and documented.
70. Add database indexes, connection pooling, caching, queues/background jobs, retry controls and backpressure where the audit shows a measurable need.
71. Implement graceful startup/shutdown, health checks, readiness checks, restart behavior and zero/low-downtime deployment procedures.
72. Add structured logs, metrics and alerts for concurrent connections, active rooms, matchmaking time, event-loop delay, CPU, memory, errors, disconnects, latency and transaction failures.
73. Run load, stress, soak and failure-recovery tests against an agreed concurrency and match-volume target; provide scripts and raw result summaries.
74. Document infrastructure topology, scaling rules, environment configuration, estimated operating cost and rollback procedure for the client-approved hosting provider.

Laravel Redevelopment and Data Migration

The Laravel work is a controlled redevelopment or major upgrade, not only a collection of patches. The team must preserve required business records while replacing unsupported or fragile structure with a clean, documented and testable codebase.

75. Use a client-approved current supported Laravel version with a compatible PHP runtime and maintained packages.
76. Rebuild or refactor the admin panel and APIs using a clear modular structure, service boundaries and consistent validation/error handling.
77. Version external/mobile APIs and preserve backward compatibility during the agreed migration window.

78. Implement secure authentication, role-based access control, admin activity logging and least-privilege permissions.
79. Create reviewed database migrations, seed/reference data where required, index strategy, backup and rollback procedures.
80. Implement queues/scheduled jobs, notifications, reports, exports and background processing required by the approved modules.
81. Provide automated tests for critical APIs, wallet/reward behavior, billing records, tournaments, inventory and administrative permissions.
82. Migrate existing data without silent loss, duplicate credits or broken references; provide reconciliation results after migration.

Module-Wise Timeline

The developer team must complete every blank field below. Quoted amounts must cover all work required for a production-ready module, including backend, Unity integration, database, testing, fixes,

#	Module	Quotation Coverage	Duration
1	**AI Opponent**	Multi-bot logic and balancing	_____
2	**Play with Friend**	2P/3P/4P rooms, invites and Team Mode	_____
3	**Online Multiplayer + Search**	Matchmaking, search, security and server fixes	_____
4	**App Launch Screen**	Ads, offers, free coin and daily reward	_____
5	**Free Coins + Events + Mobile**	Unified tab, wallet and event/admin flow	_____
6	**IAP + Ludo Inventory**	Billing, shop, skins, coins/gems and ownership	_____
7	**BS Inventory**	Physical/promotional item admin and integration	_____
8	**Tournament Implementation**	Lifecycle, reward ledger, security and fixes	_____
9	**Friends Tab (Phase 2)**	Friend lifecycle and game invites	_____
10	**Optimization & Experience**	Unity/network/server performance and UX	_____
11	**Design / UI Polish**	New UI and minor cross-app changes	_____
12	**Testing, QA & Buffer**	Regression, revisions and release support	_____
13	**Node.js Modernization**	Scaling, load balancing, monitoring and load tests	_____
14	**Laravel Redevelopment**	Supported version, APIs, admin and migration	_____