Treasure Hunt Game Management System - Technical PRD

1. Overview & Objectives

Purpose: Enable admins to configure & run live, real-world treasure hunts; give teams unique, randomized routes via QR-scanned checkpoints; automatically score & rank teams in real time.

Primary Goals:
1. Admin Experience: Create & manage MCQs, puzzles (with optional images), checkpoint sequences, scoring rules and team accounts-all fully dynamic.
2. Team Experience: Simple login -> "Scan QR" -> answer MCQ -> receive puzzle clue -> next checkpoint -> repeat -> see live timer & point tally -> finish.
3. Real-Time Monitoring: Admins view each team's current checkpoint, elapsed times, points earned, and overall leaderboard, updated instantly.

2. Actors & Key User-Stories

Admin:
- CRUD MCQs with per-option point values.
- CRUD puzzles (text + optional image) and link to unique codes.
- Pre-create team credentials.
- Configure total checkpoints, base points, and time-based bonuses/penalties.
- Launch game: randomize each team's checkpoint roadmap.
- Monitor real-time dashboard with live updates.

Team:
- Login, scan QR, answer MCQ, receive puzzle, proceed to next checkpoint.
- See real-time timer, checkpoint progress, points earned.

3. Functional Requirements

Admin Panel:
- Authentication: Hard-coded super-admin.
- MCQ Manager: Question, options, points per option.
- Puzzle Manager: Text + image, unique code, QR generation.
- Teams Manager: Pre-create login credentials.
- Global Settings: Base points, bonus/penalty per minute.
- Event Launcher: Randomized roadmaps per team.
- Live Dashboard: Real-time team progress and rankings.

Team Interface:
- Login, dashboard with timer, scan QR, submit MCQ, receive puzzle.
- Only correct QR for current checkpoint allows progress.
- Tracks checkpoint times and cumulative scores.

4. Data Model (Firestore)

/settings/global: N_checkpoints, base_points, bonus_per_minute
/teams/{teamId}: username, passwordHash, roadmap[], currentIndex, totalTime, totalPoints, legs[]

/mcqs/{questionId}: text, options[{text, value}]
/puzzles/{puzzleId}: text, imageURL?, code
/admins/{adminId}: username, passwordHash

5. System Architecture

Frontend: Vite + React + TypeScript, TailwindCSS, react-qr-reader, Firebase Auth & Firestore
Backend: Firebase Cloud Functions
Database: Firestore (NoSQL, real-time listeners)

Key Cloud Functions:
- startGame(): randomizes team roadmaps
- submitCheckpoint(): verifies QR, logs times, computes points
- fetchNextPuzzle(): returns puzzle data after MCQ submission

6. UI Flow & Wireframes (Conceptual)

Admin:
- Tabs: MCQs, Puzzles, Teams, Settings, Live Monitor, Launch Game

Team:
- Login -> Dashboard -> Scan QR -> MCQ -> Puzzle -> Next

Real-time progress and responsive layout for mobile and desktop.

7. Non-Functional

- Mobile-first and desktop responsive
- HTTPS enforced via Firebase
- Real-time updates through Firestore
- Suited for moderate scale (tens to hundreds of teams)
- No GDPR or export/reporting requirements

8. Timeline (Proposed)

Week 1: Setup, Auth, schema, basic routes
Week 2: Admin CRUD (MCQ, Puzzles), QR generation
Week 3: Team flow, checkpoint submission logic
Week 4: Admin dashboard, real-time sync, responsive polish
Week 5: UAT, deployment