PROJECT SHOWCASE / ENGINEERING & PRODUCT PORTFOLIO

MathQuest Arcade

Technical Case Studies

Prepared for professional review | November 4, 2026 | English edition

Evidence note: Based on prior project discussions, described architecture, selected development/debugging history and portfolio records available at preparation time. This is a portfolio case study, not a fresh source-code audit, external security review, market study or verified commercial due diligence. Planned capabilities and unverified outcomes are labeled accordingly.

MathQuest Arcade | Technical Case Study

Educational logic / math games | Stage: Playable mini-game work and extensive visual design; full catalog and polish evolving

Technical summary

MathQuest Arcade is a web portal for mathematical reasoning and logic mini-games, built around an illustrated medieval tavern hub. Users select an avatar and access experiences such as Cows and Bulls and Hidden Tiles. The project emphasizes approachable learning-by-doing and a coherent visual world.

Problem and engineering context

Static math practice can feel abstract and repetitive. A game-based collection may encourage exploration, iteration and reasoning through bite-sized challenges without requiring a long learning session.

Technology stack

- React 19, TypeScript and Vite for the portal interface.
- Phaser 3 for interactive mini-game boards and game rendering.
- Responsive layered visual assets for mobile/desktop environments.
- Reusable pixel-art graphics, animation assets, avatars and multilingual Spanish/English UI.
- Docker or static front-end deployment is plausible in the project stack; exact production configuration should be confirmed.

System architecture and component boundaries

- Portal shell: screen progression, avatar selection and central tavern/game-selection hub.
- Mini-game registry: route game identifiers to independent state, components and assets.
- Rendering: Phaser for interactive boards while the surrounding interface can be React-driven.
- Localization: common labels, tutorials and validation messages separated from game rules.

- Asset composition: layered backdrops, frames, banners, carpet, lantern/candle animations and mobile-specific offsets.

Data model and domain contracts

- Cows and Bulls rules: secret 4-digit number with distinct digits; count correct positions (bulls) versus correct digits in wrong positions (cows).
- Guess history: attempts, validation results, win condition and optional time/score tracking.
- Hidden Tiles: game-specific tile or card state, moves, completion feedback and sprite rendering.
- Player avatar and language preferences; avoid overclaiming accounts or long-term progress unless verified.

Primary application flows

- Enter the illustrated world and choose an avatar.
- Select a game from the tavern hub, then navigate into its dedicated experience.
- Receive feedback after each move and complete a win state.
- Return to the hub without losing UI control or leaving panels disabled.

Engineering challenges and tradeoffs

- Layering mobile overlays and responsive imagery without obscuring controls.
- Sprite scaling, win-message layout and contrast across backgrounds.
- Proper cleanup of game overlays and restoration of hub clickability.
- Consistent rules and explanations in both supported languages.
- Accessible interaction for touch, keyboard and differing cognitive abilities.

Available evidence and limits

- Previous work describes a Cows and Bulls implementation, assets and hub/phone visual experiments.
- MathQuest Arcade repository and React/TSX files were discussed.
- An integrated production build, complete game catalog and evidence of learning outcomes have not been independently verified.

Recommended technical validation / next steps

- Extract each mini-game behind a small typed interface and test deterministic rules.
- Add accessible step-by-step tutorials, keyboard support and progress states.
- Audit responsive images and overlays across common viewport sizes.
- Instrument game completion, repeat sessions and user comprehension without collecting unnecessary child data.
- Develop educational efficacy testing before marketing claims about learning improvements.

Recruiter interview prompts

- How is each game isolated from the parent shell?
- How do you test Bulls and Cows rules and game-result consistency?
- How are touch, keyboard and overlay focus handled?

Disclosure: This report intentionally distinguishes observed discussions from proposed engineering improvements. Tests, benchmarks, revenue, user adoption and security compliance are not asserted without supporting evidence.