QUIZZOR: AI-Powered Interactive Quiz Platform

Technical Documentation | Team: Time Limit Exceeded - IIT Bombay Techfest 2025 | Submitted: December 11, 2025

1. Problem Statement & Solution

Traditional quiz platforms face critical limitations in scalability, AI integration, and feature comprehensiveness. Educators spend hours manually creating quizzes, while existing solutions rarely combine AI-powered generation, real-time competitions, and analytics in one unified platform.

Quizzor's Key Innovations:

- **Instant AI-driven quiz generation** from documents (PDF, PPTX, images) with intelligent content extraction
- **Unified platform** combining single-player practice, multiplayer competitions, and persistent leaderboards
- **Performance-optimized architecture** using Redis for sub-millisecond leaderboard access
- **Comprehensive analytics** tracking user performance, quiz history, and competitive rankings

2. AI-Powered Quiz Generation Architecture

Quizzor leverages **Google's Gemini 2.0 Flash** model through three intelligent generation modes optimized for different use cases:

Mode 1: Turbo Mode

Use Case: Text-heavy documents without images

Process: Content extraction → Text parsing → Direct API call

Performance: 3-5x faster than file-based processing (5-7 seconds typical)

Libraries: mammoth, pdf-parse, xlsx, officeparser

Mode 2: Slow Mode

Use Case: Documents with images, diagrams, visual content

Process: File upload → GoogleAIFileManager → Multimodal analysis

Advantage: Supports image-based questions and visual interpretation

Trade-off: Higher latency due to file upload processing

Mode 3: No-File Mode (Topic-Based Generation)

Use Case: Quick quizzes on any subject without document upload. User provides topic, difficulty, and question count for rapid quiz creation on general knowledge topics.

AI Integration Technical Details

- **Model:** `gemini-2.0-flash-exp` via `@google/generative-ai` SDK
- **File Manager:** GoogleAIFileManager for document processing
- **Rate Limiting Strategy:** API key rotation system to maximize free-tier usage (prevents exhaustion)
- **Parameters:** numberOfQuestions (5-50), difficulty (Easy/Medium/Hard), optional title & description

3. Technical Architecture

Frontend Stack

- **Framework:** React 18 with React Router DOM
- **Styling:** Tailwind CSS, Framer Motion animations
- **HTTP Client:** Axios for API communication
- **UI:** React Icons for iconography

Backend Stack

- **Runtime:** Node.js with Express.js
- **Auth:** JWT with bcrypt password hashing
- **Sessions:** Cookie-parser for secure tokens
- **Files:** Multer for uploads, custom parsers

Database Architecture

MongoDB (Primary):

- Users: Authentication, profiles, quiz history
- Quizzes: Metadata, questions, answers, timestamps
- Submissions: Responses, scores, completion times, analytics

Redis (In-Memory Cache):

- Leaderboard: Sorted sets for O(log N) rank updates
- Token Blocklist: JWT invalidation for logout

4. Key Technical Challenges & Solutions

Challenge 1: AI Response Latency Optimization

Problem: Direct file uploads resulted in 15-20 second response times.

Solution: Three-tier generation strategy with local text extraction achieved 3-5x speed improvement. Future: Server-Sent Events (SSE) for progressive question streaming.

Challenge 2: Real-Time Leaderboard Performance

Problem: MongoDB queries become O(N log N) with scale, causing latency spikes.

Solution: Redis Sorted Sets implementation provides O(log N) complexity with sub-millisecond leaderboard access, periodic MongoDB synchronization for persistence.

QUIZZOR: Technical Documentation (Continued)

Team: Time Limit Exceeded | IIT Bombay Techfest 2025

Challenge 3: Free-Tier API Rate Limiting

Problem: Gemini API limited to 15 requests/minute per key.

Solution: API key rotation pool with round-robin distribution, automatic failover on rate limits, cost-effective scaling without premium subscription.

Challenge 4: Multi-Format File Processing

Problem: Supporting PDF, PPTX, DOCX, XLSX, images required different parsing libraries.

Solution: Unified content extraction pipeline with format detection, library-specific parsers, normalized text output, comprehensive error handling.

5. Current Implementation Status

✓ Fully Functional Features

- AI Quiz Generation (all 3 modes operational)
- User Authentication with Redis token blocklisting
- Quiz Submission & Automatic Grading
- Analytics Dashboard with performance tracking
- Redis-Powered Leaderboards (sub-ms access)
- Multi-Format Document Support (PDF, PPTX, DOCX, XLSX, images)

Limitations & Known Issues

- Socket Implementation Pending (requires concurrent user testing)
- UI Refinement (color scheme and typography optimization in progress)
- API Key Management (manual rotation, needs automation)

6. Roadmap to Final Build

Phase 1: Performance Optimization

Target: 5x faster generation via SSE/WebSocket streaming

Impact: Perceived latency reduction from 7s to <2s initial response

Phase 2: Real-Time Multiplayer

Tech: Socket.io with room-based quiz sessions

Strategy: Redis pub/sub for cross-server synchronization, load testing with 100+ users

Phase 3: Advanced AI Features

- **Pinecone Vector DB:** Semantic quiz search and recommendations
- **LangChain:** Contextual suggestions based on learning patterns
- **Adaptive Difficulty:** ML-based question calibration

Phase 4: Credentialing & Gamification

- **Blockchain Certificates:** Immutable completion certificates (Ethereum/Polygon)
- **NFT Badges:** Achievement-based digital collectibles
- **Smart Contracts:** Web3.js for certificate minting

Phase 5: Production Hardening

- **UI/UX Polish:** Professional design system with WCAG 2.1 accessibility compliance
- **Horizontal Scaling:** Kubernetes deployment with auto-scaling Redis clusters
- **Monitoring:** Prometheus + Grafana for real-time performance metrics
- **Enterprise Features:** Organization accounts, bulk quiz management, API access

7. System Data Flow Architecture

```
            User Request → React Frontend → Express API Gateway
  ├→ AI Quiz Generation → Gemini API (3 modes) → Quiz Storage (MongoDB)
└→ User Actions → MongoDB + Redis Operations → Leaderboard Update (Redis)
                    → Response → Frontend Rendering
```

8. Conclusion

Quizzor demonstrates sophisticated AI integration with performance-critical architecture decisions. The three-tier generation system, Redis-backed leaderboards, and planned vector database integration showcase practical engineering and forward-thinking scalability, addressing real-world educational challenges through intelligent system design.

Technology Stack Summary

Frontend: React 18, Tailwind CSS, Framer Motion, Axios | **Backend:** Node.js, Express, JWT | **Databases:** MongoDB, Redis | **AI:** Gemini 2.0 Flash | **Planned:** Socket.io, Pinecone, LangChain, Web3.js