

# CAP4001 - Capstone Project Proposal Report

Individual Report

**Student Name:** Thokala Sravan  
**Student Register Number:** 22BCE9745  
**Programme:** Bachelor of Technology  
**Semester/Year:** Fall sem (2025-26)  
**Guide(s):** Saroj Kumar Panigrahy  
**Project Title:** A Real-Time Player Finding System

## Team Composition

Reg. No	Name	Major	Specialization
22BCE9357	Adnan Hasshad Md	CSE	Core
22BCE20420	Tatikonda Srilekha	CSE	Core
22BCE9911	Mayakuntla Lokesh	CSE	Core
22BCE9745	Thokala Sravan	CSE	Core

## Project and Task Description

### Project Summary

The project, "A Real-Time Player Finding System," is an MVP (Minimum Viable Product) social platform designed to help users find and connect with others for specific gaming activities. The app's purpose is to address the common problem of finding compatible teammates or opponents for online games. The MVP focuses on core functionality and a good UI, including user profiles, a public request board, a 1v1 and team finder. The approach is to build a modern, serverless application that prioritizes immediate, action-oriented requests.

### Individual Role and Tasks

As frontend developer, I will: (1) Configure React with Vite, establish component structure, setup Tailwind CSS, implement Wouter routing; (2) Design UI components using shadcn/ui, create forms for user input, develop layouts for profiles and request board; (3) Create landing page, design user dashboard, build player finding interface with filtering by skill and preferences, implement request board display; (4) Integrate TanStack React Query for data fetching, implement custom hooks, setup error handling and loading states, manage authentication state; (5) Connect WebSocket for live request updates, implement real-time notifications, integrate with backend APIs.

### Approach

Phase 1 (Week 1-2): UI/UX design, component library setup, design system creation. Phase 2 (Week 3-4): Page development (landing, dashboard, finder), forms and routing. Phase 3 (Week 5-6): Real-time features, request board updates, state management. Phase 4 (Week 7-8): Design refinement, accessibility, performance optimization, testing.

Outcome Matrix

Outcome	Plan for demonstrating outcome
a) Apply knowledge of mathematics, science, and engineering	Will apply software engineering principles and data structures for player discovery; utilize relational database theory; implement distributed systems patterns.
c) Design system to meet needs within realistic constraints	Will design comprehensive system architecture balancing feature completeness, performance, scalability, 8-week timeline, and free/low-cost cloud platform constraints.
d) Function on multidisciplinary teams	Will collaborate effectively with team members across frontend, backend, and project coordination roles; facilitate communication and teamwork.
e) Identify, formulate, and solve engineering problems	Will identify system bottlenecks, formulate solutions for challenges, and troubleshoot issues across multiple system components.
g) Communicate effectively	Will create comprehensive documentation, clearly communicate requirements and decisions, provide technical guidance, and maintain code documentation.
k) Use modern engineering tools	Will utilize React, Express.js, TypeScript, PostgreSQL, Drizzle ORM, WebSocket API, OAuth 2.0, and version control systems.

Realistic Constraints

**Time:** 8-week development cycle requiring prioritization of core MVP features. **Team:** 4-member team with varying expertise levels. **Resources:** Free/low-cost cloud infrastructure (Replit, Neon PostgreSQL). **Technical:** Real-time request board, player matching logic, WebSocket connectivity. **Scope:** MVP focus with core features (profiles, request board, 1v1/team finder). **Performance:** Browser compatibility, responsive design across devices, fast load times.

Engineering Standards

**Code Standards:** TypeScript strict mode, ESLint configuration, consistent naming conventions. **Database:** Normalized design (3NF), proper indexing, referential integrity. **API:** RESTful principles, HTTP status codes, request validation, comprehensive documentation. **Security:** Input validation, SQL injection prevention, OAuth 2.0 implementation, CORS security. **Testing:** Unit tests, integration tests, end-to-end testing, quality assurance. **Version Control:** Meaningful commits, branch management, code reviews. **Documentation:** API docs, architecture diagrams, code comments, user guides.