Guide Approval (initials/date): ______________

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

Provide the information below for each member of the project team. Include all project team members, not just those in your discipline or those enrolled for Capstone project. Please also include yourself!

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

Provide a brief (one or two page) technical description of the design project and your specific tasks, as outlined below:

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 the frontend developer, my primary responsibilities will focus on designing and implementing the user-facing application: (1) Configure React with Vite for optimized development and production builds, establish component structure following modern React patterns and best practices, setup Tailwind CSS for styling with custom theme configuration, and implement client-side routing using Wouter; (2) Design and build reusable UI components using shadcn/ui library, create form components for user input (profile creation, requests, filtering), develop layouts for displaying player profiles and request boards, and implement interactive modals; (3) Create a responsive landing page with feature showcase, design user dashboard with request management interface, build player finding interface with filtering capabilities by skill and preferences, and implement real-time notification center; (4) Integrate TanStack React Query for efficient data fetching and caching, implement custom hooks for form and data management, setup proper error handling and loading states, and manage user authentication state and persistence; (5) Connect WebSocket functionality for live request board updates and notifications, implement real-time notification display system, integrate frontend with all backend APIs, and conduct responsive design testing.

Approach

The frontend development will follow a component-driven approach across the entire project timeline: Phase 1 (Week 1-2) focuses on UI/UX design specification, component library setup using shadcn/ui, design system creation with consistent spacing and colors, and establishing responsive design breakpoints. Phase 2 (Week 3-4) involves core page development (landing page, user dashboard, player finder), form implementation for user input and preferences, basic routing setup with Wouter, and initial integration of TanStack Query. Phase 3 (Week 5-6) includes real-time feature integration with WebSocket, notification system implementation, state management optimization, and thorough testing of data fetching. Phase 4 (Week 7-8) focuses on responsive design refinement across all devices, accessibility improvements (WCAG compliance), performance optimization (lazy loading, code splitting), and final comprehensive testing.

Design Phases

Phase 1 (Week 1-2): Planning, requirements analysis, architecture design, technology stack selection, database schema planning. **Phase 2 (Week 3-4):** Core infrastructure development, API implementation, authentication setup. **Phase 3 (Week 5-6):** Real-time features implementation, request board logic, system integration. **Phase 4 (Week 7-8):** Testing, optimization, documentation, deployment preparation.

Outcome Matrix

Describe your plan to demonstrate each of the outcomes below:

Outcomes	Plan for demonstrating outcome
a) An ability to apply knowledge of mathematics, science, and engineering	Will apply software engineering principles to system design, utilize data structures and algorithms for player discovery, implement relational database concepts, and apply distributed systems patterns.
c) An ability to design a system, component, or process to meet desired needs within realistic constraints such as economic, environmental, social, political, ethical, health and safety, manufacturability, and sustainability	Will design comprehensive system architecture balancing feature completeness, performance, scalability, 8-week development timeline, resource constraints (free/low-cost cloud platform), and team expertise levels.
d) An ability to function on multidisciplinary teams	Will collaborate effectively with team members across frontend, backend, and project management roles. Will facilitate communication, coordinate work efforts, and ensure seamless integration across components.
e) An ability to identify, formulate, and solve engineering problems	Will identify system bottlenecks and performance issues, formulate solutions for real-time synchronization challenges, solve database scalability problems, and troubleshoot integration issues.
g) An ability to communicate effectively	Will create comprehensive technical documentation, clearly communicate architectural decisions and requirements to team members, provide constructive feedback, and maintain detailed code documentation.
k) An ability to use the techniques, skills, and modern engineering tools necessary for engineering practice	Will utilize React, Express.js, TypeScript, PostgreSQL, Drizzle ORM, WebSocket API, OAuth 2.0, Cloudflare R2, and professional version control systems (Git).

Realistic Constraints

Development Timeline: 8-week development cycle requiring careful prioritization of MVP core features over advanced features. **Team Resources:** 4-member team with varying expertise levels requiring clear role definition and effective coordination. **Infrastructure Costs:** Designed for efficient use of free/low-cost cloud resources (Replit deployment platform, Neon PostgreSQL). **Technical Complexity:** Real-time request board functionality, WebSocket connectivity, and third-party service integration. **Performance Requirements:** Application must support responsive design across multiple devices (mobile, tablet, desktop) with acceptable load times. **Scalability:** Initial MVP designed for small user base but with extensible architecture for future scaling. **User Base:** MVP targets initial user base with potential for growth requiring consideration in database design and API architecture.

Engineering Standards

Code Standards: Enforce TypeScript strict mode, ESLint configuration, consistent naming conventions, and proper code formatting. **Database Standards:** Implement normalized database design (3NF), proper indexing strategies, and referential integrity constraints. **API Standards:** Implement RESTful principles with proper HTTP status codes, comprehensive endpoint documentation, Zod schema validation, and versioning strategy. **Security Standards:** Implement input validation, SQL injection prevention through parameterized queries, OAuth 2.0 authentication, and CORS security headers. **Testing Standards:** Establish unit testing approaches, integration testing procedures, end-to-end testing scenarios, and quality assurance processes. **Documentation Standards:** Maintain API documentation, database documentation, architecture diagrams, and inline code comments. **Version Control:** Use Git with meaningful commit messages, proper branch management for features, and code review standards before merging.