

# CAP4001 - Capstone Project Proposal Report

Individual Report

<b>Student Name:</b>	Tatikonda Srilekha
<b>Student Register Number:</b>	22BCE20420
<b>Programme:</b>	Bachelor of Technology
<b>Semester/Year:</b>	Fall sem (2025-26)
<b>Guide(s):</b>	Saroj Kumar Panigrahy
<b>Project Title:</b>	A Real-Time Gaming Social Platform

## 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

A Real-Time Gaming Social Platform designed to help users find and connect with others for gaming activities. Users create profiles with gaming stats, achievements, and preferences, then discover and connect with teammates or opponents through smart search and filtering (by skill level, playstyle, language, game preferences). The platform enables real-time communication through text chat and voice channels with 100ms integration, supports game portfolios to showcase clips and achievements, and functions as a Progressive Web App (PWA) for native-like experience. Core features include: user authentication (Google OAuth, Firebase), smart player discovery with advanced filtering, game portfolios, real-time voice/chat communication, match requests, connection management, notifications, and a system-level voice overlay on Android.

### Individual Role and Tasks

As support team member, I will: (1) Assist with React component development, styling, forms, responsive design, UI testing; (2) Help with API testing, debug integration issues, support query optimization, contribute to system testing; (3) Maintain documentation, participate in code reviews, document features; (4) Assist with integration testing, end-to-end testing, performance optimization, bug resolution; (5) Support various development tasks as needed, help troubleshoot issues, contribute to project success.

### Approach

Phase 1 (Week 1-2): Support planning, assist architecture, setup environment. Phase 2 (Week 3-4): Assist development, support components, participate in testing. Phase 3 (Week 5-6): Support integration, assist testing, help optimize. Phase 4 (Week 7-8): Support QA, assist documentation, prepare completion.

## 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, 100ms SDK, Cloudflare R2 API, and version control systems.

## Realistic Constraints

**Time:** 8-week development cycle requiring prioritization of core features. **Team:** 4-member team with varying expertise levels. **Resources:** Free/low-cost cloud infrastructure (Replit, Neon PostgreSQL, Cloudflare). **Technical:** Real-time voice communication, multiple service integrations, database scalability. **Scope:** MVP focus with extensible architecture for future scaling. **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, Zod 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.