Guide Approval (initials/date): _____

# CAP4001 – Capstone Project Proposal Report

(Individual Report)

| | |
|---|---|
| **Student Name** | Tatikonda Srilekha |
| **Student Register Number** | 22BCE20420 |
| **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 | Role |
|---|---|---|---|
| 22BCE9357 | Adnan Hasshad Md | CSE | Project Manager & Technical Lead |
| 22BCE9911 | Mayakuntla Lokesh | CSE | Back-End Developer |
| 22BCE9745 | Thokala Sravan | CSE | Front-End Developer |
| 22BCE20420 | Tatikonda Srilekha | CSE | QA & Support Developer |

## Project and Task Description

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

### Provide a summary of the project

"A Real-Time Player Finding System" is an MVP social platform that connects gamers to find compatible teammates or opponents for online gaming. The platform addresses the common problem of discovering compatible players through user profiles, a public request board, and 1v1/team finder functionality. Users can post requests for specific games, view real-time player availability, and connect with others based on skill levels and preferences. The MVP prioritizes core discovery features with a clean UI, real-time notifications, and serverless architecture for scalability.

### Describe the specific role and tasks that you individually will be completing

**1. Testing Planning:** Create comprehensive test plan covering unit, integration, and end-to-end tests, define test cases and acceptance criteria.

**2. Frontend Testing:** Test UI components, form validation, responsive design across devices, user interactions, accessibility compliance.

**3. Backend Testing:** Test API endpoints, database operations, authentication flows, real-time WebSocket functionality, error handling.

**4. Integration & Bug Reporting:** Conduct end-to-end testing, identify and document bugs, verify fixes, performance testing.

**5. Support & Documentation:** Assist with development tasks, support problem-solving, contribute code comments, compile final project documentation.

### Discuss in detail the specific approach that will be used to complete your portion of the design

**Phase 1:** Test plan creation, test case development, setup testing environment and tools.

**Phase 2:** Component testing, API endpoint testing, database testing, unit test execution.

**Phase 3:** Integration testing, real-time feature testing, end-to-end user flow testing, performance testing.

**Phase 4:** Final QA validation, bug verification, documentation support, project completion testing.

### Describe the phases of the design process

**Phase 1 (Week 1-2):** Project planning, requirements analysis, database schema design with PostgreSQL, API architecture design.

**Phase 2 (Week 3-4):** Backend development with Express.js, database implementation with Drizzle ORM, API endpoint development, authentication setup.

**Phase 3 (Week 5-6):** Frontend development with React, real-time features with WebSocket, UI component creation, system integration testing.

**Phase 4 (Week 7-8):** Performance optimization, comprehensive testing, deployment configuration, documentation completion, QA validation.

## 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, relational database design, algorithms for player discovery, data structures, and distributed systems patterns for real-time functionality. |
| c) an ability to design a system, component, or process to meet desired needs within realistic constraints | Will design system architecture balancing feature completeness, performance, scalability, 8-week timeline, and serverless platform constraints (Replit, Neon PostgreSQL). |
| d) an ability to function on multidisciplinary teams | Will collaborate across frontend, backend, and QA roles, facilitate communication, coordinate efforts, and contribute to team success through defined responsibilities. |
| e) an ability to identify, formulate, and solve engineering problems | Will identify system bottlenecks, formulate solutions for real-time sync challenges, solve scalability issues, and troubleshoot integration problems during development. |
| g) an ability to communicate effectively | Will create technical documentation, API specifications, maintain code comments and documentation, and communicate design decisions clearly within the team. |
| 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, Git, and modern development workflows. |

## Realistic Constraints

**Development Timeline:** 8-week MVP cycle with feature prioritization.

**Team Resources:** 4-member team with defined backend, frontend, and QA roles.

**Infrastructure:** Serverless platform (Replit), Neon PostgreSQL, Cloudflare R2 storage.

**Technical Complexity:** Real-time request board, WebSocket connectivity, OAuth integration, database scalability.

**Performance:** Responsive design across devices, acceptable load times for concurrent player searches.

**Scalability:** MVP architecture with extensible design for future features and player base growth.

## Engineering Standards

**Code Standards:** TypeScript strict mode, ESLint configuration, consistent naming, code reviews.

**Database:** Normalized schema design (3NF), proper indexing on frequently queried fields, referential integrity.

**API:** RESTful principles, HTTP status codes, Zod schema validation, OpenAPI documentation.

**Security:** Input validation, SQL injection prevention, OAuth 2.0 authentication, CORS configuration.

**Testing:** Unit tests, integration tests, end-to-end tests for critical user paths.

**Documentation:** API documentation, architecture diagrams, database schema documentation, code comments.

**Version Control:** Meaningful commits, branch management, collaborative code reviews, Git workflow.