CS: Tactics Fullstack Project Outline

Alex Allen

Contents

[Project Architecture Overview 4](#_Toc206513223)

[Frontend Components 5](#_Toc206513224)

[Backend Components 5](#_Toc206513225)

[Database Design 6](#_Toc206513226)

[Game Logic & Real-Time Processing 6](#_Toc206513227)

[Authentication & User Management 6](#_Toc206513228)

[Stat Tracking & Persistence 6](#_Toc206513229)

[Deployment & Infrastructure 7](#_Toc206513230)

[Optional Features 7](#_Toc206513231)

[Testing & Quality Assurance: 7](#_Toc206513232)

[Defining Game Design Requirements 8](#_Toc206513233)

[Setup Development and Environment 8](#_Toc206513234)

[Bootstrap Backend Solutions 8](#_Toc206513235)

[Prototype Real-time Communication 9](#_Toc206513236)

[Design Database Schema 9](#_Toc206513237)

[Develop Core APIs 9](#_Toc206513238)

[Build Frontend Starter 9](#_Toc206513239)

[Establish Testing Pipeline 9](#_Toc206513240)

[Plan Deployment and DevOps 9](#_Toc206513241)

[Iterate & Expand Features 10](#_Toc206513242)

# Project Architecture Overview

**Frontend:** interactive user interface for gameplay, accounts, and stats display.

**Backend:** Game logic, user management, matchmaking, stat tracking, real-time communications.

**Database:** Persistent storage for users, match results, player stats.

**Deployment:** Cloud hosting, scalability, CDN, security.

# Frontend Components

**Technologies:** React/Vue.js (SPA), real-time websockets, CSS/Canvas/WebGL(for game rendering)

**Features:**

* + Landing page, login/register forms
  + Lobby: Create/Join games, chat, match listings
  + Game board/UI: Counter Strike maps, units, weapons, action buttons
  + Stats dashboard: Player profile, match history, rankings
  + Responsive Design: Desktop/Mobile
  + Real-time updates: Game state, opponent moves.

# Backend Components

**Technologies:** C# Language, ASP.NET Core, SignalR, Entity Framework Core, SQL Server/PostgreSQL/MongoDB, Identity Framework

**Features:**

* + Authentication & Account Management
  + User Data & Stats Storage
  + Lobby Logic & Matchmaking
  + Game Logic Processing
  + Real-time multiplayer sync
  + Stat Tracking & History
  + Admin Dashboard
  + Logging and Monitoring
  + Deployment/Dockers
  + Security

# Database Design

**Technologies:** PostgreSQL, MongoDB

**Tables/Collections:**

* + Users: ID, username, password hash, profile info, stats
  + Match History: game\_id, players, outcomes, moves, timestamps
  + Live Matches: current game states, player turns, time remaining
  + Leaderboards: ranking by stats
  + Chat Messages: game\_id/context, sender, content, time

# Game Logic & Real-Time Processing

**Server Side:** Authoritative State, anti-cheat, logical game simulation

**Client Side:** UI Rendering, input validation, smooth animations

**Sync:** Websocket updates push/pull, rollback/resolve for latency

# Authentication & User Management

**Password/Security:** Hashing/Salting

**Session Management:** JWT or server side sessions

**Profile Management:** Avatar Uploads, profiles, emails/utils

# Stat Tracking & Persistence

**Save Stats:** Update on match end, profile display

**Achievements:** Auto-update on milestones

**Leaderboards:** Real-time ranking updates

# Deployment & Infrastructure

**Cloud Hosting:** AWS/Azure/GCP, Docker Containers

**Domain/CDN:** For web traffic

**Scaling:** Load balancing, stateless backend for game server clusters

**Security:** HTTPS, rate limiting, anti-cheat, data integrity

# Optional Features

**Friends System:** Add, Chat, Invite

**In-game shop/skins:** cosmetic items

**Tutorials & Guides:** First-time prompts

**Analytics:** Usage engagement tracking

**Medals/Achievements:** unlock medals for milestones when playing

# Testing & Quality Assurance:

**Unit Tests:** API, game logic, UI

**Integration Tests:** end-to-end

**Load Tests:** Matchmaking/gameplay at scale

**Security Audits:** Auth, game logic, data flows

# Defining Game Design Requirements (5-6 Days)

Write out a full fledged and concise game documentation containing:

* + mechanics
  + rules
  + interfaces
  + victory conditions
  + player experience

Identify core features for your MVP (Minium viable product):

* + registration/login
  + matchmaking
  + game lobby
  + basic game environment
  + stats tracking

# Setup Development and Environment (4-5 Days)

* Install required tools: Visual Studio/VS Code, .NET SDK(.NET 6/7/8), SQL Server, PostgresSQL, or MongoDB
* Create version control (git repository on github/gitlab)
* Decide on project structure (solutions, projects or API, game logic, models, client)

# Bootstrap Backend Solutions (9-11 days)

* Scaffold ASP.NET Core Web API Project.
* Set up entity Framework Core and configure your chosen database
* Add basic models: User, Game, Match History
* Implement user authentication (ASP.NET Identity, JWT if needed)

# 

# Prototype Real-time Communication (9-11 Days)

* Create SignalR Hub for game and lobby.
* Test basic message broadcast between clients

# Design Database Schema (5-6 Days)

* Draw Schema for Users, Game, Match histories, etc
* Set up migrations with Entity Framework Core

# Develop Core APIs (18-21 Days)

* Account management: Register, login, user profile
* Game Management: Create/join game, submit moves, retrieve match history.
* Stats API: Fetch/Update player statistics, leaderboards

# Build Frontend Starter (9-11 Days)

* Setup project with react/blazor or other preferred
* Build login/register page, lobby, and basic game interface wired to backend

# Establish Testing Pipeline (5-7 Days)

* Unit Tests for backend logic (XUnit,NUnit with C#)
* Integration Tests for API endpoints

# Plan Deployment and DevOps (9-11 Days)

* Dockerize backend for easy cloud/VM deployment.
* Choose hosting platforms (Azure, AWS, DigitalOcean etc.)
* Set up CI/CD workflow for automated building and deployment.

# Iterate & Expand Features (Ongoing)

* Add advanced game logic, more efficient game entities, polish UI, implement advanced stat tracking and anti-cheat mechanics.