# PROJECT PROPOSAL

*Casino game - Craps*

Table of Contents

## Project Abstract

This project aims to develop a digital craps/crappless craps casino game that delivers an immersive and interactive gaming experience. The game will simulate real-world craps mechanics, including dice rolls, betting strategies, and payout systems, while incorporating engaging visuals and a user-friendly interface.

## Conceptual Design

The Digital Craps Casino Game is a virtual implementation of the classic casino dice game, designed for an engaging and realistic user experience. Any modern device (PC, Mac, or mobile) with an internet connection, a standard CPU/GPU, and at least 4GB RAM for smooth gameplay. JavaScript (for web-based), C# (for Unity), or C++ (for Unreal Engine) or maybe Python (For Pygames) **Unity Engine (C#)** – If 3D animations and physics-based dice rolling are required. **Matter.js or Box2D (JavaScript)** – For realistic physics-based dice rolling.

## Proof of Concept

https://github.com/Ben-O1/casino-craps.git

provides a basic text driven implementation of craps, no visual or animations yet.

## Background

The proposed digital Craps Casino Game is a software-based simulation of the classic casino dice game, providing players with a fair, engaging, and interactive experience. It aims to capture the excitement of real-world craps tables while enhancing accessibility through an online or downloadable platform. The game will feature realistic dice physics, multiplayer options, AI opponents, statistical tracking, and an intuitive betting system.

Several open-source craps game implementations exist, primarily as simple text-based or web-based simulations. Some notable examples include:

- **"Craps.js" (GitHub Repository)**
  - A JavaScript-based browser craps simulator that focuses on basic game logic without advanced graphics or multiplayer features.
  - **Potential Use:** The proposed game could utilize similar game logic but will be expanded with graphical interfaces, multiplayer capabilities, and an enhanced betting system.
- **"Python Craps Simulator" (Open-Source Project)**
  - A terminal-based craps simulation using Python, focusing on mathematical probabilities and game flow.
  - **Potential Use:** Parts of the probability calculations or logic flow could be referenced, but the proposed game will be built from scratch with more extensive UI and networking features.

This project seeks to **differentiate itself from both basic open-source implementations and commercial gambling-focused craps games** by offering a balance between realism, fair play, educational value, and entertainment. The goal is to **create a high-quality, non-exploitative, and engaging craps experience** for both casual and experienced players. Also

there is also a version of craps called crapless Craps which is essentially the same thing with multiple minor differences, which will also be implemented in my version.

## Required Resources

 **Craps Rules & Strategies:** Studying the official rules, bet types, payout structures, and player strategies.
 **Probability & Randomization:** Ensuring fair dice rolls using random number generation (RNG) methods.
 **Game Development Principles:** Understanding UI/UX best practices, game physics (for dice rolls), and multiplayer networking.
 **Security & Fair Play:** Researching methods to prevent cheating, such as cryptographic RNG and server-side validation.
 **Comparison to Existing Games:** Analyzing how commercial craps games handle user experience, engagement, and monetization.
Standard PCs / Laptops
 **Windows/macOS/Linux** for development.
 **VS Code or PyCharm** for Python-based development.
 **Unity Engine (C#) or Unreal Engine (C++)** for 3D game development.
 **Node.js or Django (Python)** for backend development.
 **Frontend:**
- JavaScript (React.js or Vue.js for web version).
- C# (Unity) or C++ (Unreal Engine) for advanced 3D implementations.

 **Backend:**
- Python (Django/Flask) or Node.js (Express) for API handling and multiplayer.

 **Database:**
- PostgreSQL or Firebase (for tracking game sessions, stats, and user profiles).

**APIs & Libraries**

- **Physics Engines:** Matter.js (for dice physics) or Unity's built-in physics.
- **Multiplayer Networking:** WebSockets (Socket.io), Photon Engine (Unity).
- **Random Number Generation (RNG):** Python secrets module, JavaScript crypto.getRandomValues(), or Random.org API for external validation.
- **Security:** SSL/TLS encryption, Firebase Authentication (for user accounts).

This project will primarily rely on standard CS lab resources but may require additional cloud hosting, physics engines, and game development tools. A discussion with the instructor is necessary to determine access to multiplayer servers, mobile testing devices, and potential GPU requirements for advanced 3D simulations.