**المحكمة الافتراضية**

فكرة **محكمة افتراضية** مستوحاة من واجهات لعبة مثل *Counter-Strike: Source* قابلة للتنفيذ وممتعة! أعطيك هنا خطة عملية ومفصلة لبناء المشروع (تقنية، هيكلية، تصميم واجهة، أمثلة شفرة، أدوات، ونصائح للنشر)، بحيث تقدر تبدأ فوراً وتشقّ الطريق من بروتوتايب إلى نظام قابل للتخصيص والنشر.

**1) ملخص الفكرة (هدف واضح)**

بناء تطبيق/لعبة تُمكّن المستخدمين من دخول «محكمة افتراضية» حيث:

* تظهر الواجهة كـ HUD وشاشات ومؤشرات شبيهة بالألعاب.
* يمكن إضافة/تغيير اللاعبين (قضاة، محامون، شهود، هيئة المحلفين).
* يمكن تخصيص الأحداث (قضايا، أدلة، مرافعات).
* يمكن تغيير الأماكن (غرف محكمة، قاعات استماع) والخصائص (وقت الجلسة، قواعد العرض).
* دعم وضع فردي (AI) و/أو متعدد لاعبين عبر الشبكة.

**2) اختيار المحرك والتقنيات (توصياتي)**

* **Unity (C#)** — أفضل توازن للـ UI/UX ثلاثي/ثنائي الأبعاد، أدوات واجهة سهلة، دعم واسع للمحاكاة والشبكات، مناسب إذا تريد واجهة شبيهة بـ game HUD. مكتبة كبيرة وموارد.
* **Unreal Engine (C++ / Blueprints)** — ممتاز للرسوم الواقعية إذا الهدف تجربة بصرية قوية. لكنه أثقل لتطبيقات واجهة-مركزة.
* **Godot (GDScript / C#)** — خفيف ومفتوح المصدر، جيد إذا تريد مشروع أصغر أو مفتوح المصدر.

**شبكات (Multiplayer)**

* Unity: استخدم **Mirror** أو **Unity Netcode**.
* Unreal: استخدم نظام الـ replication المدمج.

**قواعد بيانات / تخزين**

* محلي: ملفات JSON / SQLite.
* سحابي (اختياري للمستخدمين/جلسات): Firebase, Supabase, أو REST API مع PostgreSQL.

**3) هندسة النظام (مكونية)**

1. **Presentation / UI Layer**
   * HUD (اسم القاضي، حالة الجلسة، وقت، قائمة الأدلة، دردشة صوتية/نصية).
   * شاشات جانبية: ملف المتهم/المدعي، سجل الأدلة، خيارات الكاميرا.
2. **Game Logic / Simulation Layer**
   * إدارة الجلسة (State Machine: Pre-trial → Opening → Witness → Closing → Verdict).
   * قواعد المحكمة (قوانين العرض، اعتراضات، قرارات القاضي).
3. **Entities / Data Models**
   * Player (role, permissions, avatar, stats).
   * Case (id, title, evidence[], timeline[]).
   * Evidence (type: doc/video/audio, metadata).
4. **Scripting / Scenario System**
   * محرّك نصي/سلوكي يسمح بكتابة “قضايا” قابلة لإعادة الاستخدام (مثلاً JSON أو لغة سكربت بسيطة).
5. **Networking Layer**
   * مزامنة حالات الجلسة، صلاحيات، ونظام تصويت/تصريحات.
6. **AI / Bots**
   * سكربتات بسيطة لمحامين/شهود (tree of responses, probability-based).
7. **Editor Tools**
   * محرر قضايا داخل التطبيق أو أداة ويب لرفع/تعديل القضايا والخرائط.

**4) نموذج بيانات مثال (JSON)**

{

"caseId": "CASE-2025-001",

"title": "سرقة متهم",

"participants": [

{"id":"p1","name":"القاضي أحمد","role":"judge"},

{"id":"p2","name":"المحامي وليد","role":"defense"},

{"id":"p3","name":"المدعي عام","role":"prosecutor"}

],

"evidence": [

{"id":"e1","type":"image","title":"صورة كاميرا","path":"assets/evidence/cam1.jpg"},

{"id":"e2","type":"audio","title":"تسجيل","path":"assets/evidence/rec1.mp3"}

],

"timeline": [

{"stage":"opening","script":"المرافعة الافتتاحية..."},

{"stage":"witness","script":"السؤال للشاهد الألف..."}

],

"rules": {"objectionsAllowed": true, "maxSpeakingTime": 180}

}

**5) مثال شفرة بسيطة (Unity C#) — إدارة حالة الجلسة**

public enum CourtState { Lobby, Opening, Witness, CrossExamination, Closing, Verdict }

public class CourtSession : MonoBehaviour {

public CourtState state = CourtState.Lobby;

public float stateTimer = 0f;

void Start() {

LoadCase("CASE-2025-001");

}

void Update() {

stateTimer += Time.deltaTime;

// مثال بسيط للتحكم بالانتقال بين الحالات

if(state == CourtState.Opening && stateTimer > 60f) {

SetState(CourtState.Witness);

}

}

public void SetState(CourtState newState) {

state = newState;

stateTimer = 0f;

// اشعارات للاعبين / UI update

UIManager.Instance.UpdateStateUI(state);

NetworkManager.Instance.BroadcastState(state);

}

void LoadCase(string caseId) {

// تحميل JSON وتحويله إلى نموذج بيانات

}

}

**6) تصميم واجهة مستخدم (اقتراحات)**

* استخدم شبكات وعناصر HUD مشابهة: شريط علوي يظهر اسم القضية والوقت، يمين/يسار أشرطة جانبية للأدلة والقوائم، منتصف الشاشة المساحة ثلاثية الأبعاد للقاعة مع كاميرا قابلة للتحريك.
* أصوات نظامية (clicks, gavel strike) وFX عند اعتراض أو رفع الأدلة لإحساس اللعبة.
* أوفر وضع “المحاكاة” و«وضع التحرير» (Editor Mode) لتصميم قضايا بسهولة.

**7) أدوات إنشاء المحتوى وModding**

* صمم ملفات **Scenario JSON** سهلة التعديل.
* وفر محرر داخل التطبيق أو صفحة ويب لتحرير القضايا والخرائط.
* وثّق واجهة برمجة التطبيقات (API) للـ Mods (مثلاً كيفية إضافة evidence/custom roles).

**8) الشبكات وحقن الصلاحيات**

* صمم نظام صلاحيات: القاضي يمتلك صلاحيات التحكم (إيقاف الجلسة، قبول الأدلة)، المحامون لديهم حقوق عرض/طرح أسئلة، المشاهدون لديهم صلاحيات مشاهدة فقط.
* عند استعمال Mirror أو Netcode — أبقِ الحالة authoritative على الخادم لمنع التلاعب.

**9) الذكاء الاصطناعي (اختياري)**

* استخدم **Finite State Machines** أو **Behavior Trees** لسلوك الشهود والمحامين الآليين.
* يمكنك إضافة نموذج احتمالي للردود (مثال: احتمال قبول الاعتراض = دالة على القاضي/نوع الأدلة).

**10) تجربة المستخدم والخصوصية/الأخلاقيات**

* لاحظ أن محاكاة قضايا حقيقية قد تلامس موضوعات حساسة — ضع تحذيرات، سمح بإخفاء الأسماء/تفاصيل حساسة، واحرص على قوانين النشر والخصوصية.
* إذا هناك تسجيل صوتي أو فيديو، علّم المستخدمين بالحصول على موافقات.

**11) بروتوتايب فوري — ماذا نفعل الآن (خطوات عملية تبدأ بها اليوم)**

1. اختَر المحرك (أنصح Unity).
2. أنشئ مشروع Unity جديد (3D) وأبسيط واجهة HUD (Panel علوي، لوحة أدلة جانبية).
3. نفّذ نموذج بيانات JSON وبنية تحميله.
4. أضف مشهد قاعة بسيطة (platforms + camera) وحرك الكاميرا.
5. أنشئ state machine (الكود أعلاه) لتحكم بمراحل الجلسة.
6. صمّم واجهة لتحميل قضية JSON وتجربة المراحل داخليًا.
7. بعد نجاح البروتوتايب، فكر بالـ networking وEditor tools.

**12) مرفقات عملية: قائمة مكتبات/موارد مفيدة**

* Unity: Mirror (شبكات)، Cinemachine (كاميرات ديناميكية)، TextMeshPro (نصوص).
* أدوات صوت: FMOD أو Wwise (اختياري).
* Assets: نماذج قاعات من Unity Asset Store لتسريع التطوير.
* DB/Backend: Firebase / Supabase (لجلسات متعددة ومزامنة).

**13) مثال واجهة سيناريو قابل للتخصيص (نص مُقترح لمشهد)**

* شاشة عنوان كبيرة: "محكمة 3D — جلسة: سرقة متجر"
* HUD: الوقت المتبقي، اسم المتحدث، زر اعتراض (إذا كانت الصلاحيات تسمح)، قائمة الأدلة (تصغير/تكبير).
* نافذة جانبية: لوحة حالة القاضي، زر صدور حكم.

لو حابب، أقدر الآن:

* أكتب لك **ملف JSON جاهز** لقضية تجريبية تقدر تضعها في بروتوتايب.
* أزوّدك **مشروع Unity بسيط (Skeleton)** يتضمن HUD و state machine وقراءة JSON — أُقدّم شفرة كاملة قابلة للنسخ واللصق.
* أو أعمل **مخطط واجهة (wireframe)** بصري يوضح موقع العناصر.