

Arian Hassani

Game & Full-Stack Developer · Unreal Engine (C++), Unity (C#) · Node.js, TypeScript, React

Istanbul, Turkey · Open to remote and on-site roles, ready to relocate

arianhassani.business@gmail.com · arianhassani.dev

SUMMARY

Game and full-stack developer with seven years of experience and 40+ shipped projects. I have used C++ for client and personal projects since 2019, mainly in Unreal Engine 4 and 5 (C++ and Blueprints), plus Unity (C#), and spent two years as the sole game developer in a small fully remote studio, shipping 10 titles. My game work covers gameplay systems on the Gameplay Ability System, game AI (behaviour trees with C++ tasks, NavMesh, vehicle and pursuit AI), replicated multiplayer, editor tools and published FAB Marketplace plugins.

Alongside games I build the products around them: Node.js, TypeScript and NestJS backends deployed to AWS and Linux, React and Next.js web apps, React Native and native Android (Kotlin) mobile apps. I built and operated an authoritative multiplayer mobile game for a year in production on Linux dedicated servers with a match manager I wrote. I analyze scope at the start of a project, prototype the riskiest part first, and hold delivery to what was agreed.

SKILLS

Engines & Languages: Unreal Engine 4 & 5 (C++ and Blueprints) · Unity (C#) · C++ · C# · TypeScript · JavaScript · Python · Kotlin · Java · Swift

Gameplay Systems: Gameplay Ability System (custom AbilitySystemComponent and AttributeSet, ability tasks, event-driven damage flow) · weapon and ability frameworks driven by gameplay tags and effects · character abilities and movement · Enhanced Input · animation events (AnimNotify, AnimNotifyState) · UMG UI · data-driven design (DataAssets, DataTables) · actor component architecture · interfaces and delegate hubs

AI: behaviour trees with C++ tasks (police vehicle and pursuit AI) · enemy AI on NavMesh (cover-slot reservation, melee rush, strafing, randomised attack timing, accuracy and crit rolls, weighted wave spawning with a self-tuning difficulty ramp) · suspect AI with context-based decisions (comply, flee, resist) · vehicle AI (pathing, traffic navigation, pursuit, PIT, road blocks, last-known-location search) · crowd NPC AI · waypoint AI opponents (Unity) · utility-based AI known from self-study only, not used in production

Multiplayer & Networking: UE replication, server RPCs, RepNotify and ReplicatedUsing properties · authoritative dedicated servers on Linux · listen servers · Online Subsystem and session handling · Node.js server orchestration over WebSockets (spawning server processes on free ports, session routing, two-way control plane) · server-authoritative design (server-side shuffling, token-gated privileged messages) · deterministic simulation known conceptually only

Tools & Performance: custom C++ procedural generators with editor tools · Unreal PCG graphs · editor-facing plugins · FAB Marketplace plugins and templates · profiling and cull-distance tuning · async asset streaming · engine-level C++ debugging

Mobile: UE5 mobile (iOS and Android) · Android JNI plugins · mobile ad SDK bridges · native Android (Kotlin, Jetpack Compose, Java) · React Native with Expo · Google Play release

Backend, Web & Cloud: Node.js · NestJS · Express · MongoDB · PostgreSQL · REST · WebSockets · JWT and role-based access control · React · Next.js · MUI · AWS (ECS Fargate, ECR, ALB, ACM, CloudWatch) · Docker · GitHub Actions CI/CD · Linux VPS operations

Workflow: scope analysis at project start · prototype-first de-risking · documentation for other developers · Git and Git LFS · JetBrains Rider with a source-built Unreal Engine

PROFESSIONAL EXPERIENCE

Freelance Game & Full-Stack Developer

08/2019 – Present

Self-employed / Freelancer.com and Upwork · Remote

- Delivered 40+ projects from concept to shipping build for international clients, strongest in tools and gameplay programming: 2D puzzle and card games, RPG-style systems, FPS, battle-royale tooling, plus web, mobile and backend products.
- Shipped games in Unreal Engine (Blueprints and C++) and Unity, including mobile titles, multiplayer products, and published FAB Marketplace plugins and templates.
- Built backends and web products end to end: data model, REST and socket layers, deployment and production support, on Node.js, TypeScript, NestJS, React, React Native and native Android.
- Worked solo and in teams under contract and fixed-scope terms: analyzed scope at the start of each project, suggested cuts where they helped, and held delivery to what was agreed.

Unreal Engine Developer

03/2023 – 12/2024

More Than Just A Game · Remote (small fully remote studio)

- Two-year contract as the sole game developer in a studio of under five people, alongside a QA person, a backend developer and at times a designer. The studio started with casino games and shifted to casual games; almost all titles were Unreal Engine C++, a few were Unity C#.
- Sole developer across 10 shipped game projects, spanning casino and card games (blackjack, poker, slots) and casual titles.
- Built complete multiplayer products end to end: Unreal client, C++ gameplay, server-side replication and supporting backend, not just isolated features.

- Expanded the client's project templates and took them to production state; developed reusable Unreal tooling adopted across projects, including multiplayer session handling and shared gameplay systems plugins.

PROJECTS: GAMES & UNREAL ENGINE

Live Poker Madness

2024

Solo Developer · Unreal Engine 5, C++, Node.js, Express, MongoDB, Linux, WebSockets

Free-to-play mobile poker game with an original fold-or-all-in variant, built and operated solo end to end. UE5 C++ client shipped to iOS and Android, with a custom Android JNI plugin that calls the native gallery intent and returns the image to Blueprint as base64, used for avatars and KYC capture. A Node.js match manager I wrote runs the multiplayer lifecycle: it matchmakes players up an eight-step ladder, spawns Linux dedicated server instances on free ports (scanning TCP, UDP and TCP6 and handling the race between finding a port and another session taking it), routes players to the right instance, and tears everything down cleanly. Manager and game server talk over WebSocket in both directions: the manager stays authoritative over matchmaking, the game server over gameplay. The deck is shuffled server-side and pushed down, never trusted from the client, and every privileged message is admin-token gated. Express and MongoDB backend with JWT auth, payments, KYC onboarding and an admin panel. Ran a year in production with me as the only person on call, handling every incident.

UE5 Law Enforcement Simulation Prototype

2026

Client project · Unreal Engine 5, C++, Blueprints, AI Systems

PC prototype covering patrol, dispatch, arrest and pursuit gameplay, in C++ with Blueprints for UI and scripting: player character, vehicles, dispatch, suspect AI, arrest flow, a small district and basic HUD and audio, on Epic's City Sample (Small City) as the world base. Built suspect AI with cover-taking, evasive movement and context-based decisions (comply, flee or resist), and police vehicle AI with per-behaviour logic in C++ driven by a behaviour tree, covering pathing, traffic navigation and pursuit. The focus was believable, non-scripted behaviour driven by the player's actions and the scene state. Moved the work into a clean project with only the used assets migrated, under Git LFS.

Police Pursuit AI Prototype (technical assessment)

2026

Solo Developer · Unreal Engine 5.8, C++, Behavior Tree

One-working-day practical test built from scratch, primarily in C++: police pursuit AI structured as a Behavior Tree with C++ tasks, including the full optional bonus of pulling alongside, a PIT manoeuvre and a second unit blocking ahead. Delivered as a project, a video and a short plain-text README, and passed.

Project F (action roguelite prototype)

Started 2019 · modernised 06/2026 – Present

Solo Developer · Unreal Engine 5, C++, Gameplay Ability System

Isometric top-down action roguelite in UE5 and C++ on the Gameplay Ability System, built with a deliberately small scope: one shapeshifting weapon with several forms. Complete GAS foundation from scratch: custom AbilitySystemComponent and AttributeSet, event-driven damage flow, abilities including a poison cloud and a rotating laser, and ten enemy archetypes drawn from Persian mythology. Weapon forms share one data-driven framework through tags and effects instead of per-weapon branching. Debugged real engine-level C++ problems (WaitGameplayEvent listener leaks, shared AnimNotifyState caching, AbilityTask pending-kill crashes, bTickingTask constructor timing). Basic Blender and Mixamo art pipeline (a fully armored low-poly Persian warrior, about 4,900 triangles, flat-shaded, FBX straight to UE5), with code-driven VFX and shaders carrying the look.

Call of Honor

2024

Solo Developer · Unreal Engine 5, C++, AI / NavMesh, Mobile

Mobile-first wave-survival FPS built for touchscreens: raw touch input drives the camera with separate hipfire and aim sensitivity, every action is an on-screen button, and the whole UI exists twice, mirrored for left-handed players, with paired bindings. Enemy AI on NavMesh: ranged units reserve unoccupied cover slots so two never pick the same spot, melee units rush the player, and both strafe, randomise attack timing and roll accuracy and crits; some melee units apply damage over time. A weighted spawner shortens its interval and raises enemy speed each cycle, giving a self-tuning difficulty curve. Headshot slow-motion, an airstrike reward for ten headshots in a row, civilians that punish careless fire, night vision, HUD post-process volumes that swap on health thresholds, persistent stats, and explicit background and foreground handling in the game instance.

Superpowers Unreal Plugin

2024

Solo Developer · Unreal Engine 5, C++, Plugin Development, Replication

C++ plugin of nine independent, fully replicated superpower components (flight, globe shield, ice riding, invisibility, random teleport, scale adjustment, super jump, super speed and thinning), each an Actor Component that drops onto any Character with no base-class lock-in. Built for a client's multiplayer battle royale. One replication contract throughout: client input fires a Server Reliable RPC, authoritative state lives in ReplicatedUsing properties, OnRep handlers apply mesh, animation and movement changes on every client, Enhanced Input is decoupled through a shared delegate. The globe shield implements a damageable interface and breaks at a damage threshold; teleport validates destinations with a ten-attempt rejection-sampling loop (capsule sweep plus ground trace); scale adjustment replicates capsule size so collision stays correct. The client shelved the game before completion; the plugin layer was mine end to end. Authored two UE C++ tutorials on it.

Procedural City Generation (UE5 PCG and C++)

2026

Unreal Engine 5, custom C++ procedural generation, PCG graphs

Procedural district generator for the prototype above, on Epic's City Sample assets. Prototyped with Unreal's PCG graphs for roads, crossings, walls, props and buildings (randomly curved roads with straight lead-ins at crossings, no-build corridors, weighted building variety). Profiled a drop from 80 fps to 20-30 fps with buildings generated, traced it to the building scatter volume, and fixed the resulting shadow popping by tuning cull distances and exempting large shadow casters. Then built a custom C++ generation system in place of the PCG graphs: a shared generator base class plus five generator actors with tunable properties and editor Generate and Clear buttons.

Beat Dash

2025 – Present

Solo Developer · Unreal Engine 5, C++, DSP / FFT

Rhythm platformer built around the idea that any chosen song becomes the level. Wrote the audio-analysis pipeline from scratch in C++: Hann-windowed radix-2 FFT, band-limited spectral flux, peak picking against an adaptive threshold derived from each song's own flux statistics. Runtime decoder for WAV, MP3 and AAC, and an asynchronous HTTPS track-fetching service with progress reporting and an idle watchdog, backed by a Node.js service I host. Gameplay is spline-driven rather than physics-driven, so every jump is locked to the detected beats; subsystems coordinate through a static delegate hub.

Projects: Web, Mobile & Backend

Staff Management Suite

10/2025 – 04/2026

Solo Developer · Node.js, TypeScript, NestJS, MongoDB, React, MUI, AWS ECS Fargate

Internal company operations platform built solo as two linked web applications under one account system: people, availability, booking requests and calendar; and tasks, one-off and recurring payments with spend analytics, two cloud-storage products (media gallery and file store, with range-request video seeking and uploads up to 10 GB) and an events system. Four-role access model (Admin, Parent, Model, Guest) enforced server-side, with connection-scoped visibility and a delegation middleware that lets a parent act on behalf of a connected user. Migrated the backend from Express to NestJS (about 9,500 lines, 10 controllers, 14 models, handlers preserved with zero build errors), containerised it with a multi-stage Docker build, and deployed it to AWS ECS Fargate behind an ALB with ACM HTTPS, CloudWatch logging and a custom domain, with a GitHub Actions pipeline for the React frontend and MongoDB Atlas. Gathered requirements, designed the data model and API, and supported it in production.

Real-Time Messenger App

01/2025 – Present

Solo Developer · Node.js, TypeScript, MongoDB, WebSocket, Express, React Native, Expo, Stripe

Real-time mobile messenger with random-stranger matchmaking, username search, friends, public and private group chats, encrypted messaging, voice and video, and a trust-and-safety layer, over a raw WebSocket backend with a large admin and analytics panel. Designed the data model, the REST and socket layers and the event flow that keeps presence, typing, reactions and edits consistent across concurrent sessions. Matchmaking scores queued users by interest compatibility while honouring gender and age filters and excluding blocked pairs; random chats auto-archive unless both users friend each other; AES-CBC message encryption on the mobile client; video face verification with admin states, peer trust scoring and live enforcement of suspensions through the socket. Strict MVVM across a large React Native codebase.

Gotta Run (Android)

06/2026

Solo Developer · Kotlin, Jetpack Compose, Android

Task timer app published on Google Play with a sequential per-task countdown. Timers run in a foreground service so they stay accurate while the app is backgrounded or the device sleeps. Took it through the full Play Console release process: signing, policy declarations, store listing and rollout.

EDUCATION

Computer Engineering (incomplete)

08/2018 – 03/2022

IUST, Tehran, Iran · Left to pursue career goals

LANGUAGES

English: Fluent · Persian (Farsi): Native · Turkish: Elementary (A2), actively studying